THE DESIGN OF A MULTIMEDIA ADAPTIVE INTERFACE FOR PROCESS CONTROL USING A MULTI-AGENT APPROACH

by

Mr C.I.J. Khalil

A thesis submitted in partial fulfilment of the requirements for the degree of

Ph. D.

Loughborough University

2001

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I am indebted to:

Professor James Alty for his patience and wonderful advice.

My parents for always providing me with support and encouragement

EPSRC and ESPRIT for funding me.

The AMEBICA project partners for providing me with a platform for my work.

All my friends who had to put up with me.

QUOTATION

"The Outcome of any serious research can only be to make two questions grow where only one existed before."

Thorstein Bunde Veblen (1857-1929) US Social Scientist. *The Place of Science in Modern Civilisation.*

Chapter 1

INTRODUCTION

Introduction

This thesis develops an approach to improving the representation, form and timeliness of data in a complex process control interface. In traditional interfaces, a mapping is made at *design time* between the *process parameters* and an appropriate *rendering* at the interface. This mapping is usually the best all-purpose mapping under a set of general constraints. It is not, however, the only mapping – others may have been discarded which might have represented a better mapping under a different set of constraints. In the general case of the system functioning under normal conditions, the general mapping may be appropriate in most instances. However, if the process moves into a disturbed state, one of the other discarded mappings may be more appropriate for the new conditions. The goal of this thesis is to investigate if these other mappings can be implemented in a *flexible* mapping system, so that an adaptive system can make a decision on which mapping to use at run-time, based on the current state of process, the environment, the actions of the operator team, and access to a human factors database. Flexibility, for example, has been identified as one of Shackel's (Shackel 1990) requirements for good usability.

Why might Adaptation be Important?

In the modern control room, the traditional hard-desk approach has been replaced by a soft-desk approach (Dicken 1999). In this new approach, the operators monitor plant conditions on a large computer monitor, or over multiple monitors, but are usually unable to view all the process information simultaneously (which used to be possible in earlier systems through the use of mimic displays). The operator must therefore switch between different views as appropriate. A common type of representation, used in process displays, is the Piping and Instrumentation (P&I) diagram, which, whilst effective in communicating the relationships between elements, can present problems for the operators:

> The diagram is mainly concerned with the topology of the plant. Hence higher-level goals are not usually immediately available within the display.
>
> Typically a one-sensor-one-representation approach is generally used, so operators have to deduce the overall process state from an amalgamation of several individual indicators.
>
> The process diagram is often too large to be represented on a single display, and must therefore be spread over a number of monitors. This means the operator has to often hunt for the appropriate information over several screens, or abstract hierarchies.
>
> Automated systems have introduced many extra complexities for the operators. Before the operator needed to monitor, understand and reason about physical forms and functions in the process. Now they need to maintain an accurate view of the process at a higher level of abstraction.

The above problems originate because of a lack of flexibility at run-time (Shackel 1990). Of course the operators can use various functions, such as zoom and translate to adjust the display and they can demand more detailed information about a particular sub-function. However, the responsibility for doing this is laid totally on the operator and if the operators have missed some important process malfunction, or are persisting in following an erroneous logical path, the system cannot help them. A more adaptive system could assist the operators at run-time by accentuating the display of particular measurements, by tuning the display to show relevant areas of concern, or by forcing reappraisals of situations by presenting higher level deductions and predictions.

It is the aim of this thesis to provide a degree of adaptability at run-time in the selection and representation of bandwidth limited information to operators. The approach will hopefully ensure that the main presentation parameters of form, location, and modality correspond to the contents and nature of relevant information. It is hoped that this will lead to higher predictability of important process occurrences, less information searching, less screen cluttering, quicker response times and generally improved operator effectiveness

The proposed interface will introduce an element of system adaptability into a highly complex process control interface. In normal circumstances, it is envisaged that very little adaptability will be required at the interface. Indeed, since the operators are highly trained, expert users, any unjustified adaptation may well hinder their effectiveness. However, when the system moves into a disturbed state, the operators may benefit from a pro-active alarm handler. It is well known that such a shift is frequently accompanied by alarm flood (Bransby & Jenkinson 1998), and consequent operator information overload. In such situations, by assigning incoming alarm signals with levels of importance, using knowledge of the current environment, by understanding the state of the operators and human factors presentation heuristics, the proposed system may be able to select the most appropriate mapping. Such an appropriate mapping should enable the most salient information to be presented to the operators in a more effective way at the most appropriate time.

It is hoped that the adaptive system will bring the following advantages:

> *Redundancy:* Multimedia often communicates through more than one sensory channel. This allows the system to exploit the natural human ability to multitask.
>
> *Accentuation of critical data:* By highlighting and accentuating representations of the most critical data the system can draw the operator's attention from less important data. By guiding the operator in this way it is hoped that handling of dangerous situations can be improved
>
> *Appropriate Levels of Abstraction in the Interface:* An adaptive system brings a greater degree of flexibility to the interface. Information overload can cause the operator to "hunt" for

appropriate information through several hierarchical layers of interface. The adaptive system has the ability to abstract away unnecessary and irrelevant data, giving the operator are clearer view of the current problem. Information can be selective displayed in such a way that only salient information is displayed to a degree in line with its importance.

Improved Attention Getting: Since time is a scarce resource for process operators, particularly in critical situations, they are required to deal with information on a basis of priority. This is, of course, the reason why alarm systems feature prominently in most control rooms. However, more present day visual and audio alarms only provide a coarse gradation of the severity of the event that triggered them. An adaptive multimedia system may be used to ensure that information is brought to the operator's attention in a timely fashion. Critical information can, to a much greater extent, be presented in forms, which are more tailored to the process situation that is developing. Customised audio or spoken messages as well as visual animations, can all be used to attract the attention of the operator. Thus the intrusiveness of the alarm can be related to its severity in a more flexible way.

Spatial Adaptation: An adaptive system can organise the interface so that related data is grouped together, and allow the optimum amount of information to be displayed in clearest possible format. Avoiding windows overlapping allows the operator to have a clear understanding of what is happening at the interface without "losing" data when it is obscured or hidden.

Thesis Organisation

Chapter 2 provides an overview of adaptive systems in general. It examines why adaptive systems are useful and the goals and form an adaptive system should have. It then examines the general structure an adaptive system should take, and the types of models a system should maintain in order to be able to adapt in an appropriate fashion. Finally, the chapter examines how these models have been used in existing adaptation system and how adaptive systems can be evaluated.

Chapter 3 examines the current state of process control interfaces. An examination is made of the current problems with process interfaces, and suggests how such problems may be tackled by the use of multimedia principles.

Chapter 4 examines the technology of Software Agents and why they are appropriate for use in an adaptive system. It looks at the advantages agents provide, and the different types of agents that are in existence. Then, a review of current agent technology is undertaken to determine which types of agents are most appropriate for an adaptive system. Finally, the chapter looks at the difficult problem of locus of control. It arrives at a method for determining an agent's degree of responsibility so that it does not overstep its authority to the detriment of the operators and their confidence in it.

Chapter 5 looks at how agents can be combined into an effective multi-agent system, for use in the process control domain. It examines the benefits of a multi-agent approach and the relative advantages that such an approach would offer for an adaptive system. The chapter then looks at some of the important implementation issues involved in building a multi-agent system

Chapter 6 discusses the WHY and HOW of adaptation. It examines the concept of adaptation and proposes the adoption of an Adaptation Matrix to describe the broad principles of why adaptation should take place and what should trigger it. It also examines how multimedia heuristics can be applied to make presentations at the interface clearer. It then describes a set of guidelines and usage rules that can be applied to the adaptive system to ensure that it will always try and select the most appropriate representation, with the correct parameters at the right time.

Chapter 7 reviews the different technologies required to built an adaptive, multi-agent system. A detailed review is provided of agent technologies, coding technologies, code optimisation techniques and interface development tools. A recommended set of tools and system configuration options is provided.

Chapter 8 provides an overview of the proposed adaptive system's conceptual architecture. It describes the founding principles on which the system is based, and describes the nature of the system. It examines the roles of the agents within the system and describes precisely how they interact.

Chapter 9 gives a more detailed view of the core reasoning processes employed by the architecture to drive adaptation. The main reasoning agents are examined, and the process by which they decide upon the form of adaptation employed by the system, is described. An overview is given of how the adaptive system relates to process control problems.

Chapter 10 demonstrates a scenario employed on the adaptive system to illustrate its capabilities. This is then followed by some usability results obtained from testing the system on real operators, and experts.

Chapter 11 critiques the successes gained from building the prototype system; it also looks at the system's limitations. The successes and limitations of the actual prototype system built are detailed and lessons drawn. The problems encountered when constructing the system are also examined. The thesis concludes with a summary and suggestions for future work.

Contribution of the Author to the Work Reported

AMEBICA is a complex system and the design and development work was spread over a number of contributors in different commercial organisations. The author made a very significant contribution to the overall work described in the subsequent pages. A detailed analysis of the author's contribution will be found at the beginning of Chapter 10. It is left until this chapter because it is only at this juncture that the whole concept of AMEBICA can be fully appreciated and the author's contribution clearly described.

Chapter 2

ADAPTIVE SYSTEMS

Introduction

Adaptive Intelligent Multimedia Presentation Systems can be characterised by their capability for "designing presentations that express information using a combination of available presentation techniques and media, in a way which achieves the required communicative purposes and supports users in performing their tasks" (Roth 1993). In other words, they are capable of dynamically *adapting* their behaviour to the requirements of users, allowing users to make improved communication decisions *at run-time* (Dietrich et al 1993),

Why Is There A Need For Adaptive Systems?

Computer applications can be difficult to use and even though the procedures for using them have been learnt, they may still be easily forgotten. As the number of computer applications and different delivery platforms proliferate, and the number of users who need to use various computer applications in their daily work continues to grow, the chances of serious user misunderstandings at the interface have increased. It is even the case that experienced users have found themselves bewildered with the software industry's predilection for development of new features, bug-fixes and production of new (i.e. slightly different and 'improved') versions (Thimbleby, 1990b). Constant "improvement" often results in an interface complexity that is constantly in a state of flux. This has been particularly true in the process control domain where the move from hard desks to soft desks, has led to a glut of new features and increasingly complicated interfaces.

The idea that computer applications should be capable of adapting their interfaces to match the needs either of individuals or of different classes of users is an apparently attractive, one (Benyon & Murray 1993). Adaptive computer interfaces are therefore not new, and have been discussed over the years through a number of advocates (Edmonds 1981; Innocent 1982; Zissos & Witten, 1985). Other useful reviews are provided by (Norcio & Stanley 1989). Early applications of the adaptive system concept, however, did not live up to expectations and often presented designers with problems of a greater complexity than those encountered in non-adaptive solutions. There was also a fear that adaptation could introduce its own complexity and present the user with inconsistent interfaces.

These early attempts to create adaptive systems were certainly hampered by a lack of processing and I/O power, and it is only recently, now that powerful computers have become available, that consideration of how to build such systems has become of interest again. The development of multi-agent systems (Maes 1991; Laurel 1990) and intelligent interfaces (Chignell & Hancock 1988) has also focused attention on adaptive systems.

Although many design solutions are possible, computer applications tend to be based on one chosen design solution that is inevitably better suited to some users than others, and whose choice may be very dependent upon the designers' experience. It is a contention of this thesis that the usability of many systems could be improved if they were able to offer a different set of design solutions to match the diversity of user populations and run-time dependent interfaces generated in context.

A particular class of users to whom adaptability is important is what are termed *Discretionary Users*. Discretionary Users are users who can solve their problems, if necessary, without resorting to the use of a computer application (even though it may be less efficient). Such users must be persuaded, or otherwise enticed, into making use of the computer facilities. Even when enticed into using the system, such users tend to stick with what they know, and are usually not adventurous in seeking out new or improved ways of doing things. They often become reliant on the initial system features used. This type of behaviour is often true of *Novice Users* as well. For such users there is a need to provide effective methods for graduating to more efficient uses of the application.

Some designers do, of course, implement systems that can cater for different users, environments, working practices and different markets. Such personalisation is implemented through the use of 'customising' facilities or 'macros'. The problem with such approaches is that the users must learn these customisation features, and they detract from their primary goal of employing the application to solve their key needs. Tailoring facilities are typically very general. They frequently do not take fine-grained, individual, differences into account and do not always cater for a user's task needs (either perceived or implicit). Although users may eventually become committed to the use of some software, and so are no longer discretionary with respect to the system itself, they may still be discretionary with respect to customising the system to better suit their needs.

It is important to distinguish between user tailoring and adaptive systems. The former are user-driven whilst the latter should be system driven. This adds considerable complexity to the design since an adaptive system has to make decisions about when and how to adapt, whereas a customized system is driven solely by user decisions. Adaptive systems are therefore systems that can alter aspects of their structure, functionality or interface in order to accommodate the differing needs of individuals or groups of users and the changing needs of users over time (Benyon, Innocent and Murray 1987). Adaptive systems seek to take over the burden of tailoring systems to individuals, groups and situations in context, or to protect the user from arduous interface-induced complexity.

Adaptation in terms of the individual is considered to be a key principle in user interface design and evaluation, e.g.

> *'Dialogue systems are said to support suitability for individualization if the system is constructed to allow for adaptation to the user's individual needs and skills for a given task.'(Oppermann et al., 1992)*

In EVADIS II (Oppermann et al., 1992), this principle has been elaborated as follows:

> Parts of the dialogue that are developed with certain user characteristics in mind (such as normal colour vision or low level experience level) support individualization if they can be modified to support users who differ in these characteristics (such as colour blindness or high experience level).
>
> The user should be able to adapt the dialogue system to support his/her individual strategies of planning, problem solving, and information processing strategy.
>
> The dialogue system should allow for individual preferences with regard to structural, procedural, and physical aspects.
>
> The dialogue system should allow the user to choose among alternative forms of representation according to the complexity of the information to be processed.
>
> Explanations (e.g. error messages, help information) should adapt according to the individual level of knowledge of the user

Techniques such as *metaphor* and *analogy* can be employed to make system functionality more accessible to user populations, and it has often been claimed that these approaches can overcome many usage problems (Alty & McKell 1986, Carroll & McKendree 1987). However, such approaches are critically dependent upon a closely shared appreciation of the basis of the metaphor or analogy employed. Criticism of metaphor in interface design adds weight to this argument (Kay, 1989).

One important question, of course, is whether the goal of Adaptation is feasible, technically, operationally or economically. For example, Thimbleby (1990b) has argued that only complex systems can benefit from an adaptive capability but this very complexity means it is not possible to provide such a capability because the user patterns of usage be will harder to determine. He asserts that there is simply not enough bandwidth in the user interface to accommodate the required functionality for adaptation. Similar arguments can be made with respect to the cost of building an adaptive capability into an application and with the resulting technical problems such as processing speed, knowledge representation and so on. In addition, the capability of systems to incorporate enough suitable knowledge about an individual user in order to make sensible adaptive responses, and the basis on which such user characteristics can be inferred, has also been questioned (Kobsa & Wahlster 1989). However the truth of any of these arguments has not been theoretically proven. For example, a very simple adaptive mechanism may be highly effective in some circumstances, or the cost associated with the implementation of adaptive systems may still be justified if they significantly improve usability and the quality of interaction even if the cost is high (in process control this is particularly true). Furthermore, the inclusion of an adaptive capability may not be such a large overhead if it arises as a natural consequence of better attention and metrics being applied to interactive system design

Technical Definitions of Adaptivity

According to (Trigg et al. 1987) there are four ways in which a technical system can exhibit adaptability:

> 'A system is *flexible* if it provides generic objects and behaviours that can be interpreted and used differently by different users for different tasks.
>
> A system is *parameterised* if it offers a range of alternative behaviours for users to choose among.
>
> A system is *integratable* if it can be interfaced to and integrated with other facilities within its environment as well as connected to remote facilities.
>
> A system is *tailorable* if it allows users to change the system itself, say, by building accelerators, specializing behaviour, or adding functionality.'

Browne (Browne et al., 1987) views adaptive systems as technical 'systems that "behave" differentially, dependent upon the current user of the system'. The same authors give the following independent dimensions of adaptable systems:

Contextual adaptation allowing users to navigate gracefully through tasks

Changing levels of guidance / feedback, and

Changes based upon a user's knowledge of similar systems.'

(Browne et al., 1990) classify software as *adaptive* if it can change its own characteristics automatically (perhaps after user consultation), thereby adapting itself to the users' needs. They use the term *adaptable* in contrast, to denote the provision of end users with tools that enable the users to change the software features and eventually the behaviour of the application. Hence, *adaptability* can be considered as a prerequisite for achieving *adaptivity* and vice versa. However, in the literature, *adaptation* and *adaptivity* are often used synonymously.

Goals of An Adaptive System

The goals that the adaptivity process attempts to fulfil vary substantially in current systems, according to the requirements of the application and user group. (Dietrich 1993) provides the following list of adaptivity goals:

easy, efficient, effective use

make complex systems usable

present what the user wants to see

speed-up use

simplify use

provide a user interface that fits heterogeneous user groups

provide a user interface that considers the effects of increasing user experience

Additionally, one might state several other goals, such as:

minimise number of errors

maximise user satisfaction

minimise cost, in terms of computational resources

Components of an Adaptive System

Benyon and Murray (1993) describe a generalized architecture for adaptive systems as shown in Figure 1.

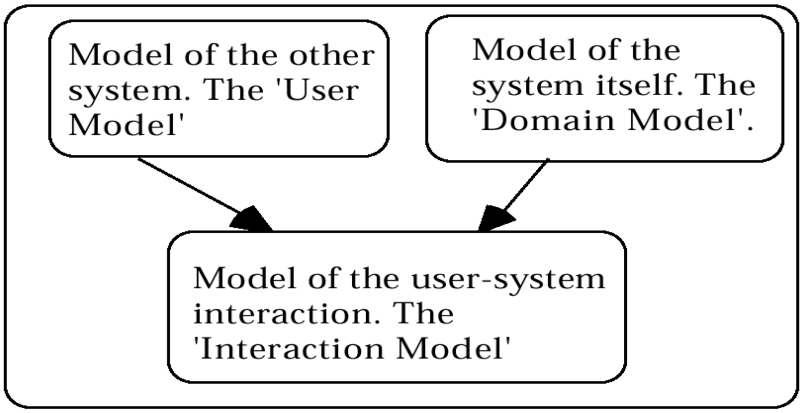

Model of the other system. The 'User Model'
Model of the system itself. The 'Domain Model'.
Model of the user-system interaction. The 'Interaction Model'

Figure 1 Overview of an adaptive system

This architecture contains a model of the user, system and system interaction. The adaptive system uses its knowledge of these domains to determine when adaptation is necessary and what form it should take. The adaptive system takes cues for when to adapt from the user or system, that match rules contained with the user/system models. The changes can be determined from utilizing its user/system interaction model as a means of capturing salient interactions between the two domains and characterizing them within the individual models.

The User Model

The user model captures what the system believes to be the knowledge and preferences of the user. It can be used to generate adaptability pro-actively when its knowledge of the users state appears to require adaptation, or by co-operatively interacting with the user to deduce when adaptation is required. To maintain an accurate model of the user it is essential that the user model monitors and samples user behaviour and keeps both a history of past actions and an up-to-date model of the current user state. It uses this data to attempt to deduce the users current goal, and then alters the system in some way so as to facilitate the achievement of that goal.

The knowledge represented in the user model may be acquired *implicitly* from inferences made about the user or it may be *explicitly* elicited from the user. Explicit acquisition may be achieved through some co-operative behaviour such as asking relevant questions in specific contexts.

Knowledge for the user model can be acquired implicitly by making inferences about users from their interaction, by carrying out some from of test, or from assigning users to generic user categories usually called 'stereotypes'. The simplest approach - the notion of "user stereotypes" derives from the work of Rich (Rich 1983; 1989). Stereotypes represent a structured collection of traits or characteristics, stored as facets, to which is attached a value, and optionally, a confidence-level and rationale. Some traits are triggers and have an attached probability rating that can mediate or inhibit firing of a whole stereotype. They can be used to

> '…*provide a way of forming plausible inferences about yet unseen things on the basis of things that have been observed' (Rich, 1983).*

Stereotypes model users on a variety of dimensions and represent characteristics of users in a hierarchy. At the top of the hierarchy is the 'any person' stereotype that defines characteristics relevant to all users of the system. Stereotypes at lower levels in the hierarchy may inherit the characteristics of this stereotype. Lower level stereotypes will depend on the application, but retain the property of inheriting characteristics from parents. At the bottom of the hierarchy is the individual who may inherit characteristics from a large number of stereotypes. One of the problems with such a representation is to decide just what happens when conflicting characteristics are inherited. Conflict resolution rules must then be included to deal with such situations.

Rich (1989) describes the space of user models using two dimensions. The first, canonical versus individual, describes whether the model is of one single user or a collection of models of different individuals. A canonical model represents the 'typical' user and is not usually stored explicitly in the system. It is the designer's model identified earlier. Rich also distinguishes long-term models from short-term models. Long-term models are representations of fairly stable user characteristics such as expertise or interest. Short-term models are employed in transitory problem-solving behaviour for the specific task in hand, focusing on particular topics and goals in the immediate interaction. This distinction is also referred to as local (short-term) user models in contrast to global (long-term) user models.

In the intelligent interface process control domain, the user model has to deal with fundamental cognitive characteristics such as users preferences for particular styles of display, and basic cognitive capabilities such as spatial ability and preferred learning styles (Van der Veer, 1990; Benyon, 1993b). In these systems long-term, cognitively valid models are vital.

(Benyon 1993b) describes three types of knowledge the User Model should contain to correctly understand the users goals.

> A *task level* that describes user goals in the domain. Using this, the model should understand from interactions, what the user is trying to achieve within the domain rather than how they are achieving it.
>
> A *logical level* describes what the model believes the user understands about the logical functioning and the logical concepts embodied in the domain. So, the system should be able to understand, for example, that attempting to execute a database language 'Select'

statement in response to a help system prompt is a logical, or semantic error (given that the help system cannot execute database language statements) rather than a syntactic error of attempting to obtain help on the 'Select' statement.

A *physical level* that describes what the model believes the user understands about the physical aspects of the system. Thus the system should be able to distinguish between the operator making a semantic error, and syntactic error.

As mentioned above it is vital that a user model also contains fundamental data concerned with essential cognitive traits of users. There is an increasing body of experimental evidence which confirms that users differ in cognitive skills and personality traits and that this significantly affects the quality and nature of certain interaction styles and user requirements (van der Veer, 1990; Egan 1988; Jennings & Benyon, 1992). Such user characteristics are particularly resistant to change by the user and hence are particularly important for adaptive systems. If users find it difficult or impossible to change certain aspects of their make-up, these are exactly the characteristics to which the system should adapt (van der Veer; 1990, Benyon 1993b). One example is spatial ability. This is a characteristic that appears relevant in HCI (Vicente & Williges 1988; Vicente, Hayes and Williges, 1987; Egan, 1988), particularly where users have to navigate through a conceptual space of file structures or system modes.

The Domain Model

The user model is required in an adaptive system so that it can alter aspects of the system in response to certain inferred or given user characteristics. The domain model is required in order to define the aspects of the application that can be adapted or which defines the context in which adaptation should take place. Other similar terms that have been used to describe this concept include *application model, system model, device model* and *task model* and in our domain, *process model*. The domain model serves a number of purposes. It forms, with the User Model, the basis for all the inferences and predictions that can be made from the user-system interaction. It is important therefore that the model is defined at an appropriate level of abstraction to allow the required inferences to be made.

The Domain Model consists of one or more abstractions of the system. These abstractions allow the adaptive system to reason about the target application, to facilitate adaptations by other agents, and to evaluate its effectiveness. For example, if the system is to be capable of adapting the screen displays then there must a sub-model describing the screen state in the domain model. If it is to adapt the functionality of the system, then the domain model must contain representations of alternative functional capabilities and the relationship between these functions. Similarly, if the system is required to alter the description of concepts then these too must be modelled.

The benefits to be gained from having an explicit and well-defined domain model are considerable and have long been recognized in Artificial Intelligence research. A separate domain model provides improved domain independence that allows easy refinement of the domain model. This is important, as it is unlikely that any adaptive system design will have a perfect representation of the domain at the first attempt. A separate and explicit domain model can also be more easily used for other purposes, such as providing explanations of the system's behaviour.

Thus the domain model is a description of the application, containing facts about the domain, i.e. the objects, their attributes, the relationships between objects and the processed involved. The domain model is the designer's definition of all aspects of the application relevant to the needs of the adaptive system. A central question in constructing a domain model is deciding what level of description should be represented.

To achieve the right level it is important to understand the behaviour of complex systems from various viewpoints such as a physical view, a design view and an intentional view. The physical view, (also called the physical stance or physical strategy) argues that in order to predict behaviour of a system, the physical constitution and the physical nature of any inputs are determined and then predictions are made about the outcome based on the laws of physics. However, sometimes it is more effective to switch to a design stance. In this strategy, predictions are made about how the system will behave by believing that it will behave as it was designed to behave. However, only designed behaviour is predictable from the design stance. If a different sort of predictive power is required then the intentional stance can be adopted. This can be summarised as follows:

> Treat the system as a rational agent
>
> Figure out what beliefs *it ought to have* given its place in the world and its purpose
>
> Figure out what desires it *ought to have*
>
> Predict that this rational agent will act to further its goals in the light of its beliefs and hence
>
> Predict what the agent will do on the basis of what it *ought to* do.

The Interaction Model

The third component of an adaptive system is a representation of the actual and designed interactions between user and application - the interaction model. This use of the term is very different from the interaction model proposed by Norman (1986), which is a theoretical representation of human-computer interaction in general. The Interaction Model is much closer to the notion of a discourse model, or dialogue model

An interaction is an exchange between a user and the system at a level that can be monitored. Data gathered from monitoring this interaction can be used to make inferences about the user's beliefs, plans and/or goals, long-term characteristics such as cognitive traits, or profile data such as previous experience. The system may tailor its behaviour to the needs of a particular interaction or, given suitably 'reflective' mechanisms, the system may evaluate its inferences and adaptations and adjust aspects of its own organization or behaviour.

In some representations (Alty & McKell, 1986) the interaction model is seen as a part of the domain model. However, it can also be modelled as an entity in itself or be seen as a function of the operator model

There are two main aspects to the interaction model:

Capturing and analysing appropriate raw interaction data

Representing the inferences, adaptations and evaluations which may occur

Raw data is processed to create a dialogue history or dialogue record, which is a trace of defined aspects of the user's observed behaviour. This dialogue record is kept for as long as is required for adaptation decisions. There are interesting issues as to when elements of it should be deleted. It is likely to contain details such as the sequences of keystrokes, mouse clicks and mouse movements made, timing information and system messages. It is an abstraction of the interaction since it cannot capture everything that takes place on the interface.

The second part of the interaction model is a description of 'stereotyped' interactions; the Interaction Knowledge Base (IKB). This describes the inferences that can be made from the dialogue, the evaluations of the interaction which are possible and the changes (or adaptations) that the system can make.

The User Model and Domain Model define on *what basis an inference can take place*. The IKB actually carries out the inferencing process by combining the various Domain Model concepts to infer user goals, strategies and characteristics or by combining user model concepts to adapt the system. The IKB represents the relationship between domain and user characteristics, and interprets the dialogue record.

The interaction model is a key component of an adaptive system, but it is difficult to design. The developer of adaptive systems must decide the levels of abstraction that are required for the dialogue record, the individual user data and the interaction knowledge-base.

Examples of the Adaptive System Concept

One of the difficulties of discussing adaptive systems is that similar ideas have emerged from different disciplines, employing their own terminology, which make comparisons and generalisation difficult. Systems that are described as 'intelligent' may take many forms, and are built for many different reasons and to achieve many different goals (Elkerton, 1987; Mason & Edwards, 1988). Claims of success, however, have been exaggerated and implemented systems have usually only been successful in well-defined, more manageable areas or where they deal with limited issues that are more tractable. Benyon and Murray (1993) have provided a detailed review of adaptive user interfaces, from which three illustrative examples have been chosen.

Intelligent Support Systems

A popular application of intelligent interfaces is in the provision of context-dependent 'active' help (Fischer, Lemke and Schwab, 1986; Hansen, Holgaard and Smith, 1988;). On-line help systems track the user's context and attempt to aid the user in times of difficulty. Such systems are called "Intelligent Support Systems" (ISS) and often attempt the very difficult task of deducing a user's higher levels goals from a series of low-level interface interactions. Various strategies and approaches have been suggested. (Fischer et al., 1986; Fischer, Chin, 1986; Jerrams-Smith, 1985). Intelligent help has further developed into 'critiquing systems' (Fischer, 1989), where users are competent in the subject domain being critiqued, rather than being tutees or learners (Moore and Swartout, 1988; Fischer, 1987; Fischer, 1989).

A major problem with these mixed-initiative and co-operative dialogue approaches is the position of 'locus of control' at any one time. There is a difficulty in deciding who has control at any point, the operator or the interface? If a computer assistant or critic offers a piece of advice that an individual operator or user overrides, how can that assistant adjust to this situation and operate in a useful way for the duration of the same task. The problem is made worse if the same task is repeated. Can the system learn from the user's past behaviour and so avoid giving the same bad advice over and over again? The system must have knowledge of how to be a 'competent assistant' through understanding the limits of its own competence.

Explanation Systems

A variant of help systems attempts to provide an explanatory facility of the behaviour of the system to the user (Paris, 1989). This was a goal of early expert systems but they were criticised for failing to provide adequate and suitably tailored explanations. It was realised that to be effective, these systems had to tailor their explanations to the assumed knowledge of the user (Carroll and McKendree, 1987). However, these explanation-based systems have presented extremely stubborn problems to researchers and to system builders because they combine the problems of natural language generation, help and tutoring in one system.

Co-operative Intelligent Agents

Recent interest in computer supported co-operative work (CSCW), distributed artificial intelligence (DAI) and HCI have taken the adaptive system concept further and this has led to the development of interface agents. Co-operative systems require models of all the systems and humans participating in the interaction (Seel, 1990). Agents are entities capable of voluntary, rational action carried out in order to achieve goals through holding a representation or 'belief' in the state of the world. These beliefs are generated through the agents existing knowledge, and through observation of its environment and interaction with the user.

Interaction schema will depend upon the number of interacting agents. In more complex systems, agents may be interacting in a number of ways, modelling other agents and adapting and responding to a variety of needs. Initially, agent systems promised much but delivered very little. Over the last few years, however, there has been a movement towards creating multi-agent systems in the "weak AI" mode, whereby the intelligence of the system arises through interaction and negotiation between agents rather than from the inherent reasoning abilities of individual agents. Multiple agent systems have thus become increasingly more realistic in their goals since their inception. Such developments can be seen a natural extension of dialogue assistants (Alty and McKell, 1986; Alty and Mullin, 1987).

Essentially, agents are adaptive systems that are specialised and know about only a very small part of the world. An important issue for co-operative or support systems is, firstly, how knowledge is actually represented and, secondly, how the interaction or conversation can be realised. Co-operative support systems can also be thought of as task-orientated dialogue systems, which are actually characterised by the 'conversational roles' that each partner is expected to adopt. Active dialogue partners in mixed-initiative dialogues are those that try to identify a user's intentions in order to exhibit co-operative behaviour. To behave co-operatively, the system must discover the plans underlying the user's questions or statements; represent those plans in its knowledge base; examine them for hidden obstacles and provide information that overcomes those obstacles.

Adaptive System Commonalities

Although the three system categories described above different in many respects and originate from different disciplines, they share many similar characteristics. All are adaptive systems in that they automatically alter aspects of the system to suit the requirements of individual or groups of user - or more generally to suit the needs of other actors in the system. All have to infer characteristics of the other actors from system interactions.

Examples of Web Based Adaptive User Interfaces

There are now many examples of adaptive systems in existence which range from rather simple implementations, to multi-agent based distributed adaptive systems. In this section we will examine a broad spectrum of web based adaptive applications will be examined to get a feel for how such systems work.

One very common use of a simple adaptive system is the domain of *information filtering*, in which the aim is to select for the user, material that they will find informative or useful. Systems of this type have a long history in the field of information retrieval, but have grown in popularity since the development of the World Wide Web. There are now a number of information filtering systems that incorporate user feedback and attempt to adapt to user preferences.

One example interface is Pazzani and Billsus' (1997) "Syskill & Webert", which recommends Web pages on a given topic that the user is likely to find interesting. Starting from a handcrafted page for the topic, the user marks suggested pages as desirable or undesirable, and the system uses this feedback as training data to develop a model of user preferences. Syskill & Webert represents each user profile as a naive Bayesian classifier, which stores a conditional probability distribution over a set of predictive features, in this case words that occur in the Web page. The system invokes this user profile and compares it with the words in a candidate document when deciding whether to recommend that document to the user. The approach biases its response towards documents that are similar to ones the user has previously ranked highly. This system has a simple adaptive mechanism, but seems to work rather well. It certainly has higher quality results than earlier systems such as *Firefly*.

Although recommending Web pages is a common application of adaptive information filtering, other applications exist. The NewsWeeder (Lang, 1995) system recommends news stories to readers, again using the words in each story to predict whether the user will find the article interesting. Another popular task involves sorting and prioritising of electronic mail, typically using words that occur in the message headers and body (e.g., Boone, 1998). This system is a big improvement on earlier systems such as *NewsWire*. The quality of the approach ensures the results are usually good. Another common technique used for electronic mail delivery recommends items that the user might enjoy based on the user's ratings and the ratings from other users with similar profiles. Amazon.com uses such collaborative filtering mechanism to recommend books to its customers.

Another adaptive user interface (Rogers & Langley 1998) provides advice to car drivers. Their Adaptive Route Advisor accepts a current and desired location from the user, carries out a best-first search through a digital map and selects a few high-quality routes that are then presented to the user. When the user accepts one of the suggested routes, the system incorporates this decision into its training set and revises its user model, which it represents as a set of relative weights for global route features such as the number of turns, the distance, the number of intersections, and the estimated driving time. The algorithm that updates this user model carries out a hill-climbing search through the weights space, endeavouring to characterise parameters that summarize past choices the user has made. The system then draws on the revised model in directing the search for routes on future tasks. This system is a prototype, but according to the authors has returned some impressive results.

Yet another adaptive interface, Inca (Iba, Gervasio, and Langley 1998), focuses on scheduling in the domain of chemical spills and fires. The Inca system retrieves a schedule from a case library that best matches the features of the current incident, and then lets the user interactively adapt them for application to their situation, for which the system suggests likely repairs. Once the user decides on an acceptable schedule, Inca passes this solution to an execution module, which may lead to new events and the need for further repairs to the schedule. Personalisation occurs through storage in the case library of the final agreed schedules, which presumably reflect user preferences about desirable solutions, and through the induction of rules about the conditions under which the user makes each type of repair. Inca uses the expanded case library on future problems and incorporates a revised repair model to recommend future revisions.

Examples of More Complex Adaptive Systems

MMI 2 System

The MMI 2 system (Chappel & Wilson 1993) supports a system user dialogue in which users aim is to design a computer network for a building. The system acts as an expert in network design, and provides a number of interface options, which allow Natural Language interaction through

English, French and Spanish. It also provides a variety of interaction modes such as command language, non-verbal audio and design gestures (such as editing symbols). The user can also manipulate graphical displays of CAD style building and network diagrams and business graphics style charts

The knowledge-based design component uses heuristics to make decisions about the selection and design of graphical responses. The heuristics depend on knowledge about graphic design, the capabilities of the graphical tool selected and the information in the system's reply. However, they also rely on knowledge about the application, the users, the tasks that the users perform and the dialogue between system and user.

The MMI 2 system does not appear to have any support for spatial adaptation, and seems limited to selection of various charts from among an array. It does however demonstrate the general principles of an adaptive system in that it relies on a user/domain model.

The DIGBE System

Dynamic Interaction Generator for Building Environments (DIGBE) (Penner 1998) automatically designs and presents a user interface that is also dynamically adaptive. DIGBE is used in the domain of building management for handling tasks such as configuration, monitoring and control of security and environmental systems, management of users and data analysis. DIGBE's adaptation mechanism attempts to identify the operator and use profile information to configure a specific interface for that particular operator. To do this the system uses a combination of mechanisms including role-based task composition and object specialisation. It responds dynamically by constructing and maintaining real-time models of the system of interest and the user-system interaction. The DIGBE system provides a degree of domain independence by separating the interaction and presentation reasoning models. The system is agent-based, allowing multiple presentation agents to use a single interaction design.

DIGBE works by specialising the interface according to the nature of the user, thus a heating and ventilation technician would only see heating and ventilation information, a security guard would have access to very different information pertaining to their role and task. The DIGBE architecture adheres closely to the general structure of an adaptive system, in that, at its core, are a domain, task/interaction and presentation model

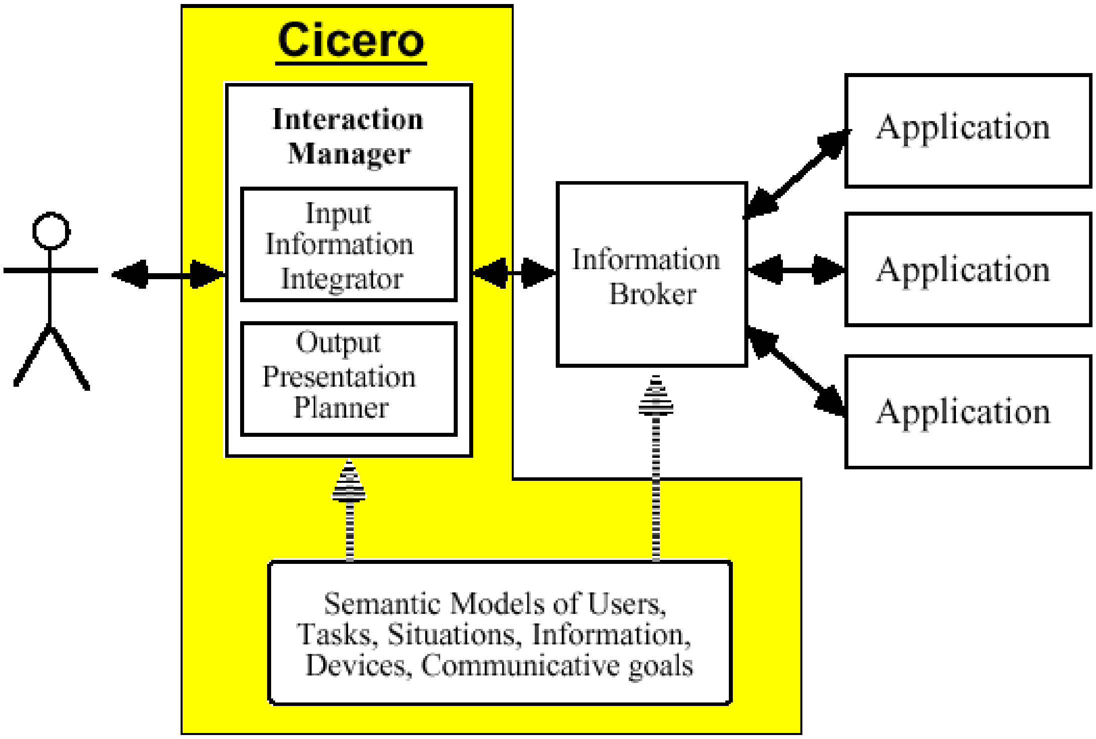
The DIGBE system uses agents as actors in the roles of domain reasoning, interaction and presentation reasoning. The system intelligence is centred round the knowledge structures contained within the agents and the interactions between them. Agents within DIGBE are actually little more than very simple model managers when separated from their knowledge structures.

DIGBE's user adaptation is strictly limited to determining the type of interface required based on the type of operator. It therefore specializes the interface to match the operator. It does not dynamically adapt to the users needs on the fly. It also adapts itself to the state of the objects within the interface. Thus, when a heater value is changed, this change is represented within DIGBE as a change within a continuous data object. DIGBE then attempts to select an appropriate representation that best represents a continuous data object, and sends the "create yourself" message to its object instantiator.

The Cicero system

The Cicero system (Arens & Hovy 1995) is an adaptive presentation manager that liases between the domain and the interface to present the best possible configuration. It aims to be a generic system or an interface capability platform onto which different applications and different media can be grafted without altering the basic operation of the system. To achieve this general-purpose ability, the Cicero performs its job using a collection of declarative models that embody all knowledge needed to manage interface communication. In particular, it has access to both generic and specific knowledge of the characteristics of information to be displayed or input, the characteristics of available media, the communicative context, the presenter's communicative goals (whether human or machine), and the perceiver's goals, interests, abilities and preferences.

The general architecture can be summarized as follows in Figure 2



Cicero
Interaction Manager
Input Information Integrator
Output Presentation Planner
Information Broker
Application
Application
Application
Semantic Models of Users, Tasks, Situations, Information, Devices, Communicative goals

Figure 2 The Cicero Adaptive System

On the one hand Cicero has the data that needs to be presented (or a description of the type of data that must be input by the user), and on the other a collection of media that may be used, possibly in some combination, for this purpose. Cicero makes a match dynamically at run time, using the properties of the data, the communicative goals involved, and the present interaction in the context of the ongoing dialogue. It then proceeds to select media with features that satisfy the display desiderata and to create the content of the display itself.

The key components in this architecture are the semantic models, where characteristics are matched to corresponding models. For example, a typical presentation planner plan stipulates that when information carries the value *high* for the feature *urgency,* it should be presented on a medium whose model contains the characteristic *high* for the feature *noticeability* (such as a speech synthesizer or a flashing icon). Once the specific characteristics of a new medium have been defined in terms of the appropriate generic model, all the system's other modules will immediately be able to make use of it.

The system utilises five distinct models, all of which can be defined under a single high-level set of semantic terms:

Media characteristics and capabilities.

Information characteristics.

Application tasks and interlocutor goals.

Communicative goals and discourse structure.

User's capabilities and preferences.

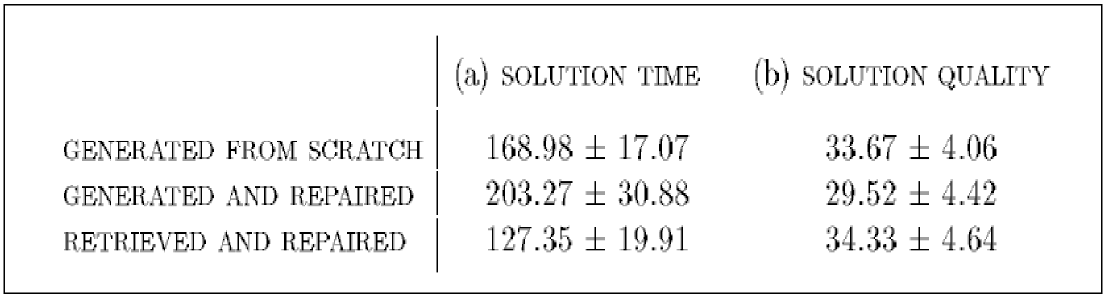
These models are then used to dictate which representation is best used under a certain set of conditions. This system proposed to use domain models to match criteria, however this system was never actually built and tested. Additionally, it is limited to displaying only certain type of representations and does not include a spatial adaptation planner. Lastly, although it is deemed to be generic it is only designed for reasonably simple systems and not for complex systems such as the process control area, where solutions are often required in a time-critical fashion. This system, if implemented, would almost certainly take too much time in reasoning on the nature of the representation.

There now follows an examination and evaluation of adaptive systems in general, and methods for assessing them.

The Examination And Evaluation Of Adaptive Systems and Methods For Assessing Them.

Measures of Efficiency

Users typically employ computational decision aids, including adaptive user interfaces, because they expect the software will let them accomplish some task more rapidly, and with less effort, than they could accomplish on their own. This makes the efficiency of the decision-making or problem solving process an obvious dependent variable to use when evaluating such adaptive interfaces. However, a metric such as efficiency is complex and reflects only part of the picture. One obvious candidate for measuring efficiency is the time users takes to complete their interaction with the adaptive system to achieve a goal. Another is the quality of the solution provided which will be discussed later.



	(a) SOLUTION TIME	(b) SOLUTION QUALITY
GENERATED FROM SCRATCH	168.98 ± 17.07	33.67 ± 4.06
GENERATED AND REPAIRED	203.27 ± 30.88	29.52 ± 4.42
RETRIEVED AND REPAIRED	127.35 ± 19.91	34.33 ± 4.64

Table 1 Inca Result

For example, Table 1 shows results from an experimental study with Inca, the interactive scheduler described earlier. One version of the system presented the user with an empty schedule, another used heuristic search to generate the initial schedule which the user then repaired, and a third version retrieved a schedule from its case library for revision by the user. The dependent measure was the number of seconds taken to transform this initial schedule into one the user found acceptable.

Another possible measure of efficiency is the effort that the user must exert to make a decision or solve a problem. Here, a plausible metric could be the number of user actions that occur during solution of a given problem. In evaluating their system for aiding the completion of repetitive forms, (Hermens & Schlimmer, 1996) measured the number of keystrokes that the user took to complete the form, which they found generally decreased over time as the user progressively interacted with the system. Keystrokes were the obvious performance measure for this type of interface, but different metrics such as mouse clicks would be more appropriate for an adaptive graphics package, or utterances might be more appropriate for systems that incorporate a speech interface.

Measures of Quality

Another important reason why users employ adaptive systems is to improve the quality of solutions of their tasks. This quality goal is especially common in problem-solving activities that involve many steps, but it is also relevant for systems that have the goal of selecting an appropriate item from many choices, like a book or a Web page. As with efficiency, the notion of quality can be defined in many different ways.

The measurement of quality can be simplified if there exists some objective measure of quality in the domain. This can then be used directly as the dependent variable in an experimental study. For example, some popular advisory adaptive systems search the World Wide Web to find the site that offers a given item (a particular book or software package) at the lowest price. For such tasks, the selected price constitutes an objective measure of the decision aid's success, which can then be compared with that achieved by another advisory system or with the user's performance without computational support.

Evaluating quality is complex in domains that involve more than one criterion for success. For example, the Inca system, whose results were provided in Table 1, operates in a domain where the user wants to minimize chemical spills, chemical fires, and hazard to human life. To evaluate the quality of system solutions in Inca, the developers (Iba, Gervasio, and Langley, 1998) constructed a simulator that could execute the generated schedules and then measured their percentage improvement on these dimensions over the alternative of taking no action. However, to obtain a single quality metric, they had to combine these separate factors in some manner, and for simplicity they chose to give them equal weights. Table 1 column (b) shows the resulting quality measures using this technique which suggested that Inca's seeding schedules using retrieved cases did not significantly improve the final quality measure compared with either solutions produced from scratch or solutions produced by seeding the repair process with schedules generated by heuristic search.

However, giving equal weight to different quality criteria conflicts directly with a core assumption of adaptive interfaces: that users differ in the relative importance they assign to such criteria. One obvious source for such information is the "learned" user model, but using this would be circular in that it would guarantee improvement in quality. In cases of multiple criteria, we need some external measure that is subjective but that is not tied directly to the user model, which may only partly reflect the user's true preferences.

Measures of User Satisfaction

The above observations suggest that some separate measure of user satisfaction to determine quality of the system's behaviour, is needed. One way to collect this information would be to present each user with a questionnaire asking them about their subjective experience. Although embedding a questionnaire in the system itself makes extra demands on the user (which seems undesirable) this does not prevent a researcher from presenting a form to experimental subjects after they have finished using the adaptive interface However care must be taken since questionnaires can be unreliable in predicting whether a person will continue to use the system or not

Another measure of user satisfaction involves giving the user some control over whether they are allowed to use certain system features or not. If a user requests the system's adaptive capability or alternatively disables its after some initial interactions, the system may be able to conclude that in the first case the user seems to appreciate the advice given, or in the latter case the user has not been satisfied by the experience with these features.

Measures of Predictive Accuracy

Because the user model in an adaptive interface makes predictions about user responses to system advice, there is a natural temptation to rely on predictive accuracy as a surrogate measure for efficiency and quality.

However, there are some inherent problems with using predictive accuracy to determine the success of an adaptive interface. Although this measure can be a useful analytical tool for understanding the details of system behaviour, it does not directly reflect the overall efficiency or quality of the solutions obtained, which should be the main concern. Correct prediction of user responses may be correlated with these direct measures, but it cannot substitute for them. Also, some studies (including (Gervasio et al., 1998)) have involved collecting user traces in a non-adaptive setting, and then using learning to create a user model from the data, measuring the model's accuracy on the remainder. This scheme violates the standard assumption that adaptive interfaces change their user model over time, making the results of marginal relevance.

Independent Variables

A scientific experiment must do more than measure behaviour under some condition. Because it aims to understand the factors that influence that behaviour, it must measure the dependent variable in two or more situations that differ on some dimension. Because these factors can vary independently, they are often referred to as independent variables. As with dependent measures, different controllable factors make sense for different disciplines. Here we consider four classes of independent variables (Effects of experience, quality of decision, personalisation, task characteristics) that are appropriate in the study of adaptive interfaces.

Effects of Experience

We have seen that adaptive interfaces develop user models by observing user behaviour. This feature distinguishes them from traditional advisory systems, which remain static in their response over time or which the user must reconfigure explicitly. However, their reliance on this approach makes it important that adaptive interfaces learn rapidly, since most users will want to see their feedback have an effect soon after they have provided it. The issue here is not CPU time but rather the number of training cases needed before the system can accurately predict user preferences. Other things being equal, users will prefer adaptive interfaces that learn rapidly over ones that learn slowly. As (Langley 1997) has noted, this concern with rapid learning encourages the use of simple induction algorithms, since they usually achieve reasonable accuracy in much shorter times than more sophisticated methods that have many more parameters.

This concern with learning rates also has implications for the evaluation of adaptive user interfaces. In particular, it suggests the number of training cases that the system has collected from the user as a natural independent variable. Plotting some performance measure against the number of training items produces a learning curve. Such graphs are common in the psychological literature but remain rare in machine learning, where most researchers report results on training set of pre-selected size. In general, one hopes that the learning curve for an adaptive interface will increase quickly in the early stages, even if the curve levels off as the training set increases.

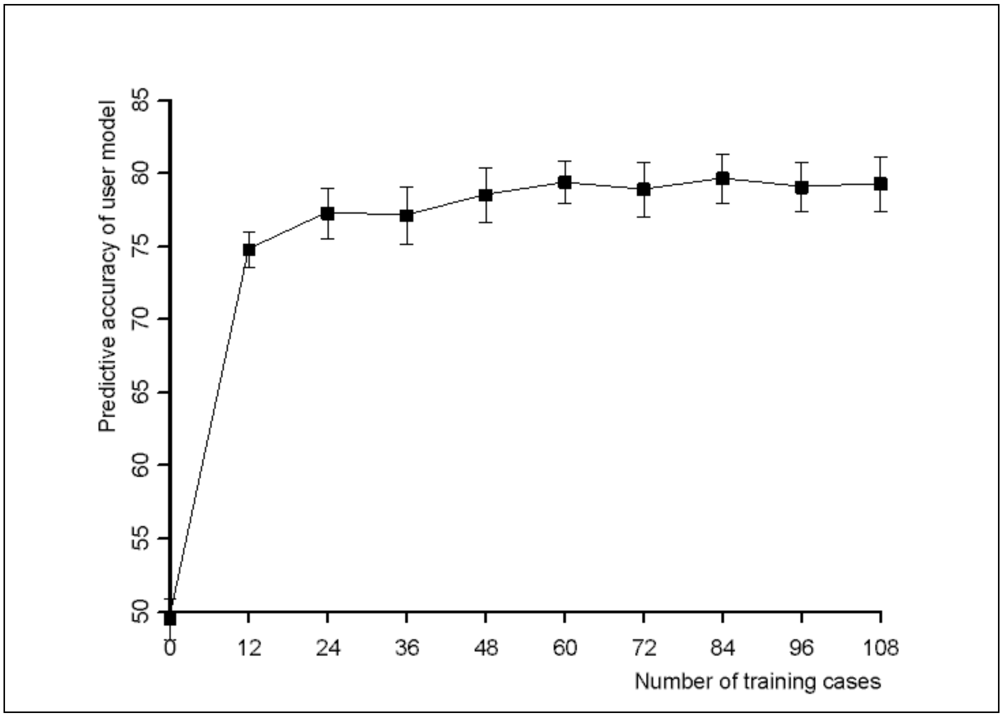


Figure 3 A learning curve showing the percentage accuracy of a personalized user model

Figure 3 shows a learning curve from Rogers and Langley's studies of their Adaptive Route Advisor. Here the dependent variable is the percentage of route pairs for which the learned user model correctly predicts the route the subject prefers, averaged over 24 users and over ten training test splits for each user. As expected, the accuracy increases quickly from random to 75% after 12 training pairs, and then grows more slowly until it levels off at 79% at around 60 training pairs. Although more complex induction methods might have higher asymptotic accuracy after many more interactions, the Route Advisor's simple perception scheme serves it quite well in achieving a reasonable accuracy quickly.

(Pazzani & Billsus 1997), (Hermens &Schlimmer 1994), and (Gervasio et al. 1998) also report learning curves for their adaptive user interfaces, which suggests that they all recognize the importance of steep early learning rates for their systems' success. However, most studies still collect user decisions in a non-adaptive setting, and only then use these traces to train and test the user modelling method off-line. As noted earlier, the results obtained in such experiments can differ from those observed in actual system use, since adaptation can lead to different recommended options as the user model is updated and since the users may react to these changes in system behaviour.

Quality of Decision – Making Assessments with Non-Adaptive Approaches

Another key claim of adaptive user interfaces, and computational decision aids in general, is that they help their users make decisions more effectively. Testing this claim requires independent measures of effectiveness like those considered in the previous section, but it also requires a comparison between user behaviour with and without the advisory system. Variations of this sort constitute an important independent factor in the experimental study of adaptive interfaces. A clear advantage of adaptive user interfaces is that their interactive nature makes it easy to collect data on user behaviour. But this also means that it is typically difficult to measure user performance in the absence of the interface. As a result, most experimental studies compare the full version of a system with a version that lacks certain features but that retains its interactive (often graphical) nature. Such studies indicate whether the omitted component actually aids user performance, but not whether users fare better with the limited interface than with no computational aids at all. Most researchers simply assume the latter holds, although it could be tested empirically as well, with some difficulty.

(Pazzani & Billsus 97) report one limited interface study with their Syskill & Webert recommendation system. They compared one version of their system, which based its user models on a combination of words that the user suggested and a set selected by cross validation, with a more limited version that used only the former and a third that used only the latter. Figure 4 reproduces their learning curves from one domain, involving Web pages about biomedical topics, in which both of the more limited systems did substantially worse than the full version of Syskill & Webert. Similar results occurred for two other domains, which led Pazzani and Billsus to conclude that both sets of features provided important sources of power for their recommendation system.

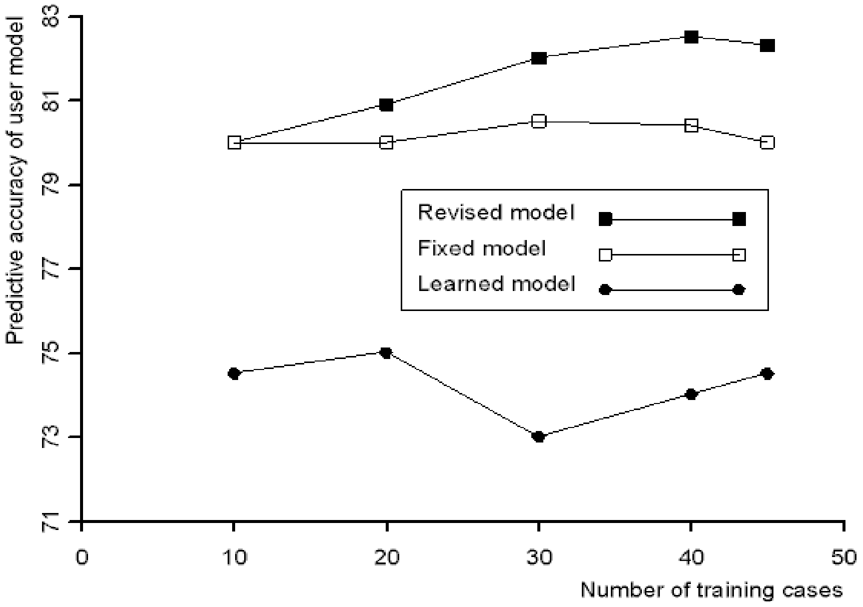


Figure 4 Comparing models

In some situations, it makes more sense to replace one component of the interface with another component than to remove it entirely. Such a replacement study contrasts system behaviour using the standard module to behaviour with another module that, intuitively, should not produce as good results. This straw man may use less information, use less computation, be less adaptive, or be otherwise more limited than its analogue in the basic advisory system. The conclusions one draws from such experiments are the same as in limited function studies; if the straw man leads to worsened performance, then the standard module contributes to the success of the original system.

The form completion system developed by Hermens and Schlimmer (1994) lends itself naturally to such a replacement study. Their experimental evaluation examined three conditions, one for the system's standard adaptation method which relies on decision-tree induction, and two others for simpler induction methods: predicting the most recent value for a given held and predicting the most common value. These simpler techniques played the role of straw men, in that one would expect a system which relies on them to fare worse than one which relies on the decision-tree method, at least if the more sophisticated method is truly useful. The results of their study supported this conclusion, since the two straw men reduced keystrokes much less than the decision-tree module.

Personalisation and User Effects

Another type of independent variable that arises in the evaluation of adaptive user interfaces concerns the nature of the person using the system. The importance of user characteristics has long been recognized in human-computer interaction, where different types of interface may be appropriate for different types of users.

Similarly, the notion of aptitude-treatment interaction has made its way from educational psychology into some computer-based tutors, which present material in different ways depending on student learning styles. Although such issues are relevant for adaptive user interfaces, they are less central than the claim that such systems benefit from adapting to individual users.

One can best test such hypotheses about personalization by testing the system on different users than it was trained on. For example, (Iba et al. 1998) report a personalisation study with their interactive scheduler, Inca, which involved two separate users. In one condition, they used schedules from a given user's case library as the starting point for that user's repairs; this corresponds to the system's default mode. In the second condition, they presented each user with starting schedules from another user's case library. They predicted that subjects in the first setting would complete their revisions in less time, and produce schedules with higher quality, than those in the second situation, since schedules constructed by a particular users should reflect their preferences better than those created by another.

However, their experiment revealed no significant differences between the two conditions on either dependent measure, suggesting that personalization at this stage of the system is less important than expected. (Rogers & Langley 1998) present a different approach to testing personalization claims in the context of their Adaptive Route Advisor. Their first experimental condition was analogous to that in the previous study, in that it tested a learned user model's ability to predict route preferences for the user on which it was trained. But in their second condition, instead of using a model trained on a particular user, they used a generalized model trained on decisions from 24 different subjects. Their hypothesis was that the personalized model would more accurately predict user responses than the generalized model, even though the latter had been trained on 24 times as much data. Figure 5 shows the results of this study, with accuracy shown separately for each subject. In this case, the personalized models clearly fared better than the generic one.

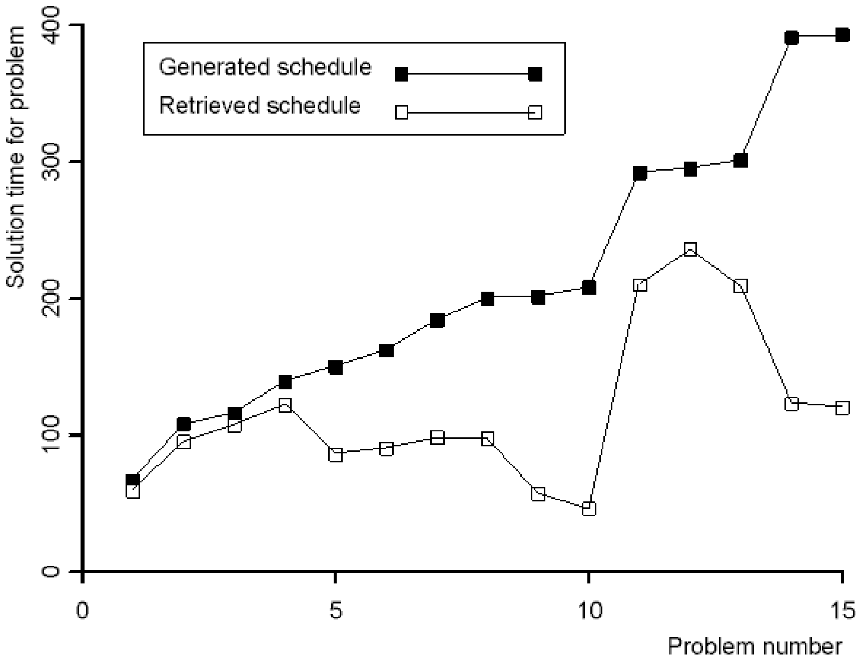


Figure 5 The time taken to repair a schedule as a function of task difficulty

A fourth important class of independent variables concerns the characteristics of the task that the decision aid aims to support. In general, adaptive interfaces are intended to help users handle difficult tasks effectively, so most task variables involve some measure of problem difficulty. For instance, one can make a selection task more challenging by increasing the number of items available or by increasing the number of features that describe each item. Similarly, one can make a configuration task harder by increasing the number of slots to be filled, the number of components possible for each slot, and the constraints that must be checked amongst them. The general prediction is that, as task difficulty increases, performance on the task will decrease.

However, designers of adaptive interfaces are less interested in the task effects themselves but more in the ability of their computational aids to minimize these effects. When present, this ability should appear as an interaction between task variables and system variables. So, an increase in task difficulty is expected to result in a lower reduction in performance when using an adaptive interface than when operating without such assistance. Users will still take longer to make decisions and generate solutions with lower quality when they encounter more difficult problems, but we would expect the rate of reduction to be less than if the task was attempted without an advisory system.

An illustrative example of a task-oriented experiment comes from the Inca study (Iba et al 1998) that compared the times taken to repair an initial schedule, retrieved from a case library, with another schedule generated by heuristic search. In a follow-up analysis, the researchers decided to order the scheduling tasks by their solution time under the second condition, which constitutes a rough measure of problem difficulty. Figure 5 presents the two resulting curves, which show that the time to repair generated schedules increases with problem difficulty, but that this trend is much weaker when users repair a schedule that Inca retrieves from its case library. Note that the definition of task difficulty here is somewhat circular, as it is linked to the dependent measure rather than being defined independently. Still, the experiment illustrates the interaction between task complexity and components of the advisory system.

Conclusion

As interfaces become more complex, and require greater degrees of interaction, it becomes increasingly important that the interface provides greater support for the user. Adaptive systems provide just such a mechanism, they attempt to glean a understanding of the context in which the user is operating the interface, and utilise this knowledge to configure the interface in such a way as to make the user's goals more attainable. They aim to make the operation of the interface a more efficient and easier task whilst maximising the users satisfaction. Adaptive systems are of most use within complex domains where, typically, the user has to deal with large amounts of information in short amounts of time.

For an adaptive system to operate satisfactorily it must have adequate knowledge about the current context of the users operation. To do this, a general model is proposed of an adaptive system, which incorporates a user model, a domain model and an interaction model. The user model captures contain knowledge of how the user interacts with the system, user trends and expected behaviour. The domain model captures knowledge about the domain the user is operating in, and the interaction model captures how the user interacts with the system.

There exist many types of adaptive system, from primitive web-based systems to more complex adaptive presentation systems such as DIGBE, Cicero and MMI2. The more complex systems adhere to the general model of an adaptive system, and exist in several different domains. Agent based systems lend themselves very well to an adaptive system, since independent agents can represent the actors (domain, user and interaction) that should exist in an adaptive system (Chapter 4). These multi-agent systems tend to rely more on the weak notion of AI to deliberate about required run-time adaptations. (Chapter 5)

Several means exist of assessing an adaptive system including: efficiency, quality, and user satisfaction.

This chapter has therefore shown that adaptive systems are an interesting area of investigation, particularly in complex domains where their usefulness can be maximised, and that multi-agent systems are an attractive means of implementing such systems.

The next chapter investigates one such complex domain, the domain of process control, and examines why an adaptive system would be useful. It also describes issues that exist within process control that are important for building an effective adaptive system.

Chapter 3

PROCESS CONTROL INTERFACES

Introduction

In this chapter an examination is made of the domain of process control, and the problems this complex domain can present to the operators trying to control it. In the process control domain there is a surfeit of information that can be presented in many different representations. These different representations support different problem solving techniques, and the appropriateness of a particular technique will depend upon the problem, the context and the operator. Clearly, an adaptive approach would be beneficial since the system could choose the most appropriate technique relevant for a particular context. Recent developments in multi-media interfaces have relevance here since they can provide a set of alternative representations that would form the base material upon which the adaptive system would act. The problems frequently encountered by operators are first outlined, and then a number of proposed design methodologies are examined. Approaches using multiple media, extracted from the literature, are proposed as a possible means of alleviating these problems.

Process Control Work Domain

Designers have continually strived to improve the design of process control interfaces in the belief that:

Economically substantial savings can be made

Safety margins can be improved

Process control interfaces are becoming more complex because the introduction of automation has put a physical and cognitive layer between the process and its human controllers. Operators now tend to control what (Wickens 1984) termed the "outer-loop variables" as opposed to the "inner loop variables" which are handled by the supervisory and control system automatically. Outer Loop variables need higher level cognitive frameworks and it is hoped that adaptive presentation systems which fully exploit multiple media, will provide the operators with a clearer idea of what is happening in the process, particularly during disturbances.

The Process

There are several problems that are unique to process control systems, which cause operator difficulty. Firstly there are inherent delays between operator actions and observable effects. Crossman's Waterbath (Crossman & Cooke, 1974*)* is a simple process control system, which illustrates the difficulty of controlling a simulated process when the response is delayed.

Secondly, processes are often very complex and involve the control of several concurrent sub-processes. The sheer size of many industrial processes means that the number of parameters that can, and must be controlled, is very large. Similarly the amount of sensor data required to support these control actions is also high, regardless of whether control is carried out by the operator or by automatic feedback loops. For example, in a gas cooled nuclear power plant such as Scottish Nuclear Power's Hunterston B plant in Ayrshire, Scotland (Shahidi et. al., 1990), the coolant gas flow, power supply, reactors, steam turbines and power generation all constitute complex sub-systems of the overall process being controlled. Each of these sub-systems must function independently and in parallel. Consequently, there are multiple concurrent tasks, which require the operator's attention.

All these problems add to the complexity of the operator's task in a process control environment. It is the hypothesis of this thesis that an adaptive presentation system will aid the operators in controlling and manipulating the process more efficiently

Process Automation

It is the complexity and critical responsiveness of the process to be controlled that has motivated the partial automation of many industrial processes. The technology that enabled this to happen, has improved the safety, as well as the efficiency, of modern industry (Sanderson 1989). However, the operator has not really benefited from this. Bainbridge (1987) articulates that many operator problems stem from the fact that the automation system has taken over the relatively easy tasks and left the most difficult problems to be handled by the operator. As operators cease to exercise real manual control of the process, these skills will be degraded. Instead they must execute qualitative judgements, according to objectives that may be fuzzy and often conflicting (Sanderson, Verhage and Fuld 1989). The control actions now consist of relatively rare adjustments to the system parameters that are effected in a discrete manner through the automation system.

A large part of the operator's task is to monitor the trends of the variables under automatic control. The fact that the role of the operator has changed to a more cognitively challenging job under normal operation is in itself not a problem. However, when failures occur it may be impossible for the automation system to react in a flexible enough way to keep the process within normal operating conditions. Under such circumstances the operator is expected to revert to the traditional role of manual controller and maintain operation whilst locating, and preferably eliminating, the fault. (Bainbridge 1987) regards this expectation as unfair on operators due to the lack of training that they have in manual control. The situation is confounded by the fact that when something goes wrong with the process or the automation system, such systems often behave unpredictably. Consequently they are much more difficult to control.

Bainbridge further claims that the lack of experience in actually controlling the process will degrade the operators' deep knowledge of how it works. The kind of experience that is gained from observing how the process reacts to manual control cannot be developed whilst monitoring the process through the view provided by most supervisory control and information systems. Traditional operator skills are therefore being lost.

There are also difficulties with the new skills that operators must acquire to carry out their new tasks. Monitoring is a task that involves passive assimilation of information over long periods with relatively few actions. The human attention span may therefore be stretched. This requires adjustments to the working practices of operators in order to minimise the dangers of boredom. The monitoring task can be made easier by improving the quality of the information, for example by means of intelligent data filtering mechanisms and support through knowledge based systems. Multi-media approaches can also make tasks more engaging by improving the way in which this information is presented.

A final relevant aspect of the operator's environment is that each plant (process and associated control and information system) is purpose-built. The individual differences between plants are normally so large that a generalised approach to controlling chemical processes (for example) is not feasible (Wickens 1984). The most valuable experience for controlling a process is therefore gained only by controlling that particular process. This differs from, for example, the job of a TV- and radio technician, who has enough general experience in their fields to diagnose different devices using standardised strategies (Rasmussen 1986).

In summary, the main sources of difficulties for process operators include:

> The presence of three different systems: the process, the control and automation system and the information system.
>
> The dynamic nature of the process.
>
> The complexity of the process and automation system.

The large volume of process information.

The existence of multiple concurrent sub-processes.

The slow response from the process.

The cyclic causal relationships inherent in the process.

The need to control the process through "Hands off", discrete, outer-loop control mechanisms.

The infrequency of disturbances and the unpredictable behaviour of the process when they do occur.

The uniqueness of the process.

Communicating Process Information To The Operator.

A number of media are currently employed in communicating the state of the process to the operators. Most are visual (functional displays, alarm lists etc.) but audio techniques are also used mainly for alarms.

Functional Displays

Functional displays are used to communicate the state of the process of the operator. The mental models, which are required to diagnose complex process conditions and exert knowledge-based control in unstable conditions, are often externalised for the operators through such displays. In addition to the displays based on Rasmussens abstraction hierarchy, examples of displays which aim to externalise the mental models are described by (Beltracchi 1988) and (Bock 1988)

Beltracchi's (1988) display is a functional model display for nuclear power plants, which has been integrated with alarm information. The computerised model of the energy cycle in the process is based on the Rankine Heat Cycle and used to train operators of nuclear power plants. Although

Beltracchi does not report formal evaluation of the display, observations made during testing indicate favourable results, both for the tasks of monitoring and of diagnosing plant trips.

(Bock 1988) describes a computer aided process information system, PRISCA that has been installed in several nuclear power plants. It provides multiple views of process information through computerised displays. The highest level is a panel graphic of the whole process, offering the operator initial information in the event of a disturbance. Graphical representations, presented through dedicated displays, or process variables, overlaid with production target quantities, serve to focus the operator on the relationship between the current process state and the desired process state. More detailed and functional and physical views of the process are also available through the system. Illustrations of displays that clearly show information at different levels of abstraction are given in the paper, although the extent to which these have been implemented in plants is unclear.

Alarm Displays

Alarms are a key part of any functional display. Typically many alarm displays employ a *first-out technique*. This is in recognition of the difficulty of determining the root cause of a cascade of erroneous symptoms. Faults have a tendency to remain latent until sufficiently compromising conditions arise. From that point the propagation of the fault may be rapid. Built in security defences in areas of the plant which have no relation to the original fault may be triggered and the operator can miss the vital initial stages of the incident. Traditional annunciator systems also normally trigger a visual, and perhaps, an audio alarm for each discrepancy detected. Thus, the sheer density of visual and audio information may turn into an avalanche and become an interfering factor rather than a diagnostic aid. The 'first-out- approach is used to indicate to the operator which alarms were the first to be triggered, and the subsequent sequence of events.

In order to reduce the density of alarms, filters can be applied so that only the most pertinent alarms are relayed to the operator. The success of such an approach clearly depends on how well the system is able to discriminate between alarms. Because alarms mean different things in different context, a full understanding of their function is a difficult knowledge-based problem, which is hard to solve particularly when the operators are under pressure. Intelligent filtering of alarms is not always beneficial, however. (Corsbergh 1988) who reports on experiments evaluating, amongst other things, operator's response to filtered alarm information, showed that operators were uncomfortable about the possibility of losing important information.

A positive aspect of Corsbergh's displays was a link between the filtered alarm and the relevant process information. This allowed a partial or complete diagnosis of the problem to be obtained from the alarm display. *(*Baltracchi 1988) produced a similar alarm system. In this case, the fault-information displayed the discrepancies between process variables and their set points. Baltracchi's study reported positive effects from this integrated approach, although comparative studies with non-integrated fault-information were not carried out.

Human Errors In Process Control

Whatever the technique used to communicate information to the operators, they will inevitably make mistakes in interpretation, in diagnosis and in control actions. Errors committed by operators often have their root cause in a lack of regard for human performance limitations in the design of the operator's environment. It is therefore necessary for the designers of process control presentation systems take into account the tasks the operators are required to perform and the resulting cognitive demands. Designers should then intelligently use media to provide the most efficient way of communicating the process state and deviations to the operator. In order to understand problems that arise at the interface, we need to have a clear view of the job of an operator.

Process control operators are involved (Sohier & Bertels 1992) in five well-defined activities:

Monitoring

Monitoring involves checking critical variables to spot deviations from predefined set points as soon as possible. Generally this activity is highly automated and manual control is generally not required. However, it is not always such an inactive process. Automated systems generate a large number of warning indication, many of which do not signify real disturbances. Therefore, operators are often busy acknowledging alarms of little significance under normal working conditions.

Diagnosis

Diagnosis involves identifying the causes of deviations and deducing plans to stabilise plant conditions. This process requires information from a number of sources such as the process dynamics, the current process state, operating procedures and production goals. Diagnosis techniques may therefore involve a wide range of other activities such as information browsing and hypothesis generation/testing. Efficient diagnosis also depends on the extent to which the operator has formed an accurate view of the process state prior to when the deviation occurs. Efficient information processing is usually an essential ingredient of correct diagnosis.

Prediction

Prediction seeks to identify potential consequences of plant deviations in order to prevent them. Prediction and diagnosis are similar in that both involve speculations about causes and effects within a system. Whilst diagnosis is concerned with explaining the causes of an observed effect, deduction seeks to find the effect of a known cause.

Control

Control involves effecting changes in the operation of the process system. Thus, control can be the implementation of the plan identified during diagnosis or prediction phases.

Overall

During a typical operation, the operator may have to perform several of these activities usually beginning with monitoring, then diagnosis and prediction and ending up with the execution of control actions. However, this order can never be predicted as often complex tasks usually requires several iterations of solving sub-problems, involving further observations and predictions etc.

Although operators make genuine mistakes, a large number of errors have their root cause in the design of the system or in the human interface (Rasmussen 1986, Sanderson 1989)

The Nature Of Operator Errors

Types Of Errors

Reason (1992) distinguishes between three types of errors. The first type, *slips* and *lapses* occur when an unintended actions takes place often caused by attentional failures or memory lapses. The second type, *mistakes* are intended actions that happen to be wrong. They can be divided into two sub-categories. A *rule-based* mistake is an action triggered by a known rule, which is inappropriate under the circumstances for whatever reason. A *knowledge-based* mistake on the other hand happens when there are no known rules to apply and the operator incorrectly deduces the action to be taken. The final type of error is *violation*. This is an unsafe act carried out in good faith, contrary to the prescribed procedure.

Five of the seven acts categorised as unsafe in the Chernobyl accident were violations (Reason 1988). The fact that operators saw the need, in adverse plant conditions, to violate prescribed procedures indicates that the procedures were inadequate. However, the problem is more fundamental than this. (Vicente 1991) distinguishes between events in a system that are anticipated by the designers and those that are not. Anticipated events can be dealt with through procedures or other forms of explicit support for the operators that help to minimise the likelihood of knowledge-based mistakes occurring. However, by definition there is no way of prescribing responses to unanticipated events. The operators in these cases must rely on experience as a basis for forming a knowledge-based diagnosis.

In his work on identifying human errors in process control environments, Hollnagel (Hollnagel 1993) has made an important distinction between the underlying *cause* or *genotype* of an error, and the observable *manifestation* or *phenotype* of the error. Most work on human error has been primarily concerned with the causes of error rather than the errors themselves. However, as Hollnagel points out it is often a mistake to mix the classification of observable phenomena with the interpretation of their causes. For example, an operator who fails to carry out an action is often said to have "forgotten" the action, when actually the observable manifestation of the error can only be seen as an omission, which may or may not be rooted in the operator forgetting. Furthermore, depending on the purpose of identifying the error, the cause of he omission may be irrelevant. Once the agent is "reminded" to do the action, the reason they omitted it may not be an issue.

In (Hollnagel 1993) Hollnagel sets out to define a classification of observable errors, or what he calls *error phenotypes*. Rather than relying on an analysis of actions and plans in a limited domain, he begins by enumerating all possible errors involving a single action that can occur in a generic plan, defined as a totally ordered sequence of actions all of which address a single goal.

The phenotypes are classified according to the level of observation or inference needed to identify them. *0-order* phenotypes are those that can be detected based on the observation of a single action together with an expectation for that the next action will be. The more complex, *1-order* phenotypes are derived from the combination of two or more o-order phenotypes. The 0-order phenotypes are simple to define and can be identified immediately in a plan since they involve only a single action. On the other hand they do not allow us to get a bigger picture of what is going wrong with the plan, since the underlying cause of the error may effect a whole sequence or sequences of actions. Looking at errors involving sequences of actions, simple phenotypes can be expanded into larger, more complex set of 1-order phenotypes. The 1-order phenotypes have the serious disadvantage that they cannot be recognised unambiguously on the basis of a single action.

Hollnagel's error definitions rely on the characterisation of a plan as a totally ordered sequence of actions that are necessary and sufficient for achieving a goal. This implies that once the observer has determined the operator's goal, they then know every action that the operator should perform to achieve that goal, and in what order they should be performed. This assumption is not justified in most realistic situations for a number of reasons.

Firstly, this characterisation of plans does not take into account the possibility of alternative ways of addressing a goal. Therefore it does not allow the classification of situations in which the operator is addressing their goal, but in a sub-optimal manner. Secondly, it also does not consider the interactions that may occur when multiple plans are executed concurrently. For example, a *realistic* system must be able to recognise which of the currently active plans an action is intended to participate in.

Finally, while Hollnagel defines plans in terms of totally ordered action sequences, in general, the actions in a plan do not have to be totally ordered. It may be up to the operator executing the plan to decide what to do first, and some actions may be done in parallel. The relative ordering of actions in a plan can be constrained by a number of relationships or interactions, including precondition achievement, logistical or resource constraints, and avoidance of contraindications. Additionally, defining plans as totally ordered sequences is many years behind the state of the art in AI planning. Contingencies and non-linearity's are important these days. Tasks are sequential actions when there are no other choices for the operator, therefore this assumption is rather naive.

Design Methodologies For Process Control Interface Design

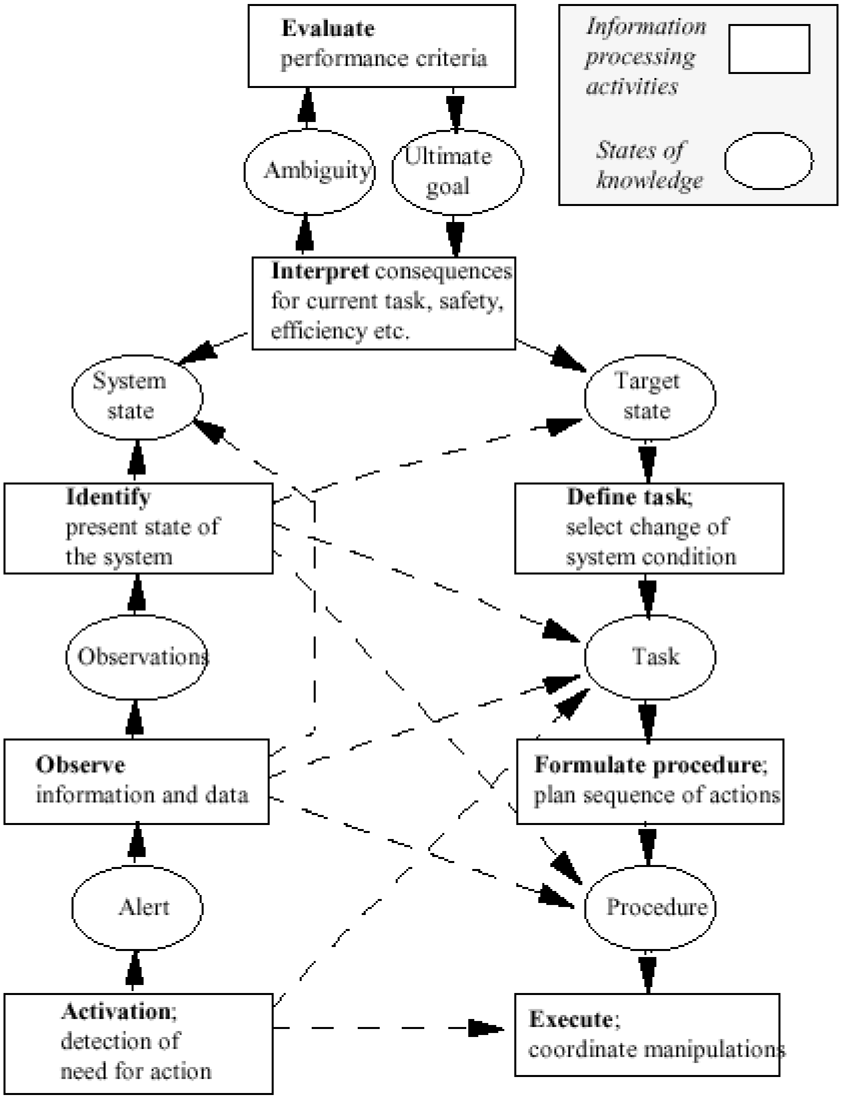
Rasmussen's Decision Ladder

Rasmussen (1986) has proposed a generalised model (the Decision Ladder) to describe the sequence of cognitive information processes that are involved in process control decisions (see Figure 6). The model has been derived from verbal protocol analysis of operators of real process plants. It identifies the mental processes without assuming how each process operates or what kinds of knowledge structures are brought to bear.

The outcome of an information process is a new "state of knowledge". An information process may produce different states of knowledge depending on the experience of the operator and the certainty of the situation.

The sequence indicated by the solid arrows in Figure 6 is the kind of processing required for difficult problem solving, that is, for problems that involve a high degree of uncertainty. Each step in the ladder ascending the left hand side poses questions, which are increasingly abstract, and which need to be answered through an information process. Ultimately the operator must consult the overall objectives of the process to evaluate what the desired outcome of the actions should be. The descending right hand side comprises the decomposition of goals into tasks, procedures and actions that are physically carried out. However, most disturbances in process plants are minor and are responded to by routine operator actions.

The stepped arrows, that bypass various phases in the decision sequence, represent these more direct ways of initiating the correct actions. The Decision Ladder is activated when a deviation is detected during monitoring, and terminated when the control action is taken. Thus, this model can be used to describe the complete decision task in a closed loop with the environment. However, not every decision will have a control task as its outcome. During the monitoring of normal operations, for example, the operator may continuously engage in part of the analysis phase on the left hand side. That is, observations are made and the current, predicted and desired states of the process may be evaluated. If the predicted state is the same as the desired state no action is needed and the chosen task is said to be a null task. Rasmussen's Decision Ladder is an example of an operator model.



Evaluate performance criteria
Information processing activities
States of knowledge
Ambiguity
Ultimate goal
Interpret consequences for current task, safety, efficiency etc.
System state
Target state
Identify present state of the system
Define task; select change of system condition
Observations
Task
Observe information and data
Formulate procedure; plan sequence of actions
Alert
Procedure
Activation; detection of need for action
Execute; coordinate manipulations

Figure 6 Rasmussen's (1986) Decision Ladder

Rasmussen's Abstraction Hierarchy

The Decision Ladder is based round the classification of information processing into three types: skill-based, rule-based and knowledge-based (Rasmussen 1986). The premise for this theory is that humans are equipped to control their environment according to abstract objectives and there are three layers of processing which help them in achieving this.

At the bottom of the hierarchy are the sensory and motor skills (**skill-based behaviour**).

> *"Skill based behaviour is control of activities (that) require on-line, real time control based on tacit knowledge that cannot be described by the actor. It depends on interaction with temporal-spatial configuration of objects that can be real material objects or configurable representations of concepts"*(Rasmussen 1994*)*

These acts require no conscious control, and function independently of central processing and working memory. Skill based behaviour is often exhibited as people learn to master a task involving sensory-motor processing. Tracking an object round a screen with a joystick driven cursor is an example of such behaviour. Each time an action is taken (change direction or speed) the response must be observed and used to determine the next action. Thus it is a stimulus-response chain, which forms a closed loop with the environment. However, human beings are also capable of ordering their motor system to perform sequences of actions without relying on feedback between actions. Writing a signature is an example of this. Composite actions can be performed in the absence of feedback by executing what is called a motor programme. Such programmes (skills) can be built-up with practice, and be controlled by higher levels of cognitive behaviour.

In particular, a sequence of actions may be activated by a stored rule or procedure. This is **rule-based behaviour** or "know-how".

> *"Rule based behaviour is the generation of proper organisation of patterns of movements into plans depending on access to stored rules and to experience from past work scenarios. The planning is done ahead of the action,*

> *that is, it is not synchronised with the interaction and is based on recall of past experiences and imagination of future encounters. It depends on the availability of convenient cues to release acts, cues that are only conventional signs with no functional significance"(*Rasmussen 1994*)*

Typically, this indicates the operator is functioning within a well-defined and pre-set method for dealing with a problem. That is they have identified the symptoms of a problem, and are acting in a regulated and rigorous method to resolve the problem.

In and adaptive system, it may be possible to deduce the nature of the problem the operator may be working on from their behaviour. For instance, if the operator's actions match a routine for shutting down a valve, the system may assume that is what the operator is doing. It can thus compare expected actions with operator actions; if there is divergence the system may act (adapt) to converge the actions of the operator towards the expected.

In contrast, the highest level - **knowledge-based behaviour** - is invoked by the absence of previous experience of a situation.

> *"Knowledge based behaviour is based on symbolic, mental model representing the deep, internal sources of regularity of the behaviour of the work, environment and information is interpreted symbolically with reference to this model"*(Rasmussen 1994)

This is where the operator faces a problem but does not, at that time, know the solution. The operators utilise the available information, along with their experience to deduce the nature of the problem. This is an area where an adaptive system can really aid the operator. Since the adaptive system would have knowledge of the process, and of the operators' actions, it could attempt to ascertain the problem, and then help to point the operator towards the appropriate solution

Rasmussen's taxonomy provides a framework for the understanding of information processes of human perception and cognition. It also broadly maps behavioural types on to the Decision Ladder in Figure 6. At the bottom of the Figure, in the initial and final phases of the decision task, skills are required. Direct stimulus-response behaviour occurs across the bottom from left to right. If the problem is relatively well known, a stored rule will be triggered from which shortcuts can be used to move from observations on the left hand side to task selection on the right hand side. So rule based behaviour resides in the middle section of the diagram. The knowledge-based domain is at the top where goals are explicit, the environment is considered in an abstract way, and plans are evaluated.

The methodology proposed by (Rasmussen 1986) for the design of supervisory and control systems incorporate the three frameworks described thus far: the Decision Ladder, the skills, rules and knowledge behavioural framework and the Abstraction Hierarchy. The methodology assumes that the starting point is a physical plant with certain characteristics and control requirements. It therefore involves all aspects of the control design.

Initially the control requirements of the system are represented as constraints at all levels of abstraction from the operational objectives down to individual process components. Safety requirements are also introduced as constraints. The decision tasks that are needed to fulfil these constraints must then be specified. This can be done using a Decision Ladder for each task. The variability in the knowledge requirements for different decision tasks can be effectively represented in the ladder. The decision sequences can be annotated with external data and knowledge requirements for each step, in the form of references to entities in the abstraction hierarchy. Simple feedback loop decisions can be identified at this stage and earmarked for automated implementation.

The next step is a cognitive task analysis, which involves identifying the various information processing strategies that can be used in each processing phase in the decision tasks. In other words, the information that the designer has identified as the desired outcome of a processing stage must be discerned by the operator. The various ways in which the operator (or the automation system) can do this must be made explicit so they can be accommodated in the implementation. A number of diagnostic processing strategies are discussed by (Rasmussen

1986). The tasks can now be allocated to either the human operator or the computer system. This should be done to optimise the decision making performance, bearing in mind human behavioural characteristics and information processing limitations as well as software and hardware resources and limitations. The final phase is to design and build the systems, the control room, the operator interfaces and training procedures.

Although Rasmussen's approach incorporates some useful formalisms, it is not a detailed design methodology. Indeed, the method stops when the actual system design begins. The following three points about the method are important to note in this context, though:

> Thorough analysis of the system to be controlled at multiple levels of abstraction is essential. If the control requirements are not known by the designer in such detail it is unlikely that interfaces, which effectively support the operator, will be produced.
>
> Rasmussen assumes that the system is designed for a new process. However, it is often the case that information systems are designed for existing plants with an already operational control system. In such cases the control requirements that must be identified are already constrained by the functions of the automation system.
>
> Decisions and information processing strategies in problem solving must be explicitly represented. Models of human behaviour can inform such representations. To be useful in interface design, however, these representations must also make explicit the items of information required by the operator to fulfil the strategies. These items of information range from production objectives for higher-level decision making, to the state of an individual component for low level decisions.

To fully incorporate the abstraction hierarchy a full system analysis would be required, which is often in process control a very difficult task, other problems with the abstraction hierarchy can be found in (Lind 1999).

An alternative to the design method proposed by Rasmussen, is the grammar-based methodology called **Multi-level Flow Modelling (MFM)** (Lind et. al. 1989). MFM represents an approach to modelling intentional knowledge in the domain of complex process control. By taking into account not only the physical component level representations of the plant, but also the objectives and goals a deeper, more meaningful insight can be achieved as to the state of the process. This knowledge can then be used to aid in diagnosis of problems within a process. MFM utilises abstraction along *means-end* and *part-whole* dimensions and utilises flows in terms of mass and energy to interpret process behaviour in terms of goals. Since, MFM integrates reasoning about goals and functions within a single framework, in times of disturbance it can support the operator in assessing the problem and deducing its cause

However, one of the main problems with MFM is finding a suitable presentation format for the model to be displayed in. The symbols used by MFM may be unfamiliar to the operator, and it is still undecided as to the best means of presenting abstract concepts such as goals, objectives and functions in a meaningful way to the operator. Still, this is a very useful methodology and could be utilised successfully as a means of constructing the Process Model Agent, and loading it with appropriate conditions. Additionally, MFM representations could be set as appropriate representations for the adaptive system to switch to if it seems the operator is confused as to the nature of a process problem.

Another method for interface design, which is related to both Rasmussen's approach to supervisory and control system design, as well as Lind's **Multi-level Flow Modelling** approach, is **Ecological Interface Design** (Rasmussen & Vicente 1987, Vicente & Rasmussen 1990, Vicente 1991). This method implements a cognitive approach to interface design based on the abstraction hierarchy. It is also said to have a base in the field of ecological psychology, which emphasises the duality between an organism and its environment. A founding principle is to provide a rich information environment, which renders the process system at all levels of abstraction. This is intended to improve knowledge-based behaviour by reminding the operator of the composition of the system (Vicente & Rasmussen 1987). The principle of making system component states visible is also advocated as beneficial for the formation of mental models by (Wickens 1992). Rather than basing the method on a description of decision tasks which would ultimately derive what information was needed and when, the approach elegantly side-steps this problem by proposing to make all the information visible all of the time. In the example design described by (Vicente 1991), which is a graphical interface for a small scale but complex simulation, this principle is followed to the extent that selected items from the different levels of abstraction are shown.

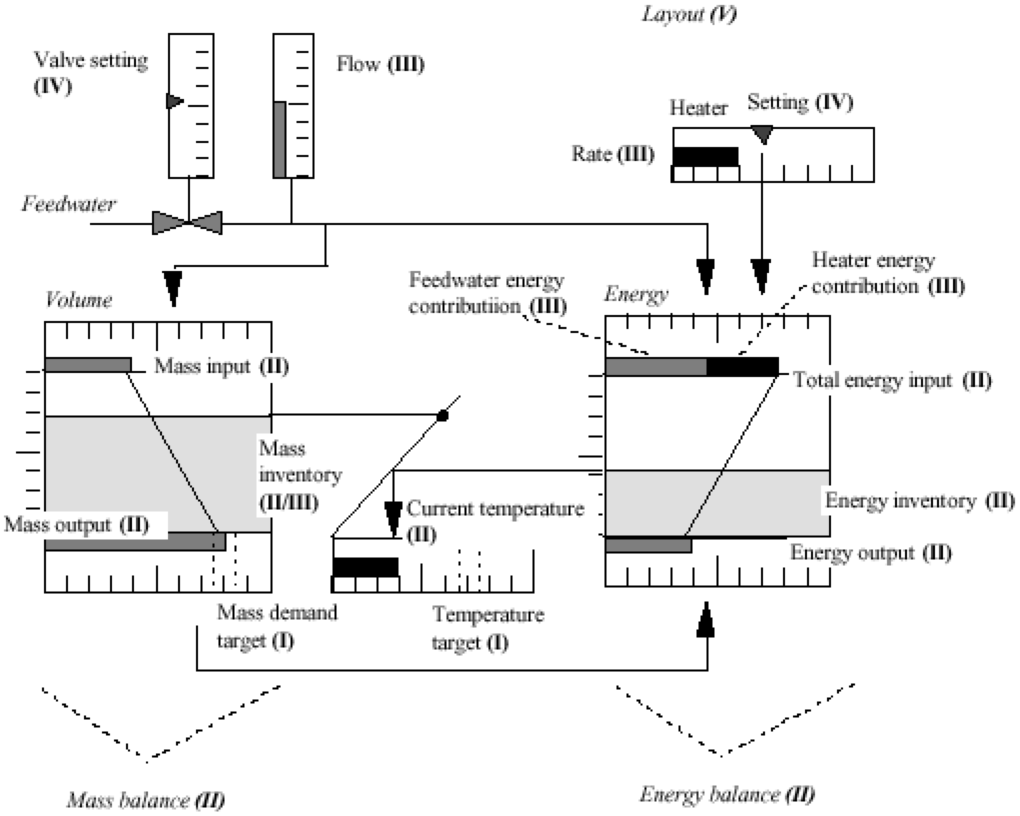


Figure 7 Vicente (1991) Ecological Process Display

Figure 7 shows part of the interface display used by (Vicente 1991) for the mass and energy flows in the DURESS simulation. The roman numerals represent the level in the abstraction hierarchy for that item. Legend:

Functional Purpose (to maintain demand and temperature) (I)

Abstract Function (represent mass and energy causal structures separately) (II)

Generalised Function (elements of standard heating and liquid functions) (III)

Physical Function (variables in the physical system that can be controlled) (IV)

Physical Form (the layout of the components) (V)

Figure 7 illustrates the 'richness' or information density that results from this method. It should be noted that the actual simulation used consisted of 2 identical vats, 2 pumps and 6 valves, which actually made the display considerably denser than the impression given in Figure 7.

The information shown in the display has been selected through a detailed, formal representation of the system at all levels of abstraction. This means-ends decomposition of the complex system starts with the purpose at the top, which is to maintain the system within certain flow demand and temperature targets. Next, the mass and energy balance is represented in terms of sources, sinks and storage. This is equivalent to the input, inventory and output represented in the integrated geometric shape for each system in the display. The levels of generalised and physical function bring in rates of mass flow and heat transfer as well as component states such as heater and valve settings. The layout, which is said to represent the physical form, shows connections between components. However, the fact that the components have been separated into distinct abstract functions somewhat obscures the physical form.

(Vicente 1991) compared operator performance using the above interface to that achieved using a display, which only rendered the physical system. The 'ecological' approach produced a better performance. However, it was more difficult to operate and was thus only fully exploited by expert users. Tests on memory recall indicated that expert users who did not have the functional information available, attempted to derive the missing information during diagnosis.

This experiment is an elegant demonstration that knowledge-based reasoning can be actively supported by making artefacts visible and emphasizes the role appropriate multimedia representations might play in aiding the operator decision process. A word of caution is needed regarding the approach, the demonstrator process is very small scale. For very complex processes, the notion of continuously rendering information at that level of detail is likely to stretch the attention limitations of the operator. A more flexible layering of the information, so that operators have some control of what information is displayed is desirable (Lind et. al. 1989). Ideally both the level of aggregation and the level of abstraction shown should be adjustable.

Using Multi-Media In Process Control Interfaces

Studies have developed multimedia guidelines for use directly with process control. (Alty et al. 1992) details a study of the use of various media combinations in supporting a process control task in which flow-rate of water into a water tank had to be controlled. The task required a number of types of knowledge, such as spatial information concerning water level, action information concerning valves and a heater, and causal information concerning relationships between these components. The study made use of a number of presentation conditions: text versus graphics, sound versus no sound, speech versus no speech. The usefulness of media depends upon the problem situation. For example, the sound of the flow of water into a water bath was provided for the operator. However:

> For simple tasks, the sound of the flow rate in the study had a detrimental effect upon performance and understanding of the task: "use of a complex naturalistic sound may not have the discriminability required"
>
> For more complex tasks, sounds improved performance: "the clear pattern that emerges.is that operator performance with interfaces using sound was better for difficult

tasks than for easier ones." This could be used within our adaptive system in the case of not utilising sound when there is no disturbance within the process. However, if the situation becomes complex it may be valuable to introduce sound to improve task efficiency.

Speech warnings improved performance, but may be overly dominant in a presentation: "speech did seem to improve performance during learning but is very intrusive"

Graphical representation improved performance: "for this particular task graphical representation proved to be the most preferred. Thus our contention that this spatially orientated task would be best presented by graphics is born out by experiment".

Conclusion

Process control is an ideal domain for the application of an adaptive interface due to its complex nature, and because of the large quantities of information that an operator must deal with. Typically in times of disturbance the operator must deal with information overload. The designers of a process control interface have many concerns, including the safety of the process and the potential economic savings that can be made when the interface supports the process correctly. Both of these concerns can be met by the use of an adaptive interface. Safety can be improved with an adaptive interface by helping the operator diagnose a disturbance in the process, by configuring the presentation so that only the most salient and relevant information is shown. Additionally, that the information shown is rendered in the best possible fashion so that the operator can more effectively understand the nature of the problem. By improving safety, significant savings can be economically by avoiding expensive and potentially unnecessary shut downs of the process.

In recent times automation has been introduced in many types of process with the aim of abstracting away from the operator many mundane, routine tasks. This has alleviated the operator from a great deal of "hands on" interaction with the system, and has often shifted their role from control to monitoring. This introduces some problems for the operator as when the process does deviate from normal operation to a disturbed state, the operator is frequently left unprepared, due to their inexperience in controlling the process. They are however aided in their tasks by improved display interfaces such as functional displays and advanced alarm filtering systems.

However, operators are still prone to making errors and this is where an adaptive system can help. When operators make knowledge-based errors, it is often due to an incorrect understanding of the problem, frequently caused by incorrect interpretation of the data. An adaptive system can help by utilising different media to highlight relevant data and configuring the interface so that the operator is equipped with the correct information, clearly.

For an adaptive system to function it requires a good process and operator model. Rasmussen's Decision Ladder and Abstraction Hierarchy provide a good basis for constructing an operator model. Lind's Multi Level Flow Model provides a good basis for deducing a Process Model, and also for providing a representation that could potentially be useful in a process disturbance. Another abstract representation that could be useful when the operator loses sight of the nature of a process disturbance is the Ecological Representation, which holistically presents the process at various levels of abstraction.

This chapter has shown that an adaptive system can be of use in the domain of process control and has presented some useful methodologies for constructing some of the necessary adaptive models.

The next chapter examines the underlying paradigm that the adaptive architecture described in this thesis will use, software agents.

Chapter 4

SOFTWARE AGENTS

What is an Agent?

There are many different definitions of agency and it is therefore rather difficult to arrive at an exact definition. The definitions range from this rather simple, broad description of agency:

> *"A component/software object/hardware that is capable of acting exactingly to accomplish tasks on behalf of its user."* (Nwana 1996)

to the more explicit, and tightly defined terminology of (Hayes-Roth 1995):

> *"Intelligent agents continuously perform three functions: perception of dynamic conditions in the environment; action to affect conditions in the environment; and reasoning to interpret perceptions, solve problems, draw inferences and determine actions."*

Maes of the software agents research group at MIT has her own definition:

> *"An agent is a computational system that inhabits a complex, dynamic environment. The agent can sense, and act on, its environment, and has a set of goals or motivations that it tries to achieve through these actions."*

FIPA (Foundation for Intelligent Physical Agents) uses a strawman definition of agency for all their specifications (FIPA1). They define an agent as:

> *"an entity that resides in environments where it interprets "sensor" data that reflect events in the environment and executes "motor" commands that produce effects in the environment. An agent can be purely software or hardware. In the latter case a considerable amount of software is needed to make the hardware an agent."*

For some researchers - particularly those working in AI - the term 'agent' has a stronger and more specific meaning than that sketched out above. These researchers generally mean an agent to be a computer system that, in addition to having the above properties, is either conceptualised or implemented using concepts that are more usually applied to human beings. For example, it is quite common in AI to characterise an agent using mentalistic notions, such as knowledge, belief, intention, and obligation (Minsky 1985).

A good description of agency has been given by Franklin and Graesser (Franklin & Graesser, 1996):

> *"An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future"*

This definition was achieved after a careful analysis of the features common to several different agents. Although this definition is not universally accepted, it will suffice here as a means towards understanding the basic nature of agency.

Why Have Agents?

There are a number of reasons for employing an agent-based approach. One important reason is the increased requirement for *inter-operability* between systems. This is now regarded as one of the most important challenges to the software engineering community. Agents can provide a convenient means of bridging the gap between heterogeneous systems without requiring significant re-engineering of existing or legacy systems.

Agents-Based Software Engineering provides a convenient means for controlling this inter-operability. It does this through facilitating communication at a high level via a standard *agent communication language (ACL)* (Genesereth, M. R. & Ketchpel, 1994). ACL allows agents (that can be as small as sub-routines) to exchange data, scripts, small programs and so forth. This ability is significant if agents are to have some degree of social ability, defined as the capability of an agent to exist co-operatively within a set of agents, communicating with them, sharing tasks and delegating sub-tasks. Social ability itself is vitally important as an enabler for agent technologies, since it allows any agent with insufficient expertise, in a certain area, to confer with peer agents that *do* have that knowledge and can thus provide the required expertise. This process allows synergy to occur between agents, the whole being greater than the parts.

In addition, as augmented and ubiquitous computing (Rekimoto 1996) becomes more commonplace (as it inevitably will) the need to reduce the amount of human-computer interaction that relies on *direct manipulation* (Schneiderman & McGill 1988) is increasing. This is because of the recent trend for software packages to incorporate increased functionality at each new release, leading to current desktop being excessively feature rich. (Alexandros 1995)

In the search for a new approach, it was suggested that a complimentary method to *direct manipulation* could be used whereby the computer, via an agent, could guide and help the user to deal with, and manage, information overload. This method is termed *indirect management* (Kay 1989). Agents, therefore, could provide an attractive way of shifting the interface paradigm away from direct manipulation to the domain of *indirect management.* This is because agents work within the scope of the *direct manipulation* domain, but allow the user to better manage desktop processes by abstracting some of the complexity away from the user. There have been many debates about the relative merits of the indirect management approach, particularly when applied to *Interface Agents*. Schneiderman (Maes & Schneiderman 1997) is a fierce advocate of the direct manipulation paradigm and argues that interface agents take elements of control away from the user. Moreover, he argues that interface agents are unnecessary. The main drive of his argument is that the cause of deficiencies in the *direct manipulation* paradigm is not due to an intrinsic flaw, rather to bad implementation. Direct manipulation, he states, is still the most powerful interface interaction tool, and if correctly designed and implemented makes the use of interface agents redundant. However, since most direction manipulation interfaces cannot be re-designed, interface agents offer a powerful alternative solution.

The agent-based approach to the indirect management paradigm is a hybrid of the two approaches and likely to be a temporary solution to this important problem. The next stage of interface design evolution towards indirect management may depend heavily on the success of current agent technology, and the confidence it engenders in users. Confidence is a key factor, since if users can be induced to engender trust in their agents to perform delegated tasks, then a full shift to the indirect management approach might occur. In the ultimate scenario, the interface would be composed of a quorum of agents each having expertise in its own area, and all agents communicating between each other, and remote distributed agents to solve interface problems. The users would merely make their requests to their local agent, and the agent would then deal with all the technology-based problems, allowing the users to concentrate their primary task (Negroponte 1989).

As the present time this vision is still some distance away, and currently most agents remain as support mechanisms working in parallel with the direct manipulation interface. They observe the user and make suggestions of better ways of doing things. They may also offer users better ways of searching for information and filter the resulting data to provide more context sensitive answers.

If agents can actively seek, filter and abstract information for users, this could become an important application. As the size of the Internet increases and bandwidths increase, more information is ported to the user at high delivery rates, it becomes increasingly hard for the user to find the information they require from a surfeit of irrelevancy. If agents are to gain widespread acceptance in this killer filtering operation, they will need to engender a sense of trust in their users (Alexandros 1995)

The most powerful tools for handling complexity in software development are *modularity* and *abstraction* (Jennings & Wooldridge 1998). Agents represent a powerful tool for making systems modular. If a problem domain is particularly complex, large, or unpredictable such as this case within process control, then it may be that the only way it can reasonably be addressed is to develop a number of (nearly) modular components that are specialized (in terms of their representation and problem solving paradigm) at solving a particular aspect of it. Process control is a natural application for intelligent agents and multi-agent systems, since process controllers are themselves autonomous reactive systems. In such cases, when interdependent problems arise, the agents in the systems must cooperate with one another to ensure that interdependencies are properly managed. The agent-based approach means that the overall problem can be partitioned into a number of smaller and simpler components, which are easier to develop and maintain, and which are specialized at solving the constituent sub-problems.

This decomposition allows each agent to employ the most appropriate paradigm for solving its particular problem, rather than being forced to adopt a common uniform approach that represents a compromise for the entire system, but which is not optimal for any of its subparts. The notion of an autonomous agent also provides a useful *abstraction* in just the same way that procedures, abstract data types, and, most recently, objects, provide abstractions. They allow a software developer to conceptualise a complex software system as a society of co-operating autonomous problem solvers. For many applications, this high-level view is simply more appropriate than the alternatives.

Having established *what* agents are and *why* they exist, the next step is to examine closely the different agent types and their properties and attributes. This examination will lead to the production of a typology of existing agents. The difficulty in achieving a standard definition of agency, because of the wide and diverse types of existing agents and their relationships, will become clear from this typology. It will also facilitate the introduction of terminology to be used in critiquing between the different agents. Agents will first be classed by application, then by function, and finally by their properties.

Agents classed by Application Area

Agents can be classified in terms of the application area in which they are applied. This is not a particularly useful classification, but it does illustrate the breadth of their application.

Entertainment Applications.

These are agent systems that include:

Real-time and non-real-time (store and forward) user avatars for messaging, low bit-rate communication, and shared virtual environments

Games (autonomous interaction between game characters and with environment & multi-player games)

Gaming and avatar applications deployed in theme parks, arcade high-end game machines and

Film/video production: I) camera agents (film/video cameras with focus, reactions, etc.), II) 3D graphical agents for storyboard design agents and avatars in computer animated feature films, cartoons

Service Management Applications.

These are systems that involve configuration delivery of user requested services at the right time, cost, and QoS (Quality of Service) required security and privacy issues. Examples include:

Multimedia services

Buy/selling services (e.g. information, material goods)

TMN/ intelligent network management services, and

Trip planning and guidance services (e.g. intermodal route parking-lot reservations, individualized traffic guidance, tourism

Business Management Applications.

These systems deal with management of business tasks and resources in provision of services and carrying out business operations (Jennings, Alty et al 1996)

They include:

Financial services,

Electronic commerce

Workflow management (Levitt et al 1994)

Office automation

Computer Supported Cooperative Work, and

Telecommuting

Manufacturing Management Applications.

These systems involve physically embodied agents designed to carry out management in relatively structured industrial environments. These processes may involve the control of industrial robots and machines via software interfaces. Some common manufacturing applications areas are:

Industrial Robotics

Factory automation (Baker 1996)

Virtual factory management, and

Load Balancing.

Service Robotics Applications.

These systems involve physically embodied agents designed to carry out tasks and processes in relatively unstructured office and domestic environments (e.g. office mail delivery, house cleaning, etc.).

Cooperative Tasks Management Applications

These systems involve collection of robotics and software agents that are being coordinated to achieve higher level tasks.

Research Applications

These systems involve using agent technology to further research in other (IT) areas such as:

Vision processing

Learning and adaptive systems

Speech processing

Distributed knowledge-based systems.

Agent Classes Defined By Function

Alternatively, agents can be separated into four different types of agent - *Interface, Information, Computational* and *Facilitator* agents. For each category, the agent of that type serves a different and orthogonal purpose to agents of other categories.

Interface Agents.

The paradigm for the existence of Interface Agents is based on Kay's (Kay 1990) dream of indirectly managed interfaces. At present direct manipulation interfaces are the norm, despite offering little or no *proactive* help or assistance to the user when performing tasks (apart from affordance). To rectify this problem, the indirect management approach offers a vision of user and computer co-operating together to complete a user task with greater ease and efficiency. For this vision to become reality, agents will have to engender *trust* in their users by performing their tasks *competently*. Otherwise users will never fully utilise the power of indirect management because they will not be willing to fully delegate many technology-based tasks to an agent.

At the moment the best practical example of the application of indirect management principles is the use of an agent as a *personal assistan*t. In this case the interface agent sits on a direct manipulation interface and aids the user by performing time-consuming, laborious tasks delegated to it. It performs a similar role to that of a human *personal assistant*, because users can entrusts their assistants to perform certain mundane time-consuming tasks, leaving them free to concentrate on more important goals.

For an Interface Agent to be able to perform these laborious tasks it is important for the agent to gain some knowledge of the user, such that a context can be drawn on the user's actions. With such a context in place, the agent can understand and perform the user's request more efficiently and with less scope for error. Thus Interface Agents need to have some degree of intrinsic learning ability or a means of understanding the full system context in which the user is acting.

This learning ability usually takes the form of an agent observing the user's actions on the interface, and then trying to match action/reaction pairs in an attempt to find patterns in the user's behaviour. If the patterns start to become repetitive, the interface induces a connection and will use this to recommend to the user ways that it can perform these repetitive tasks on their behalf. However for this approach to work, the behaviour needs to be substantially repetitive.

Typical examples of Interface Agent applications are (from Maes group at MIT)

> Eager Assistants (for instance a calendar agent that has learnt that the user prefers meetings only in the afternoon and therefore negotiates with other calendar agents on this basis).
>
> Filter/Critics (Filters selected mail)
>
> Matchmaking/referrals (An agent uses its knowledge of user preferences to contact other user agent's with similar preferences. It can then forward any recommendations the other user agent makes on those preferences)

Intelligent Filtering of unnecessary information, allowing the user to concentrate on the task in hand.

Information Agents

The amount of information published doubles every year, most being published on the web. The goal of the *Information agent* is to sort through these vast amounts of information and find the specific information that the user requires. Typically, an information agent's job is to understand the context of a user's request, and then to access several on-line databases, search them and retrieve relevant data. It must then filter the resulting data according to the user's specific request so that only the most relevant information is returned.

Information agents are usually server-based (and require a log-on). However, recent trends have seen a move towards client-based interface agents. The presence of an interface agent on the client side domain will allow the agent to observe the user's actions directly and thus obtain better contextual information. *Information* agents are similar in many ways to other types of agents in that they have similar knowledge boundaries but they are restricted to their domain. (Sycara 1995)

Computational Agents

Computational agents serve a rather simple function, to perform on behalf of the user computationally difficult tasks. They exist so that the user can delegate tedious computationally time consuming tasks to the agent, therefore freeing themselves to concentrate on their main goal. Typically, a computational agent might be asked to perform mathematical problems or to facilitate complex install procedures. Often, this type of agent is a quite simple algorithmic-based system, which typically implement a "learn by example" technique. That is, the user will initially guide the agent through a complex procedure. Once the procedure has been completed, the user defines the limits of the agents' control, and instances of when it should perform its operations. Once these pre-requisites have been complete, the agent (much like a sophisticated macro) will imitate the users approach for problems it sees as being of the same class initially defined by the user. This type of agent, although simple, is quite powerful and can be of great use.

Facilitator Agents

This class of agent serves to link and match agents of differing expertise together. Therefore they utilise a meta-class knowledge base. Their skill is to know where such knowledge can be found rather than directly returning knowledge in an expertise area. They then attempt to connect the requesting agent to the agent that has the required knowledge.

It is obvious that this type of agent will be most effective in an multi-agent system (*See Chapter 5)* existing in an open environment (such as the internet). Facilitator agents perform a vital role in the provision of information and services between a set of distributed agents.

Describing Agents in terms of their Properties

As an alternative to *classification by function*, agents can be defined in terms of a number of properties that they exhibit. A property can take any value between two extreme values, for example *Autonomy and Slave*, or *Mobile and Static*. In this section we will define a number of property axes and later use these axes to characterise agents. This characterisation is quite distinct from our earlier characterisation in terms of function.

The Static - Mobile Axis

A fundamental distinction of agent types can be made between *Mobile* and *Static* agents. Mobile agents have the ability to move around a network/distributed system, whereas Static agents do not. They access servers and perform tasks on behalf of the user. Mobile agents may report back to proxy agents, which have requested some task on behalf of a user, as well as to the users themselves.

An example of a mobile agent performing a value-added service for a user is the air-flight booking scenario. In this scenario a user requires the cheapest flight possible on a certain date, and instead of browsing various airline servers at high cost in both network time and resources, utilises a mobile agent. The mobile agent visits all the booking services for cheap flights, and finds the best buys available within the budget set by user on the desired date. Upon completion of its task, the mobile agent returns to the user (synchronously) with the relevant flight information with which the user can either book a flight, or ask the agent to go back to search again. The advantage of this method is that it provides a cost efficient way of accessing information because the user does not have to be online whilst the agent is performing its task All the results can be returned via email, thus saving network connections and hours of the user's time.

The Mobile Agent approach also suggests an innovative way of providing a distributed network, and facilitates radical ways of considering the design process. The existence of Mobile Agents, for example, will allow computers with limited local resources to carry out more complex operations by routing a mobile agent to a resource centre that can accomplish the task, and awaits the return of the results.

An agent may show some degree of mobility. This quality is not an absolute attribute and will vary depending on the task the agent is built to handle. For instance an agent constructed to collate data on aeroplane timetables and book the cheapest and most convenient flight will *de facto* have to enter the network and move around it to various online databases, and thus demonstrates a great deal of mobility. However, if the *agent*'s task is to be a guide operating on the interface on a stand-alone machine, it will only demonstrate the weakest of mobility.

The Reactive - Deliberative Axis

Another basic distinction can made between Deliberative and Reactive agents. A Deliberative agent is an agent that will react to a situation by referring to its internal expert system or symbolic reasoning model and then uses this knowledge to engage in planning and negotiation activities in order to co-ordinate and co-operate with other agents. Reactive agents on the other hand do not have this symbolic reasoning ability, and instead rely on a stimulus/response paradigm, whereby they react to the environment in a hard-wired fashion.

Much of the prevalent theory in AI suggests that *deliberative* symbolism is one of the key functions of agency, meaning that an agent should have an internal plan representing actions, goals and events that determines its reactions to real world situations. This approach has been contested for its inflexibility, after all - it is argued - although human beings in real life may follow a plan, there is much improvisation in the details of the plan. This - researchers argue- cannot happen with deliberative agents (Agre & Chapman 1987), and so the concept of *reactive agents* was suggested, whose choice of action is based upon situated action theory.

In contrast to Deliberative Agents, Reactive Software agents have no built-in planning or symbolic logic to cope with changes in environmental conditions. Instead, they use a stimulus/reaction effect to dynamically react to changes in their external environment. The smartness of these agents comes from the emergent behaviour of the interaction of the various modules, that is to say, the synergy effect of many low level (reactive) agents working together.

Maes (Maes 1991), highlights three key ideas that underpin reactive agents. These ideas are as follows.

> The dynamics of interaction between these simple unintelligent agents should lead to emergent complexity.
>
> A reactive agent is usually viewed as a collection of modules that operate autonomously and are responsible for specific tasks (e.g. sensing, motor control, computations etc.). Communication between these modules is usually minimal and low-level in nature.
>
> A reactive agent tends to function on representations that are close to raw sensor data, in contrast to high-level representations used by other types of agents.

Such agents generally consist of Actuators and Sensors connected together by a hard-wired reaction path. (Brooks 1991). In this way any change in environmental conditions will result in a discrete change in the agent's actuators, and this change will *always* be the same providing the environmental condition are always identical.

The *benefits* of such agents are they are fault tolerant, simple and easy to understand, flexible and adaptable. Many *reactive agents* are based on a *state machine* basis. The key use of *reactive* agents, at the present moment, seems to be in the entertainment world, where they are used in Video Games, to add an extra element of challenge.

Other uses include modelling of real world situation. For instance, research has been carried out on ant colonies by modelling each ant as a reactive agent programmed to react to certain situations. In this way the real world can be brought into the computer world and simulated using reactive agents to model the behaviour of simple life forms.

Little work has been done on implementing reactive agents theory. The work that has been done, has only served to point out the limitations of this type of agent.

(Maes, 1991) argues that an integration of both reactive and deliberative approaches will deliver the best service to the user. After all, by definition, agents are bounded by their knowledge base, and are therefore limited to the specific area for which they have been designed. These so-called Hybrid agents take the strength of one type of agent and apply it to the weakness of the other. This manifests itself in the form of a somewhat layered architecture whereby there is a *deliberative,* cognitive side that plans out the *strategic* long/middle term plan and a reactive side where the agent responds to stimuli received from its *reactive* agent components. The reactive agents provide the much-needed flexibility at the sharp end, and will respond rapidly to changing situations as dictated by the hard-wired logic between sensors and actuators.

There are few of these *Hybrid* systems in existence at the moment, although the argument for them is overwhelming. The ones that do exist are generally application specific and usually designed in a rather *ad hoc* way. This is certainly an important area for future research, especially for the development a more generic architecture underpinned by AI theory.

The Loner – Cooperative Axis

Co-operation is an important property for an agent. It is what makes them so attractive as a software tool. For co-operation to work properly, some sort of communication language is needed to enable an agent to communicate with other agents and human beings.

The *raison d'être* of collaborative agents is simple - *synergy* - that is the whole is greater than the sum of the parts. Because there is such a need for heterogeneous systems to communicate together, a great deal of research work has gone into collaborative agent development, and the term has become largely synonymous with the generic term agent. This is unfortunate, as the term agent should encompasses so much more. A rather striking example of this misuse of the term can be see from Lotus at a Symposium on agents at MIT in 1992 (Foner 1993). A respected member of the Lotus team presented a version of Lotus 123 that made collaboration between multiple spreadsheet users easier. The whole presentation was trying hard to shoehorn its multiple user software into an "agent-oriented piece", the software was quite clever but the Lotus team were confusing the software's multi-user ability with collaborative agency.

Typically, collaborative agents usually deal with tasks such as:

> interconnecting several legacy systems (For instance; systems that do not directly talk to each other, or are necessarily aware of the other's existence, but who communicate using an agent as a proxy)
>
> enhancing *modularity (*reduced complexity), *speed (* via parallelism), reliability (*redundancy), flexibility* (tasks are composed more readily from modular organisations) and reusability

Within the field of Collaborative agents much work has been done in attempting to find ways of improving the negotiation principles behind inter-agent communication. This has led to agents being attributed with pseudo-emotions and mental attitudes (Shoham 1993). These attitudes encompass agents having *beliefs, desires* and *intentions*. Such, so called, *epistemic* qualities are used as a means of leveraging power in negotiations. The approach is much contested within Artificial Intelligence, with some researchers arguing that agents should have more than these three attributes, and others saying they should have none at all.

The Slave - Autonomy Axis

Autonomy refers to the ability of agents to work on their own without recourse to human intervention. Hence autonomous agents have their own goals that they strive to achieve on the user's behalf. A key element of *autonomy* is the agent's ability to be *proactive* and take the initiative. Agent autonomy can vary from fully autonomous agents that rely very little on the user to slave agents that simply adhere immediately to the users wishes when called upon.

The Rote – Learning Axis

Lastly if agents are to be truly useful in the long term, then the ability to *learn* is important. This is because learning allows an agent to improve its performance over time, by finding the best ways of performing its tasks. For example, there are four ways in which an Interface Agent can learn about the user and domain models and gain *competence*. (Maes 1994).

> They observe and imitates the user's repetitive behaviour
>
> They adapt, based on the user's feedback. (Indirect: e.g. notices when the user ignores the Interface Agent's advice and learns from this e.g. Direct Negative: User says "don't do that again"
>
> They can be trained by the user on the basis of examples (user stipulate rules)
>
> They can ask and obtain advice from other users agents, who may have experienced something this user agent has not.

An agent can use any combination of these. For instance if an email interface agent receives an email and does not know what to do with it, it can automatically refer to its peer agents for advice. The user can *instruct* the Interface Agent to ask for a specific agent X for advice if a problem comes up that it does not know how to deal with (perhaps someone in a similar job who the user thinks can be trusted to make similar decisions to themselves). The most commonly used of these techniques, is *memory based reasoning* (Stanfill & Waltz 1986). Memory Based Reasoning involves the agent comparing the user's current action with a previous list of examples generated by monitoring the user's past actions, and correlating how close its predicted action is to the *actual* action. As a result, a list of confidences can be built up for certain actions, this list being reinforced by comparing it with other agents (its neighbours) for similar actions. If its confidence about a decision is above a certain numerical threshold of confidence (Some weighting metric, say 95 times out of 100 the user has done the same thing), then it does it automatically and informs the user of its actions. This is the 'Do-it' Threshold. Below this is the 'Tell-me' threshold, where the confidence is less, so the agent informs the user of its decision and asks whether it should carry it out, before doing it.

This concept of thresholds allows for the *trust* issue to be satisfactorily dealt with. This is because these agents incorporate, within them, a sliding scale such that if the user has complete *confidence* in its agent they can LOWER the threshold for 'Do-it', conversely if the user has no trust in the agent the "Do-it" threshold is raised to 100%. Prolonged use of the agent should allow the agent to predict actions more accurately, and so it is expected that the 'Do-it' threshold will gradually be lowered over time.

Describing an Agent along the Property Axes

All agents that currently exist will exhibit these properties, to some extent to a greater or lesser degree.

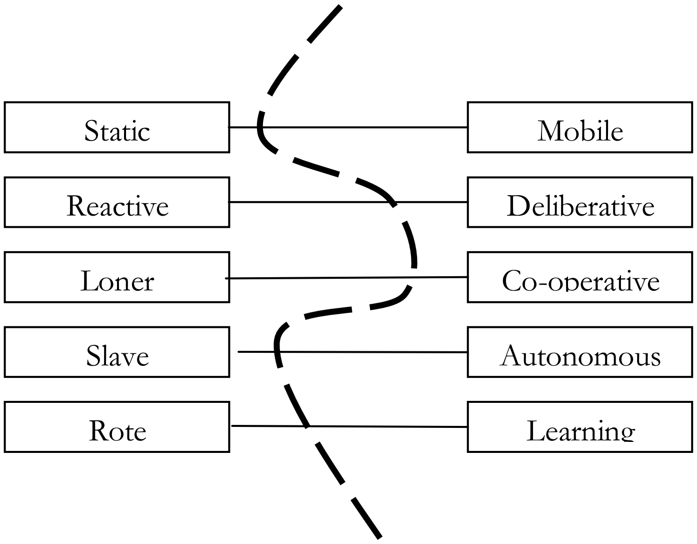


Figure 8 Agent Properties

So, in the example above, the agent is highly *static*, *reactive, loner, autonomous* and with some learning ability. Such an agent might be an information retrieval agent. For example an information retrieval agent is mobile, and reacts to changes in the external information base. However, it may exhibit some learning so as to adapt its state based on user activity. It acts on the users requests as well as pro-actively when it senses the user requires information, and as such is reasonably autonomous.

Critique of Existing Agent Applications

Basis of Critique

In order to provide an understanding of the current state of agent technology a number of agent implementations will be compared on our defined property axes. These axes as previously described are:

Static - Mobile

Reactive - Deliberative

Loner - Cooperative

Slave - Autonomy

Rote – Learning

These orthogonal factors provide a framework from which the different agent implementations are assessed. In addition comment will be made on the following attributes:

Utility

Usability

We have assigned a scale of 0-10 to our axes, where 0 represents the simplest property and 10 the more complex property. These scores are based on the judgment of the author alone.

Interface Agents

Many Interface agents actively assist in the operation of the interface. They offer ways in which the user can cut down on the amount of work they need to achieve a computational task. For instance such agents may help by automating the filtering of email, or by offering short cuts to frequently used highly repetitive user actions.

Example 1: Magic Cap for Windows by General Magic

This agent, by the makers of mobile agent communication language Telescript, is an integrated organiser and communications package. It allows the user to maintain contacts, keep track of appointments and organise personal and professional information. It also lets the user exchange information with other windows applications and sends files as attachments.

Evaluation

Static-Mobile Axis

This agent is completely static. **0/10**

Reactive-Deliberative Axis

Again, it has no deliberative ability, but due to its good interface and comprehensive functionality it often appears more intelligent than it is. **5/10**

Loner – Cooperative Axis

This agent has some cooperative ability in that it can communicate with various other heterogeneous applications, however within our frame of reference it a loner. Although, it does perform some cooperative actions they are not with other agents to achieve a goal, rather they are simple interactions with non-intelligent, non-proactive applications. **3/10**

Slave-Autonomous Axis

The user is firmly in charge here. Although the agent does have some freedom to perform actions without the user's consent, the freedom is set by a user-defined set of actions. Thus the agent is free to sort mail, but only according to instructions set by the user.

This agent is a reactive type agent, as it has no intrinsic deliberative reasoning, rather it reacts to changes in the environment. This can take the form of sorting mail as it arrives, or sending a reminder email if the date/time reaches a certain pre-defined threshold. So within the structure of the agent the user is in control at all times, whether setting appointments or defining the email type to be filtered. **5/10**

Rote-Learning Axis

The agent has limited learning ability. It is adaptable only so far as most aspects of it can be altered or personalised according to user preferences. Therefore it is not dynamically adaptable nor does it possess any inherent intelligence. **2/10**

Utility

This agent is very useful indeed. It provides a link and acts a proxy between several usually distinct applications. Thus when using the appointment manager, the user can set an alarm or get an email to be sent directly from the appointment manager. In addition, if the user sets an appointment with a certain person, Magic Cap automatically provides an email that stipulates the time/date/location of that appointment and sends it to the person required. To add to this, Magic Cap examines any email that arrives and takes the sender's name and email address and automatically stores them in the address book. This address can then be used in the appointment manager, which demonstrates the systems ability to inter-operate between heterogeneous packages.

The email editor is quite powerful and includes the ability to attach files and also allows graphics (of which it has an extensive library) or sound bites to be included in the mail. This agent seems fairly useful. However, it does not seem to quite justify the tag of being an *intelligent* agent (not that this affects its usefulness). After a short while it becomes easy to rely on Magic Cap for setting appointments and sending mail. **8/10**

Usability

This software is extremely easy to use and learn. Upon initial instalment of the software, Magic Cap takes the user through a comprehensive and easily understandable introduction, and uses a Wizard type interface to complete the required set-up. After only a few minutes usage the main characteristics of the agent can be learnt and used.

The interface is powerful and easy to use, it uses as its basis a desktop metaphor, whereby there is a desk with a fax machine (for Faxing), In/Out boxes (for email), Calendar (for Appointment Manager) and a Notebook (for leaving notes). These desktop metaphors are typical of the rest of the package in that they demonstrate good affordance. The on-line help is useful throughout and overall this agent is an impressive piece of software. **8/10**

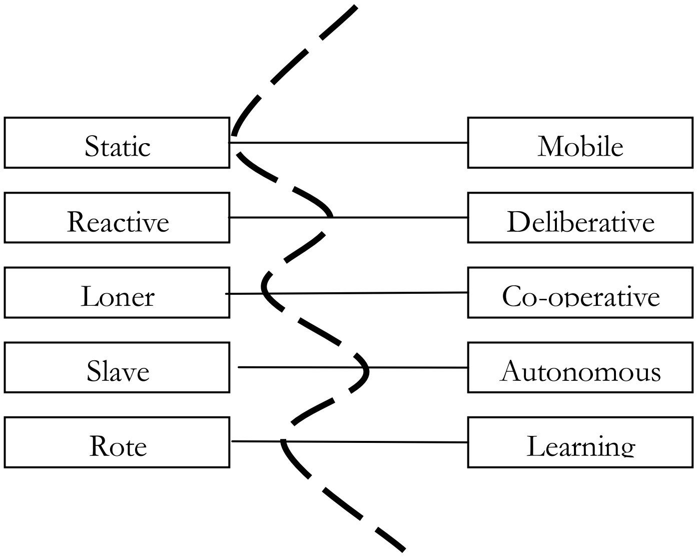


Figure 9 Magic Cap for Windows Adaptation Characteristics

Example 2 Microsoft Agent

This agent's task is to inform/entertain the user. It does this by taking on the anthropomorphic form of a blue Genie. This genie is loaded into the system and lays dormant until activated by the user accessing a Microsoft Agent enabled web site. The genie then appears on screen and interacts with the user using utilising an audio connection (using voice synthesis) and a visual connection (its words appear in the familiar cartoon style bubble). In addition to these features the genie has speech recognition capabilities and therefore allows the user to auralise questions.

Evaluation

Static-Mobile Axis

This agent is completely static. **0/0**

Reactive-Deliberative Axis

The Genie itself has only limited adaptation ability. It often appears more intelligent than it actually is due to the impressive array of features that can be accessed in many ways by web sites. **3/10**

Loner – Cooperative Axis

Limited cooperative ability, the agent needs to cooperate with other genie enabled wed sites to perform its function. However the interaction is very limited and involves no more than simple handshaking, and primitive information exchange. **5/10**

Slave-Autonomous Axis

Although largely controlled by the user, this agent does have some autonomy in that it activates itself when accessing an enabled site. **4/10**

Rote-Learning Axis

It has little significant learning ability. **3/10**

Utility

This agent appears to be used primarily for entertainment purposes, although it does have some rather clever speech recognition and character movement routines incorporated into it. The Genie system does have considerable scope for improvement and promises to be one of the more successful of agents. It is envisaged that the genie will be used as an interactive help/guide agent. The genie will use its natural language processing ability to understand the users' request and then use its knowledge base to offer potential answers to the users' problems. At the present time it has been used to great effect as a guide on selected web sites, in which it can take the user on a pre-defined tour and offer useful information. **8.5/10**

Usability

The Genie Agent system is difficult to use due to vague instructions, and due to low transparency it is unclear whether the agent is running or not. Once installed, however, little more needs to be done, as the Genie appears automatically whenever a web page with Genie capabilities is accessed **8/10**

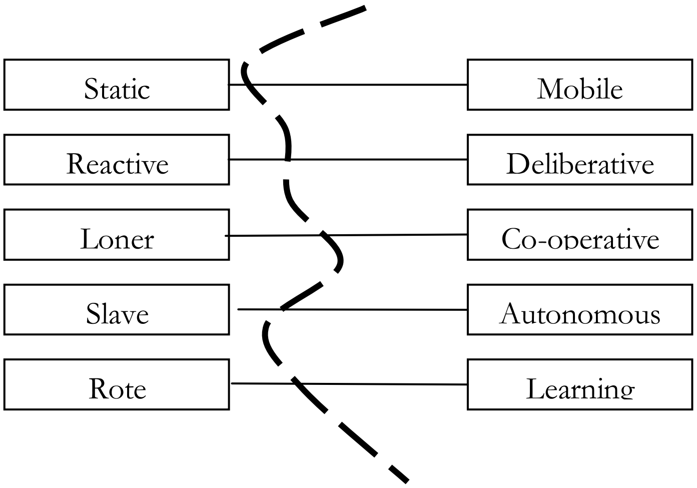


Figure 10 Microsoft Agent Adaptation Characteristics

Example 3: Web Browser Intelligence By IBM

This is a very useful agent and one of the few that demonstrate some learning ability. The Web Browser Intelligence (WBI) by IBM enhances the user's ability to use the World Wide Web by acting as a personal assistant. This agent is typical of the new breed of client based agents, which operate away from a centralised server and allow a much more direct contact with the user. It has the following functionality:

It remembers everywhere the user has been.

It allows the user to search through information that they have seen

It notices patterns in the user's web browsing

It watch's the user's favourite web pages for changes

It tests the speed of the links from pages to let the user know if they are fast or slow

It provides the user with both proxy and socks connectivity independent of the browser

It makes the user "Browser Independent" so they can switch browsers and not lose information

It lets the user look back in web time to see how the user has visited pages in the past

It improves productivity using the web.

Evaluation

Static-Mobile Axis

This agent is completely static. **0/10**

Reactive-Deliberative Axis

This agent has a reasonable deliberative faculty although it does seem rather difficult to get it to suggest shortcuts. This could be a problem due to the often stochastically nature of web surfing. **6/10**

Loner – Cooperative Axis

The agent demonstrates some degree of cooperation, although in essence it is a stand alone, loner type agent. It demonstrates cooperative ability in terms of utilizing heterogeneous applications, and in its ability to proactively monitor and assess external web sites. **3/10**

Slave-Autonomous Axis

The locus of control with this application lies in the application domain, because the agent supervises the uses actions and does tasks for them without their consent or knowledge. **9/10**

Rote-Learning Axis

It is one of the few agents that demonstrate any form of adaptability. This ability stems from its almost unique stance of having some intrinsic learning ability. To do this, it observes the patterns of behaviour that emerge as the user surfs the web. It uses these patterns to note which sites the user visits frequently and what sites they have just come from. By doing this it can then offer shortcuts to these sites, and inform users of how long it should take to accesses them. It does this by showing a small icon next to the link, which can be one of three colours Red, Yellow or Green. Where Red is the longest delay and Green is the shortest. **7/10**

Utility

Excellent application, it really does what it sets out to do and certainly does make using the web a far more pleasant experience than it currently is. Its ability to learn also makes it very valuable as it notes which pages are visited most often and offers short cuts. It also highlights links that may be slow, so that the user can chose alternative browsing paths. One of the most useful agents around. **9/10**

Usability

The system is easy to install since it configures the web browser settings automatically, and this takes away a lot of fairly complex setting up procedures. It takes away much arduous boot strapping required of agents of this type. This capability is provided through its learning ability, and therefore does not require the user to enter pages of forms stating their favourite web sites. It also does not require the user to save pages manually. It merely advises and allows users to browse off-line or choose short cuts to pages it has recommended. The users' main task, therefore, is to directly request WBI to watch certain web page and notify them of any changes. **8/10**

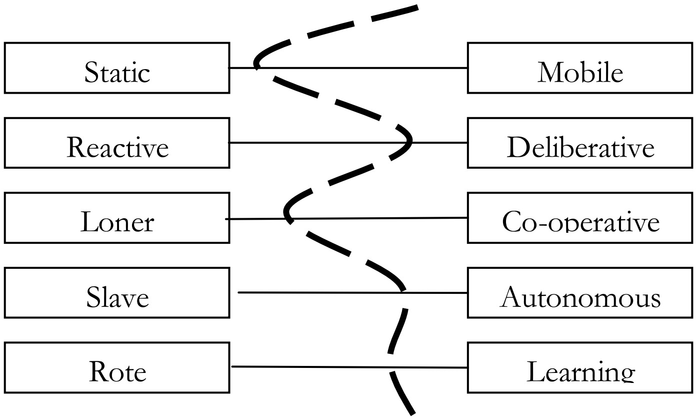


Figure 11 Web Browser Intelligence Adaptation Characteristics

Example 4: Intel Pattern Recognition Agent

The Intel Selection Recognition Agent is an experimental software application that dynamically generates hyperlinks between information on the user's desktop, relevant applications and WWW sites. When text is copied to the clipboard, the Selection Recognition Agent attempts to recognise objects such as email addresses, URL's or keywords in the text. As a result, an icon on the desktop indicates the type of object recognised. A right mouse-click on the icon displays a menu of possible operations. Users can launch a browser with a specified URL, look up a definition of a word, send email, or find information about a geographic location.

Evaluation

Static-Mobile Axis

This agent is completely static. **0/10**

Reactive-Deliberative Axis

This agent is fundamentally reactive, although it has some reasoning with regards to its ability to pick out key functions the user is accessing. **6/10**

Slave-Autonomous Axis

The locus of control for this agent, as in the above example, lies in the domain of the agent. As users perform tasks the agent monitors actions, and suggests ways to improve their working: **8/10**

Loner – Cooperative Axis

As in previous agents this is a stand-alone application that interacts with other non-agent applications. Therefore, it demonstrates some primitive cooperative ability but not in the pure intelligent agent sense. **3/10**

Rote-Learning Axis

The interface has some degree of adaptability and intelligence. It can, for instance, differentiate between the names of major cities and verbs. It therefore recognises, to some degree, the context and meaning of some of the highlighted text. **6/10**

Utility

This agent is quite useful since it attempts to integrate many of the applications that reside on the desktop and offers some sort of continuity between them It therefore makes the process of obtaining information from the web a great deal easier than is normal. An example interaction might be a user writing a paper in *Word* and wishing to obtain some information on New York City. Upon highlighting New York in their *Word* document, the agent has the ability to instantly spawn a web browser containing information regarding New York. Since the agent is also linked up to address books, date tools and email managers it can launch all of these applications if necessary. So if the user sees an email address in a hypertext document they can highlight it and the recognition agent will offer the user the opportunity to write it to their email application's address book or to send an email to that person. It is a powerful application, which has the useful facility of allowing the user to access several heterogeneous applications though one single interface, and as a consequence it has many benefits over current systems. **8/10**

Usability

When installing this agent the system requires the user to enter the file location of their desktop address book, email application, calendar manager and web browser. This information is often not known to the user. Therefore, this makes it difficult for the user to install the agent correctly. In many other applications the agent locates such tools automatically, and offer the user the option of choosing which tools they require. Therefore the installation procedure is slightly more difficult that perhaps it could be. As for the actual interface itself, it is reasonably good. It has easy to understand icons and it uses audio cues to inform the user that the information selected has been recognised by the agent. **8/10**

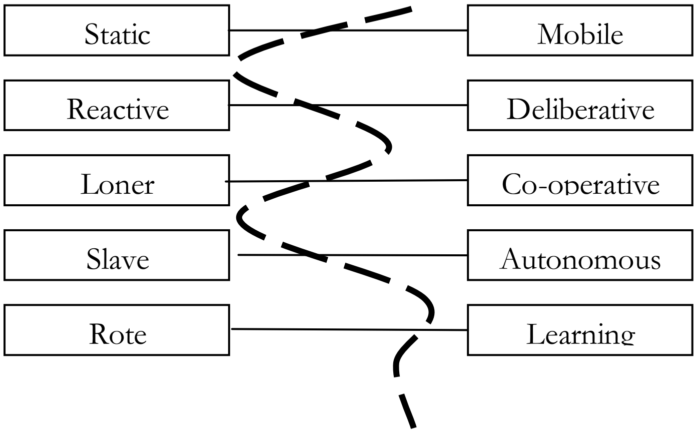


Figure 12 Intel Pattern Recognition Agent Adaptation Characteristics

Information Agents

Some Information agents, typically, roam over the Internet to access various databases attempting to collect data that the user has requested. Thus they are typically shopping type agents, where the agent will attempt to find the cheapest product at a variety of on-line shops.

Example 1: LiveAgent by AgentSoft

This agent development platform allows users to create their own agent. The agents created by LiveAgent simply follow instructions dictated to them by the user, and the user whilst browsing sets their path of actions. As users visits various sites they can 'save' certain sites or html pages. The LiveAgent records the URL's for these pages, and then, at a later date, can be set to retrieve updates and return them to the user. For example, personalised newspaper of information can be gathered by the agent and returned for the user to peruse, off-line if necessary.

This type of agent is the archetypal 'Mobile' agent in that the user does not have to be connected for the agent to work, or indeed need to be online. The agent retrieves the pages independently after visiting each site.

Evaluation

Mobile – Static Axis

The agent is completely static. **0/10**

Slave – Autonomy Axis

The agent does not interfere with the user's primary goal, and thus the locus of control is firmly defined by the user. In fact the agent does nothing more than the user requires and stipulates. In this sense the agent has no control because the user defines the sites visited and what pages are to be stored for retrieval. However once the user has stipulated their requests the agent acts wholly autonomously and performs the user's secondary goals of retrieving the pages without user assistance. **4/10**

Reactive – Deliberative Axis

Although the agent is undoubtedly useful, it exhibits no signs of having any reasoning ability. It merely follows the user instructions to the letter and has no inherent ability to reason about its own actions. **1/10**

Loner – Cooperative Axis

Shows very little cooperative ability, this is a loner type agent. **2/10**

Rote – Learning Axis

This agent shows little sign of adaptability. It has the ability to enter form data on html pages. Thus if the agent accesses a password entry form, it can enter the user defined password. **3/10**

Utility.

The agent does perform several important timesaving functions. Once the initial difficulty in using the software has been mastered it becomes a powerful tool for obtaining information. For instance, if users require the latest news in telecommunications, they can define an agent that will visit several on-line daily telecommunications magazines and return the latest news sections in each. Thus the user can develop several agents, each of which is specialised in obtaining certain types of information. The user may have a Sports Information Agent or a Financial Information Agent etc. The most practical example of this is the Search Agent that is provided with the software. The user provides it with a search string and the agent then visits several on-line search engines looking for information related to that string. Upon gathering all the results from several databases it eliminates all the redundant and common information and returns the filtered results to the user. Finally, this agent can save a great deal of money for modem users, as it can be set to download and retrieve information during the night, or whenever the cheap rate occurs. **7/10**

Usability

The software is quite tedious to install and get working correctly, and at the time of review is a beta release and thus had several bugs. These bugs rapidly became infuriating and limited the extent to which the agent could be tested. For instance, several problems were encountered when trying to record certain web pages, which resulted in the software crashing and losing a great deal of work done up to that point.

Another problem encountered was the inability of the agent to recognise user-defined bookmarks within Netscape. This means that the user cannot surf by selecting their normal bookmarks. Instead they are forced to type in URL's manually, which rapidly becomes very tedious and time consuming.

The actual software itself was reasonably easy to learn and use, and the interface was simple with well-defined icons. **6/10**

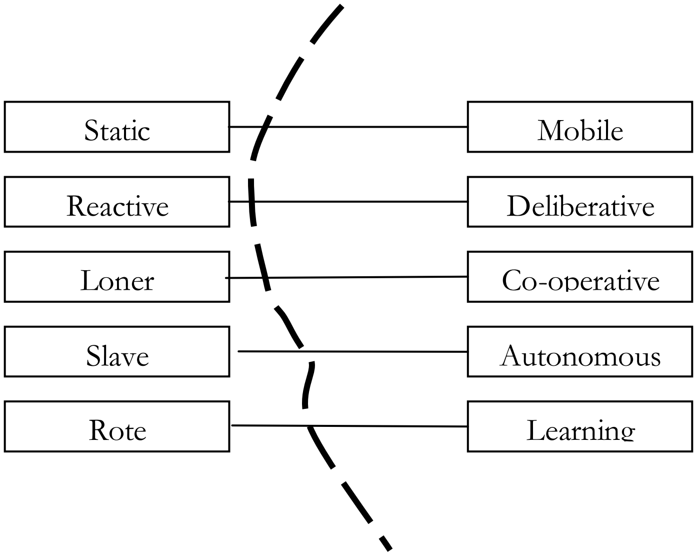


Figure 13 LiveAgent Adaptation Characteristics

Example 2: Waldo The Web Wizard.

This is a service which attempts to interest the user in a range of sites they may not have previously visited. It does this by asking the user several questions about their lifestyle, the type of car they drive, their pastimes, their news interests etc. From these questions the Web Wizard offers a list of potentially useful sites catered for user tastes. Not only does it offer sites of interest, but it also offers commercial sites where users can *purchase* items that may interest them.

Evaluatation

Mobile – Static Axis

This agent is completely static **0/10**

Slave – Autonomous Axis

In this case the user is firmly in control. The agent does not attempt to exercise any control or steer proceedings. The user is presented with a series of questions, and possible answers which are represented as a series of graphical representations of situations or items. The user selects the graphical representation they feel most appropriately answers the question. This is all the input the user is required to enter. From these answers the agent provides a set of recommendations based on the questions. **3/10**

Reactive – Deliberative Axis.

There is no significant demonstration of reasoning ability. It did, however, have an interesting method of assessing the accuracy of the answers given. It produced a summary of its interpretation of the user's preferences, by whicb the user can assess the systems accuracy. The results of this user-based assessment is used to refine its answers **4/10**

Loner – Cooperative Axis

Once again a stand alone client application, that does demonstrate some cooperative ability by interacting with a central server which connects and makes assumptions about a users browsing patters by comparing them with others. Therefore demonstrates some cooperative ability 4/10.

Rote - Learning Axis.

This agent showed no sign of learning, it appeared to utilise a primitive algorithmically based method of selecting web sites based on the user's input. **1/10**

Utility

The mechanism by which recommendations are made is basic. It does not seem to have a multiple comparison feature, instead it attempts to directly match answers given to appropriate web sites without reference to previous answers. Therefore, the web sites recommended are often a set of predicatable sites that could easily have been discovered using traditonal search methods with far less investment. **3/10**

Usability

The interface for this agent is fairly usable. It uses an anthropomorphic image of the agent to interact with the user. Although in many cases this type of interface can lead to overhead at the interface, here it worked reasonably well. The system displays multiple choice answers to the questions in the form of cartoon pictures. This method allows more ambigous answer to questions to be presented, as each image can be interpreted in a variety of ways.

However, the way the agent gathered its results was discovered to be unsatisfactory. It selected only one choice from several possible options, therefore rejecting two or three equally valid choices. For example , a typical question the system asks is "which part of a newspaper do you most enjoy?". This question may have several possible answers, for instance a user may enjoy the sports section, but equally they may also like the science and technology sections. By answering the question with a single reply "Sport", the list of suggested sites returned was biased wholly towards sport based sites, therefore ignoring the users other interests in science and technology. Perhaps a system based on order of preference would be a better way of evaluating each user's requirements. **5/10**

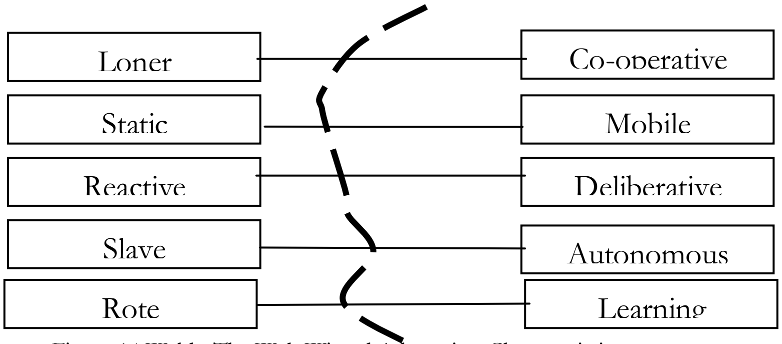


Figure 14 Waldo The Web Wizard Adaptation Characteristics

Example 3: Firefly Network.

The firefly network is the practical, commercial version of some of Maes's work. It predominantly uses memory-based reasoning techniques to achieve a degree of reasoning ability. In addition to these techniques, it uses a much improved version of the *similarity engine* for recommending titles of music and books to users.

It works, as most of these class of agents do, by users answering an extensive set of questions about themselves. From these questions the agent recommends titles of music or film and sites that the user may find interesting. It also attempts to cross-reference these recommendations. For example, the system might use its knowledge of all the movies the user likes to match them to another user with similar tastes. Once it has matched a user, the system answers this is the basis for making recommendations on subjects that were not the basis of the recommendation. So it might make a user match based on user taste in movies, and use this as a basis for recommending music titles. The system claims to make better recommendations the more it is utilised.

Evaluation

Static-Mobile Axis

This agent is completely static **0/10**

Reactive-Deliberative Axis

It demonstrated some degree of intelligence in its ability to utilise learned information and applying it to obtain a context from which data is selected. **7/10**

Slave-Autonomous Axes

This is one of the few systems where the locus of control is firmly biased towards the agent. This is because the user has no specific goal in mind other than requesting recommendations. For the agent to achieve this goal, it must be equipped with as much information as possible. Therefore, from the outset it retains the locus of control by asking a series of questions. These questions form the basis for other suggested web sites based on recommendations from other user's agents of similar tastes. Thus the user is always *responding* to requests as opposed to directing them. **8/10**

Loner – Cooperative Axis

Central server based agent that shows some cooperative abilities, by interacting with other server systems to cross-reference users preference with those of similar preferences. **5/10**

Rote-Learning Axis

This is one of the few interface agents that showed any learning ability. Its adaptability is evident in the way it learns and acts on users preferences over time. **7/10**

Utility

This agent system is not wholly effective, due in part to the nature of its operation. It suffers because its expertise in the application domain is completely based on the premise that it can use other people's tastes as a recommendation system, without regard to context. It suggestions are often nonsensical and seem to bear no relevance to the information the user has entered. In addition, the presumption that the system can use different criteria as the basis for recommendations is a bad one. It does not seem to work.

This system suffers from a lack of incentives, since it only really works when people make *new* suggestions to the database. Otherwise the system stagnates and recommendations are soon out of date with current culture, and thus the effectiveness of the system is stifled. Most users merely want to get recommendations from the agent and *not* to wish to utilise valuable time on making recommendations of their own. This is a problem that needs solving if the system is to truly succeed. However, the system is more sophisticated than other agents of the same type. As a consequence its results are generally of a higher quality. **7/10**

Usability

It is not an easy system to use, it takes a reasonable amount of time to learn how to use and become adroit at handling the interface. It is not immediately obvious how things work, and the on-line help is almost useless. It uses an applet system whereby the user has a Firefly passport, separate from the main browser window, where the user selects their options from a menu. **6/10**

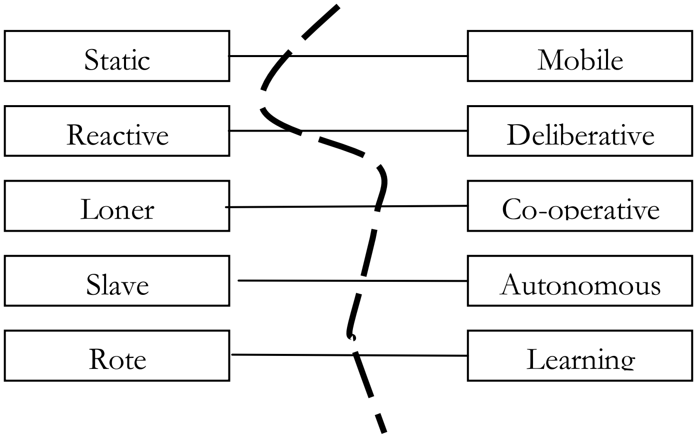


Figure 15 Firefly Adaptation Characteristics

Computational Agents

Example 1: Jini Prolog Engine Service

Jini Prolog Engine Service (JPES) is a Jini (Jini is a Java programming interface that enables embedded devices to utilise JAVA) service that provides remote Prolog engine services to Jini-enabled components in the network. Using the Jini architecture, JPES provides a flexible infrastructure for distributed components to gain access to its Prolog engine. JPES makes AI techniques like rule-based programming, constrained problem solving, knowledge sharing etc. available to the development of the next generation "intelligent" distributed system.

Evaluatation

Mobile – Static Axis

This agent is completely static **0/10**

Reactive – Deliberative Axis.

JPES is a reactive type agent that does not deliberate on its actions. It acts reactively to requests from other agents for it services. **2/10**

Slave – Autonomous Axis

JPES is a fully autonomous agent service that does not directly interact with the user. Rather, it offers services to other agents. It therefore is highly autonomous. **9/10**

Loner – Cooperative Axis

An agent whose purpose is to act cooperatively, it provides services for other Jini enabled agents. **9/10**

Rote - Learning Axis.

Since the agent does not interact with the user, it has nothing to learn from. In fact it operates as a service and therefore operates by rote. **1/10**

Utility

JPES provides a very useful service to JINI enabled devices. It enables these devices to access intelligent services and utilise its PROLOG engine to return computationally demanding and logical tasks. **8/10**

Usability
Not applicable, this agent does not directly interact with a user. The agent programmer interaces are quite straightforward though **3/10**

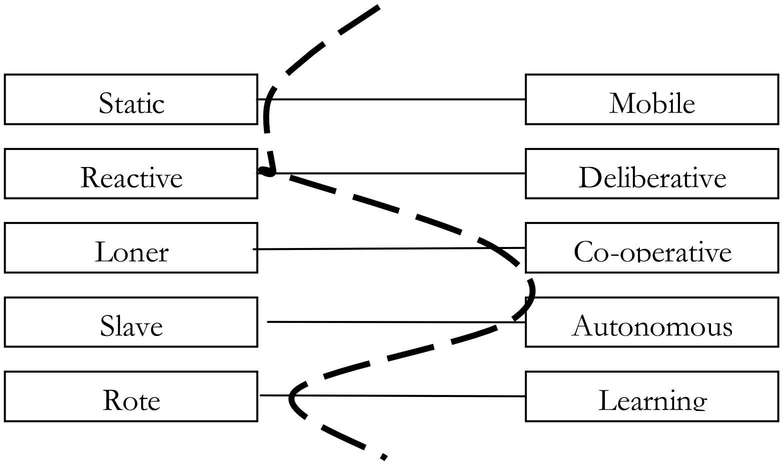


Figure 16 Jini Adaptation Characteristics

Facilitator Agent

Example 1: Ozro Negotiate

Ozro Negotiate is the negotiation engine at the heart of the Ozro Agreements application suite. This technology is designed to facilitate people-centric, iterative, and multi-attribute negotiation, providing services by prompting, capturing and synchronizing communications.

Evaluation

Mobile – Static Axis
This agent is completely static **0/10**
Reactive – Deliberative Axis.
Ozro is a reactive type agent that does not deliberate on its actions. It acts reactively to requests from the user for its services. **2/10**
Slave – Autonomous Axis
Ozro provides services for the user by finding other appropriate agents that can fulfil the users requirements. **7/10**
Loner – Cooperative Axis
A fully cooperative agent whose role is to facilitate communication between other agents, that acts as a hub making it fully cooperative. **9/10**

Rote - Learning Axis.
Ozro operates as a service and does not learn. **0/10**
Utility
Ozro provides a useful service, facilitating the discovery and negotiation with other distributed agents. Its expertise comes from its knowledge of other agents abilities and the services they offer. It is a useful agent in the e-commerce sector. **8/10**
Usability
The user does not interact directly with the agent, rather they issue a request. **3/10**

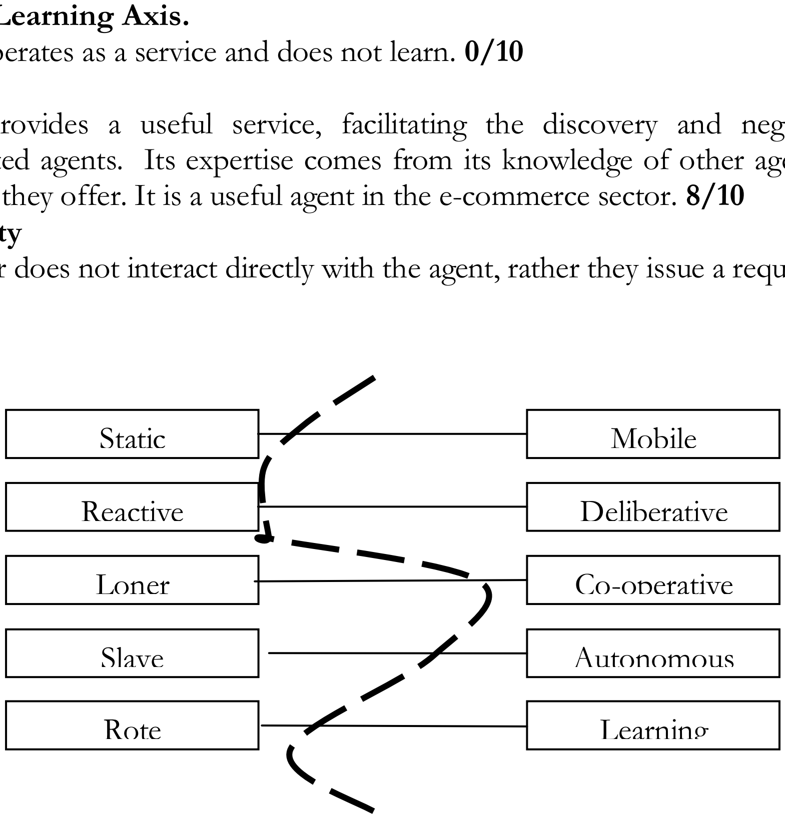


Figure 17 Ozro Negotiate Adaptation Characteristics

Special Issues relating to Interface Agents - Competence, Trust and Limitations of Autonomy

Interface agents have special requirements and responsibilities because they interface directly with human beings. Firstly, there is the issue of TRUST. Will a human being be sufficiently confident about the performance of an agent to trust the agent to provide the right service? Secondly, there is a related issue of COMPETENCE. Can the human user have confidence that the agent is competent to carry out the service. Finally, since the human being and the agent are co-operating on solving a task, there is the important question of how much delegation or autonomy is acceptable. This latter point is also connected with the PURPOSE of an agent.

Competence and Trust.

Maes (Maes 1994) has suggested the following questions in relation to competence and trust:

> How does an agent acquire the knowledge it needs to decide when to help the user?
>
> How does it decide WHAT to help the user with?
>
> Finally HOW does it help the user?
>
> How confident can the user feel with the agent, when delegating tasks to it?

The ways in which an Interface agent can be implemented affect the levels of competence and trust attained. To illustrate the point three implementation methods will be examined – End-User Derived, Knowledge Based and Learning Agents.

Method 1: End User Derived

This method involves the end user defining a set of rules to the agent (say for dealing with email). The agent then goes about automatically performs these tasks without explicitly confirming its intentions to the user (Lai, et al 1988).

This method is often effective for very simple tasks, but is highly inflexible. Any software using this method does not really deserve to be termed an *agent* as it amounts to no more than a simple script for dealing with lists. When assessing this approach, it is clear that this method has poor *competence* due to the large amount of work the end user and developer have to put in. This is because the users have to examine their activities, assess their own habits and formulate strategies to deal with those habits. The *trust* level, however, is high because the user has stipulated exactly what the agent should be doing and the agent has no facility to do anything outside of these user defined boundaries and thus cannot break them. A better technique would be to use a hybrid approach whereby the mainline goal of the user is performed using this method, but a learning based interface is also incorporated.

Method 2: Knowledge Based Approach

Another approach is the *Knowledge based Approach* (Sullivan 1991). Here the agent is given knowledge about the application domain, and the user, at the outset. The agent then may have the ability to recognise the user's problems and either correct them when wrong or provide advice.

This method has problems with *competence* and *trust*. From a *competence* point of view, this method is very inflexible as it can only be used in the specific domain it was designed for due to the heavy specialised expertise needed (i.e. A UNIX help agent would be useless for EMAIL). Lastly, the approach is highly inflexible, as there is no facility for learning from the user. From a *trust* point of view, it is rather unnerving for a user to be presented with an agent that knows everything from the start. Schneiderman (Maes & Schneideman 1997) argues that this can leave the user bewildered with a feeling of loss of control.

Method 3 Learning Based Interface

This method was developed by Maes (Maes & Kozierok 1993) and is a learning based interface agent. Here the agent starts with a minimum of knowledge in either the user or application domain but has the capacity to *learn*. Therefore for this agent to work successfully two criteria must met

> that the user application must involve repetitive behaviour of some sort, to a substantial degree (otherwise it will never learn)
>
> that other users have *different* repetitive behaviours (Otherwise these learning based agents offer no advantages over knowledge based agents)

This approach is termed a *personal assistant* metaphor. The agent starts knowing nothing but learns over time. This approach offers a solution to the problem of *trust* since the user has time to get to know the workings of the agent, as well as the agent getting to know the working of the user. In addition this method allows the agent to present explanations for its decisions based on past examples (i.e. "I did this because you liked it like this last time").

The Purpose of an Agent and the Issue of Delegation

Many existing definitions of agents involve the *functionality* of an object. However the *purpose* of an object is an important quality, since within the scope of that agent's purpose the *functionality* can be fulfilled in whatever way the designers see fit. For instance, the *purpose* of a window is, *"to allow humans to see through it".* Within that scope many window builders might construct windows with this *functionality* yet allows the windows to be tinted, or etched, as long as they still fulfil their defined purpose.

Any definition of an object can be composed of two parts; firstly a summary of the object's *attributes* and secondly a description of the object's *purpose.* Thus a *shopping window* object can be described firstly in terms of its *attributes*:

"A transparent piece of glass (that it has the attribute of being composed of glass, and has the attribute of being transparent)"

and its *purpose:*

"*It allows humans to see goods through it and protect the goods from different environments and theft"*

To give a full definition of:

"An transparent piece of glass that allows human beings to see goods through it and protect the goods from different environments and theft"

One problem with many agent definitions is that they tend to be biased by the author's viewpoint. Thus they often consist of the author's view of what properties an agent should demonstrate, and *not* what the *purpose* of the agent is.

So to cite some examples given earlier:

> *Nwana (*Nwana 1996*): " A component/software object/hardware which is capable of acting exactingly to accomplish tasks on behalf of its user."*

Here Nwana gives us a very nebulous definition, such that the attributes are vaguely described as

> *"A component/software object/hardware",* where the functionality is described as "*..accomplishing a task on behalf of its user.* "

This is not encompassing enough. What kind of task? What is the purpose of the agent? We know it is "*capable of...accomplishing tasks on behalf of its user"* but is that all it does? This definition leaves a lot unexplained, and inadequately describes the purpose.

> *(*Hayes-Roth 1995):*"Intelligent agents continuously perform three functions: perception of dynamic conditions in the environment; action to affect conditions in the environment; and reasoning to interpret perceptions, solve problems, draw inferences and determine actions."*

This is a definition, which does not describe *at all* the *properties* of an agent. Instead it provides a list of functions an agent should perform, with no idea of *why* agents exists, or of what their *purpose* is.

Even this definition's minimal attempt at trying (and failing) to define the purpose of an agent :" *to interpret perceptions, solve problems, draw inferences and determine actions",* is just describing a basic agent's functionality.

Locus of Control and Agent Purpose.

It is self-evident that an agent should carry out tasks on behalf of the user. It is also clear that an agent should not overstep its authority. The basic problem is to define how much authority should be delegated to an agent so that it can work efficiently on the user's behalf, without causing unintended damage. The problem centres round the concept of *locus of control.* Should control be wholly in the user domain (that is the agent is strictly user controlled, with minimum autonomy or decision making capability), or wholly in the agent domain (the agent is able to make every decision without user consent and has the ability to act as a user at every instance, with full user accountability and power). To better understand how the locus of control should be set it is necessary to examine the relationship between user and agent, and the *purpose* of an agent.

What is the *Purpose* of Agents

It is hypothesised here that agents exist only to provide services to human beings, carrying out support applications on computer systems in order to allow the human beings to improve their efficiency and effectiveness at particular tasks. The agents exist to take over some of the technology-based problems encountered in everyday work away from the user. However, their purpose must also encapsulate some of the degree of control, so that they are helpful without becoming troublesome or irrelevant. The initial definition of purpose – that an agent should perform a goal on behalf of the user, gives too much responsibility to the agent, and could take away control from the user. For instance, if the user's goal were to edit a word document, the user would not want an agent to cut and paste without any user intervention. Even with the best of intentions, some control would be needed.

Thus it is argued here that a better definition of purpose should be:

"To perform user defined sub-goals within the boundaries set by a mutually acceptable service level agreement".

These are usually the tedious and technology dependent tasks, which need to be done, but only the answer is relevant. The actual method employed is not important as far as the user is concerned. For instance, a user's mainline goal may be to *receive email.* An agent could perform all the underlying sub-goals such as decoding certain mails or unzipping certain files etc. The detail of how these sub-goals are achieved is not important to the overall user mainline goal. The User might want all mail filtered from a certain source. However, the way in which the filtering is achieved is important and can affect the result. In this case the user needs to be involved and it is unlikely to be completely delegated.

So the agent must carry out the sub-goal of the user in such a way it does not affect the user's primary goal. There are many ways of inadvertently and adversely affecting the user's primary goals, and so the agent must have a clear idea of what the user's primary goal is, and to what extent it can carry out its tasks.

Dealing With The Locus Of Control

At the *initialisation phase* of the agent, the locus must lie firmly in the user domain. Upon *negotiation* of the required limits of action between the user and agent it is likely that there will be an overall user oriented locus of control. However, when the primary sub-goal has been defined, the locus of control moves over to the agent domain, at which point the agent is free to perform its tasks, without user interference. This approach hinges upon the establishment of an appropriate limit of service and prime goal definition within a service level agreement.

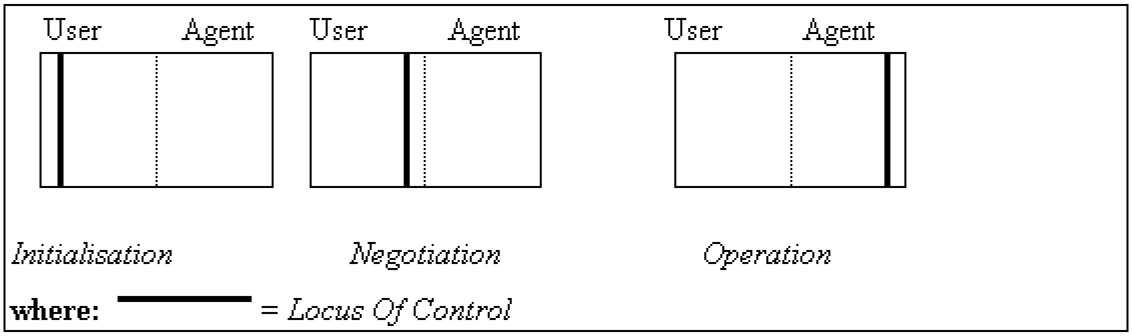


Figure 18 Diagram Showing The Locus Of Control At The Various Stages of Agent Operation

A New Agent Definition:

We can attempt an initial definition of agency utilising the principles stated previously and the preceding sections on agent properties and purpose.

"*An agent is a piece of software that to some degree demonstrates the following attributes; Autonomy, Reasoning, Mobility, Co-Operation and Learning AND whose primary goal is to fulfil, in the best way possible the sub-goals of the calling agent (who may be a human agent), defined in by a mutually agreeable service level agreement*"

It can be argued that this definition does not include those agents, which only ever fulfil the requirements of ANOTHER agent. It can be argued that it does because ultimately agents answer to human beings, so however far down a chain of inter-linked agents, each will only be fulfilling the sub-goals of the agent directly above. As a consequence each agent contacted might only be fulfilling a very small part of the user' sub-goals. There are no agents in existence, as far as the author is aware, that have constructed themselves for their own benefit. All have been created by humans with a purpose in mind. That purpose is the fulfilling of their sub goals.

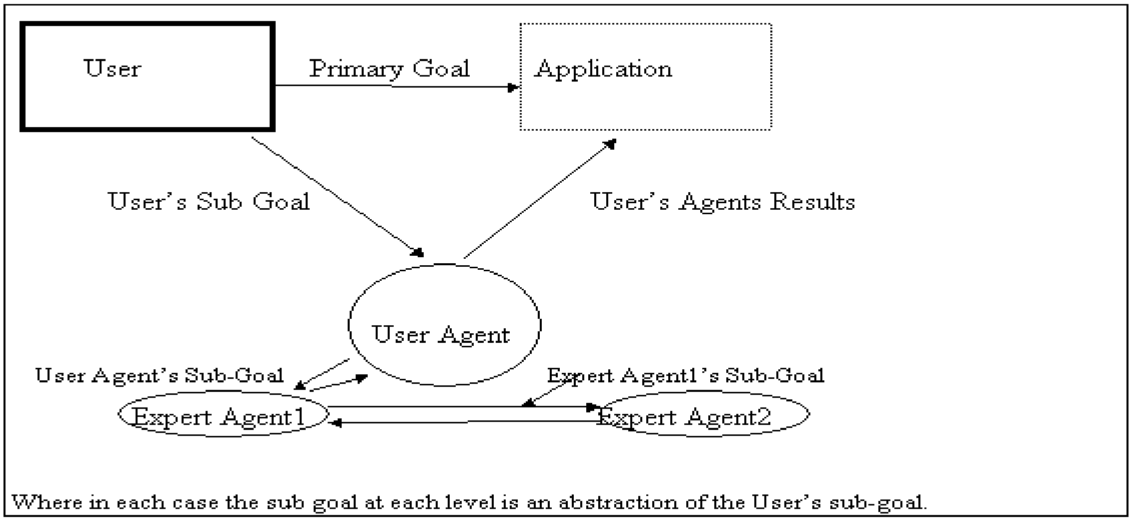


Figure 19 Use Of Service Level Agreements

We can see this idea in Figure 19, where a user is utilising an agent to execute a sub-goal that is needed by an application. In this case, the primary goal may be finding the email address of a colleague and pasting it into the application. Thus the sub-goal is the process of obtaining the required email address by searching various expert databases. As an abstraction of the initial user's sub-goal, a service level agreement is negotiated which requires the agent to provide the Email address of a colleague. This limits the autonomy. The user-agent then requests the help of Expert Agent1 to fulfil this sub- goal of finding the email address. However, Expert Agent1 finds it does not have all the knowledge necessary to fulfil the request, so it requests the help of Expert Agent2. Expert Agent1 negotiates an SLA with Expert Agent2, which might further restrict the sub-goal. Expert Agent2 then attempts to fulfil Expert Agent1's sub-goal. When completed the results ripple back up and are used by the *user agent* to return the results to the application and therefore fulfils the user's *primary goal.* All the agents involved above, use an abstraction of the user 's sub-goals as a primer from which to fulfil their obligations to their peer agent.

Conclusion

Agents are essentially intelligent objects that have the ability to proactively operate and react within an environment. They provide means of taking data abstraction within an object a step further, and provide means of encapsulating an abstract knowledgebase. Agents provide a good solution to complex problems/systems (such as process control) because of their modular nature, complex problems need to be broken down into smaller problems, and individual agents can represents each of the smaller broken down parts of the problem or system.

Agents have been used in many different application domains such as business management, manufacturing and service-based applications. Within these domains different types of agents exist such an *interface* and *information* agents. Interface agents attempt to move the interface paradigm away from direct manipulation to indirect management. For indirect management to be accepted by a user the key issues of *trust* and *competence* must be addressed.

Several fundamental properties of agents can be derived such as: static/mobile, reactive/deliberative, loner/cooperative, slave/autonomous and rote/learning. These properties can be defined by the context in which the agent must operate, and the *purpose* of the agent. Generally, speaking an agent's purpose is to fulfil the sub-goals of a user.

The agent system discussed in this thesis obeys the definition of agent purpose. The operator's primary goal is to operate the process as efficiently as possible with minimum number of errors. The adaptive system performs the sub-goal of ensuring the interface is as well configured as possible to allow the operator to perform the primary goal, and concentrate on operating the system without having to perform unnecessary and superfluous interactions.

An implicit part of the agent definition is the use of mutually acceptable service contracts that define the agent's limits of action. These contracts are of prime importance for open systems, but since the process control domain is closed loop and is not instantiated every time the operator uses it, the contract need only be defined once. This definition occurs at design time by system designers who configure the adaptive system. They define the extent of the adaptation and the limits of powers the agents have to configure the interface. There is no pre-defined contract, as a contracts prime use is within an open system, rather the contract is implicit within the system configuration.

This chapter has shown why agents are an ideal paradigm for adaptive interface construction in the process control domain. The agent system used utilises a mix of knowledge-based interface and information agents, and will operate on a direct manipulation interface, but adapt it using indirect management principles. It has reviewed existing agent applications, to learn what aspects of agents work well and which do not. The results of these lessons are applied in the architecture. A new definition of what agents are was defined. This definition was used to clarify the role the agents play within the adaptive system. The issue of locus of control was addressed and an initial solution was arrived at. Within our closed loop system the definition of control is implicitly captured within the design time configurations of the system.

The next chapter builds on the issues reviewed in this chapter and examines the area of multi-agent systems.

Chapter 5

MULTI-AGENT SYSTEMS AND PROCESS CONTROL

Why use Multi-Agent Systems in process control?

The previous chapter described the use and function of agents, and multi-agent systems, however it did not mention why agents lend themselves for use in the process control domain. This chapter will attempt to describe the rationale for this choice.

There are several defining factors (Parunak 1999) that make agents particularly suitable for this type of application.

Modularity

Agents are pro-active objects, and share the benefits of modularity that have led to the widespread adoption of object technology. They are best suited to applications that fall into natural modules. An agent has its own set of state variables, distinct from those of the environment. Some subset of the agent's state variables is coupled to some subset of the environment's state variables to provide input and output.

The process control domain, makes a good candidate for agent-hood since it has a well-defined set of state variables that are distinct from those of its environment, and its interfaces with that environment can be clearly identified. The state-based view of the distinction between an agent and its environment helps us understand why functional decompositions are less well suited to agent-based systems than are physical decompositions. Functional decompositions tend to share many state variables across different functions. Separate agents must share many state variables, leading to problems of consistency and unintended interaction. A physical decomposition naturally defines distinct sets of state variables that can be managed efficiently by individual agents with limited interactions.

Decentralised

An agent is more than an object; it is a pro-active object, a bounded process. It does not need to be invoked externally, but autonomously monitors its own environment and takes action as it deems appropriate. This characteristic of agents makes them particularly suited for applications that can be decomposed into stand-alone processes, each capable of doing useful things without continuous direction by some other process.

Changeable

Agents are well suited to modular problems because they are objects. They are well suited to decentralized problems because they are pro-active objects. These two characteristics combine to make them especially valuable when a problem is likely to change frequently as would be the case in a disturbed state of process control. Modularity permits the system to be modified one piece at a time. Decentralization minimizes the impact that changing one module has on the behaviour of other modules. Modularisation alone is not sufficient to permit frequent changes. As Figure 20 suggests, in a system with a single thread of control, changes to a single module can cause later modules, those it invokes, to malfunction. Decentralization decouples the individual modules from one another, so that errors in one module impact only those modules that interact with it, leaving the rest of the system unaffected.

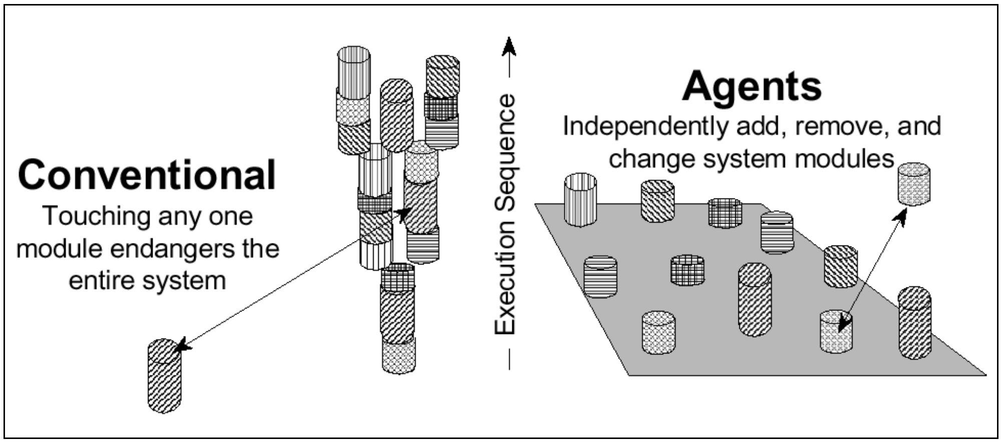

Conventional
Touching any one module endangers the entire system
Execution Sequence
Agents
Independently add, remove, and change system modules

Figure 20 Modularity + Decentralisation = Changeability

Ill-structured

In traditional systems, an architecture of the application is often produced that shows which entities interact with which other entities, and specifying the interfaces among them. Recently these have been in the UML format. However, sometimes, determining this information in advance is extremely difficult or even impossible. From a traditional point of view, an application is ill structured when not all the specifications can be provided at design time. Such a situation is a natural one for the application of agent technology. The fundamental distinction in an agent's view of the world is between "self" and "environment." "Self" is known and predictable, while "environment" can change on its own within limits. Other agents are part of this dynamic, changing environment. Depending on the complexity of individual agents, they may or may not model one another explicitly. Instead of specifying the individual entities to be interconnected and their interfaces with one another, an agent-based design need identify only the *classes* of entities in the system and their impact on the environment. Because each agent is designed to interact with the environment rather than with specific other agents, it can interact appropriately with any other agent that modifies the environment within the range of variation with which other agents are prepared to deal.

Some applications are intrinsically under-specified and are thus ill structured, and agents offer the only realistic approach to managing them. Even where more detailed structural information is available, the wiser course may be to pretend that it is not. A system that is designed to a specific domain structure will require modification if that structure changes. Agent technology permits the analyst to design a system to the classes that generate a given domain structure rather than to that structure itself, thus extending the useful life of the resulting system and reducing the cost of maintenance and reconfiguration.

Complexity

One measure of the complexity of a system is the number of different behaviours it must exhibit. Typically, the number of different interactions among a set of elements increases much faster than does the number of elements in the set. By mapping individual agents to the interacting elements, agent architectures can replace explicit coding of this large set of interactions with generation of them at run-time.

Just as well-structured systems can become ill structured when viewed over their entire life span, so a system that appears to require only a few behaviours can become more complex as it is modified in response to changing user requirements. By adopting an agent approach at the outset, it is possible to provide a much more robust and adaptable solution.

Implementation Issues

According to (Gasser & Bond 1998), in order to obtain coherent system behaviour, individual agents in a multi-agent system should not only be able to share knowledge about the problems and solutions, but should also reason about the processes of coordination among other agents. (Hewitt 1986) claims that in a multi-agent system there is no possibility of global control, globally consistent knowledge, global success criterion, or even a global representation of the system, so the task of coordination can be quite difficult. The inherent difficulties encountered in implementing coordinated behaviour in any multi-agent system as identified by Gasser in are as follows (Gasser 1991).

1. Communication: How to enable agents to communicate? What communication protocols to use?
2. Interaction: What language the agents should use to interact with each other and combine their efforts?
3. Coherence and Coordination: How to ensure that the agent's coordinate with each other to bring about a coherent solution to the problem they are trying to solve?

Aside from these inherent implementation difficulties in a Multi-agent System, there are also practical issues of assuring that pre-existing (legacy) applications can be integrated into agent-based applications and used in agent communities. Thus, specific attention should be paid to this whilst choosing a tool for multi-agent application development. The choice of a proper tool can arm the developer with many advantages, while being careless about it can prove to be constraining in the long run. Chapter 7 provides an in-depth, detailed discussion of agent and system building tools.

Communication

Communication enables the agents in a multi-agent system to exchange information on the basis of which they coordinate their actions and cooperate with each other. This raises the important question of what communication protocols and mechanisms are conducive to enhance collaboration between communicating agents. In a multi-agent system several ways have been proposed for agents to exchange information with each other. Agents can directly exchange messages, or they can organize themselves into a federated system and communicate through special facilitator agents, or they can broadcast the messages.

Another popular approach used to enable agents to intercommunicate is through a shared data repository (called a blackboard) in which information can be posted and retrieved (Chaib-draa et al 1996).

The three approaches can be summarized as follows:

Directed communication.

Directed communication involves establishing direct physical links with other agents using a protocol such as TCP/IP, which promises safe arrival of message packets by implementing end-to-end acknowledgments. The physical link implies that the agent has to be aware of all the other agents in the system. Agent addresses may be obtained either as part of received broadcast messages from other agents or from a centralized object (for example the AgentNameServer in JATLite as explained in Chapter 7) like a directory service where all the agents joining the system register. A sender agent can access the addresses of the receiver agents by looking at this centralized object. Registering is like a start-up process for the directory to learn about all the agents in the system. The FIPA 1997 Specification, V 1.0 (FIPA 97), specifies that any multi-agent system compliant with their specification should have an Agent Directory which contains information about all the agents in a particular environment and facilitates identifying and accessing of agents. A directed communication mechanism is used in most existing agent building languages and platforms. In situations where an agent is engaged in a dialogue with a particular agent, and knows exactly who to send the message to, directed communication makes sense.

Federated Systems.

When the number of agents in a system becomes very large (e.g. in a setting like the Internet) the cost and processing involved in directed communication is prohibitive (Genesereth & Ketchpel 1994). A popular alternative to directed communication that eliminates these difficulties, is to organize the set of agents into federated system. Within a federated system, agents do not directly communicate with each other. Instead, they communicate through special facilitator (mediator) agents. Here a set of agents has a facilitator who is kept informed about their individual needs and abilities. Agents can also send application-level information and requests to these facilitators and accept application-level information and requests in return. Facilitators use the information provided by these agents to transform these application-level messages and route them to appropriate places. In effect, the agents form a "federation" in which they surrender their autonomy to their facilitators and their facilitators take the responsibility of fulfilling their needs. FIPA 1997 Specification (FIPA 97) defines a specialized Domain Facilitator agent for each domain whose job is to maintain an Agent Directory for that domain and facilitate communication between agents of that domain.

Broadcast communication.

In situations, where a message has to be communicated to all the agents in the environment, or the sender agent does not know who the recipient will be (like when it announces a task and has to choose from all possible agents that can perform that task) then according to Tilley (Tilley 1996), it has two choices: it can either physically broadcast the message to all the agents in the system, or it can maintain individual communication links with all the agents in the system and send each one of them a directed message (using the TCP/IP protocol). When the message length is substantial, and there are a large number of agents in the system, the network bandwidth used to transmit the message is significant. Maintaining individual links implies that multiple copies of the same message have to be sent to each receiving agent.

Broadcasting, on the other hand, prevents network overloading by doing away with the need of making multiple copies of the same message and transmitting them to different agents. It helps in implementing a totally autonomous, scaleable and flexible multi-agent system as agents can leave or join a system without needing to inform anyone, provided they have completed all the tasks they were engaged in, which affected the actions of other agents. It is not uncommon to implement a hybrid approach by using broadcast communication to find identity and addresses of agents in the community, and then use that information to engage in directed communication.

Two main, popular approaches in broadcast communication are: the contract-net approach and the specification-sharing approach. The contract-net approach was proposed by Davis & Smith (Davis & Smith 1983). In the contract-net approach to inter-operation, agents, in need for service, distribute requests for proposals (broadcast messages) to other agents. The recipient of these messages evaluates those requests and if capable, submit bids (directed message) to originating agents. The originators use these bids to decide which agent gets the contract (directed message) for the broadcasted request. In the specification-sharing approach, agents broadcast their capabilities and needs and other agents use this information to coordinate their needs and actions.

Broadcast communication over the web is popularly referred to as webcasting (Frivold 1994). The strength of webcasting lies in the fact that the Internet can be used to transmit highly complex video, audio and other multimedia signals to any number of users all over the world.

Blackboard-systems.

In AI, the blackboard is an often used model of shared memory (Chaib-draa & Moulin 1987). It is a repository on which agents write messages, post partial results, and obtain information. It is usually partitioned into several levels of abstraction appropriate for the problem at hand, and agents working at a particular level of abstraction have access to the corresponding blackboard level along with the adjacent levels. In that way, data that has been synthesized at any level can be communicated to higher levels, while higher-level goals can be filtered down to drive the expectations of lower level agents.

Interaction: An Introduction to Speech-Act Theory

Gasser defines interaction to mean a type of collective action wherein one agent takes an action or makes a decision that has been influenced by the presence or knowledge of another agent (Gasser & Bond 1988). It is an inherently distributed concept as it is based on the coordinated action of participating agents. Since action in the system is usually goal-directed, many interactions are derived from goals. This is an important basic concept in implementation of multi-agent systems as it is the process of interaction that makes it possible for several intelligent agents to combine their efforts (Gasser & Bond 1988).

The inherently heterogeneous and distributed nature of a multi-agent system makes the implementation of interaction among agents a difficult process. Thus, a prerequisite for the successful development of multi-agent systems is an expressive common language for communication, with agent-independent semantics through which agents can communicate with their peers by exchanging messages, interacting together through explicit linguistic actions. In fact, this is where agents differ from objects. Objects can interact with each other by accessing object dependent public methods, but these methods may differ from one object to another. The agent communication language in a multi-agent system should be independent of the agents and independent of their internal data structures. This necessitates the need to know what knowledge to represent for communicating and how to do it. As communicating agents will have different knowledge bases, the communication language system must allow for these differences, so that communication and cooperation will succeed despite these disparities (Gasser & Bond 1988). Thus each agent needs a linguistic layer supporting an agent-independent semantics system, which provides a message-based interface that is independent of the agent's internal data structures and algorithms.

In the multi-agent systems, community speech-act (Austin 1962) theory is one of the most common methods used for constructing the linguistic layer and for formalising the linguistic actions of agents. Speech Act theory (Austin 1962) has made major contributions to understanding the relationship between an agent's internal state and the utterances it exchanges with other agents. It conceives communication and interaction in a framework involving goals for utterances, a knowledge of the participants, and planned actions for changing that knowledge to provide a unified description of action and communication where communication is treated as actions. It originates from the observation that utterances by agents are not simply propositions that are true or false, but actions that convey some belief or knowledge or an intention. Speech

Act Theory uses the concept of performatives to allow an agent to convey its beliefs, desires and intentions. The performatives are the speech-act component of the language and determine what one can "do" or "perform" with the content of the message. For example, performatives "assert," "affirm," "state," convey a belief, performatives "ask," "order," "enjoin," "pray," or "command" convey a wish or a desire, and performatives "vow," "pledge," or "promise" convey an intention (Searle 1969).

A speech-act language, which is commonly used in the multi-agent community, is KQML (Knowledge Query and Manipulation Language). KQML is a DARPA Knowledge Sharing Initiative contribution. It facilitates high-level cooperation and interoperation among artificial agents (Finin & Wiederhold 1993). Such agents may range from simple programs and databases to more sophisticated knowledge-based systems and they communicate by passing "performatives" to each other. KQML performatives form the heart of the language. KQML supports many different performatives.

The FIPA 1997 Specification V 1.0 (FIPA 97) defines an interaction protocol as an explicitly shared multi-agent plan containing communicative acts (like speech-acts). The specification formally defines the language semantics, using a Semantic Language SL. SL propositions are expressed in a logic of mental attitudes and actions and formalised in first order modal language with identity and can be used for actual representation of message content. The mental modal of an agent is based on the representation of three primitive attitudes: belief, uncertainty and choice. A fundamental property of SL's proposed logic is that the modelled agents are perfectly in agreement with their mental attitudes.

Parunak (Parunak & Van Dyke 1996) suggests using case theory as an alternative formalism for analysing the behaviour of individual agents and their interaction with each other. According to him, it is an important tool whose value for engineering agent-based systems has been demonstrated at three levels - *Knowledge Representation, Identifying Agents* and *Modelling Behaviour*. The universal nature of cases suggests that they represent a fundamental characteristic of human thought, one that transcends cultural and linguistic differences. Such a fundamental characteristic is of vital importance in engineering artificial systems.

Heterogeneous Collections Of Agents.

These involve an integrated approach whereby different agents can communicate as part of a larger system. To this end several languages have been developed such as ACL (Agent Communication Language) (Genesereth & Ketchpel 1994) that offer several advantages over other stand-alone agent systems:

> Stand alone applications can gain *synergy* by integrating with other stand alone applications.
>
> The Legacy problem can be overcome by allowing older non-agent systems to interact with new agent systems, thus reducing costly re-writes.
>
> Agent based systems provide a radical new approach to software engineering, whose ramifications can not be guessed *a priori*

It has often been noted that agent-based software engineering is similar to Object Oriented engineering. It does, however, differ in the fact that the meaning of messages in Object Oriented engineering may differ between different objects (*Polymorphism*), whereas this is not the case in Agent Based Software Engineering. Although not strictly an agent language, Java is widely used as the basis for most agents, and is the base from which standard agent languages are developed. Interestingly, Java removes the concept of Polymorphism and Operator Overloading.

An important case for the use of *heterogeneous* agents is in Legacy systems. Here three approaches can be taken. The first is to re-engineer the software such that it fulfils its functionality but also exists in agent form; this however is prohibitively expensive. Secondly, a proxy agent acts as an intermediary and can send and receive information to other agents but communicates back to the Legacy system in the native language of the Legacy system (a *Translator*). Lastly, a *wrapper* approach can be taken, whereby some code is *injected* into the Legacy system that encapsulates it, and can therefore access the internal data structures directly. This method is regarded as the best, most efficient method, but it does require the source code to be available.

Most of the systems currently available user the *transducer* (proxy) approach, whereby information systems have an agent front end attached, and therefore talk to other agents via the proxy agent.

Conclusion

Multi-agent systems make an ideal building platform for complex, ill structured domains such as process control. They have several key properties that make them particularly suitable in these domains such as modularity, decentralised, changeable, and ill structured and complexity.

When considering multi-agent systems it is imperative that the correct software language and tools are selected to match the problem domain (Chapter 7).

Multi-agent systems will operate completely ineffectively if the wrong intra-agent communication strategy is selected. It is important then, that at design time the communication strategy is incorporated into the system design. Key communication strategies include directed, federated and broadcast communication.

Not only must a general strategy be selected, but also the protocol of message passing must be defined. Speech act theory provides a useful basis for intra-agent message parsing, and KQML is a widely used protocol in heterogeneous multi-agent systems, where ontology and syntax are vital to agents correctly recognising the semantic meaning of a message.

This chapter has shown how well suited multi-agent systems are to developing an adaptive system for process control. It has also highlighted how important a clear communication strategy is for correct agent operation, and how important it is to select the most appropriate agent building tools for the job.

The next chapter examines the nature of the specific adaptation problems encountered in a process control environment, and discusses some multi-media principles and heuristics that could be incorporated into the system to address these adaptation problems

Chapter 6

THE WHY, AND HOW, OF ADAPTATION

Introduction

Given that adaptation, in principle at least, offers a way forward for improving operator interfaces, two really important questions remain before an implementation can be effected.

Why should the system adapt? What are the rules governing the adaptation process itself

How should it adapt? What are the basic elements or units that can be adapted and what rules should be applied in the process

The Nature of Adaptation

In order better to understand the nature of adaptation in AMEBICA, it is necessary to examine the phenomenon of adaptation in general. The main defining characteristic of an adaptive system is

> *"its ability to maintain a stable state or an equilibrium in spite of disturbances and influences from the outsid*e. *In extreme cases when the disturbance is prolonged, an adaptive system will modify its internal state so that the disturbance is now part of normal conditions*."

This definition supports two different views of adaptation. In the former case, the organism reacts in a temporary way, reverting to its original state when the disturbance is removed. In the latter case, when the disturbance persists for a long time, the internal state of the organism changes to match the nature of the disturbance so that the organism is eventually in a normal state in the presence of the disturbance (i.e. the disturbance is no longer a disturbance). In this situation the re-establishment of the original conditions will now be seen as a disturbance. However, it is difficult to envisage a Supervisory and Control System switching permanently into a new state (except in so far as start-up and shut-down might be classified as separate states).

It is therefore necessary to distinguish between *short-term adaptation* (or intelligent reaction) and *longer term true-adaptation.* We say "intelligent reaction" because adaptation must be more than a simple reaction to a stimulus (for example, a pop-up window appearing when a button is pressed). From now on therefore the thesis will concentrate upon "intelligent reaction" rather than permanent change of state.

The purpose of adaptation in process control, therefore, is to maintain the system's ability to perform according to its specifications, i.e. to maintain its equilibrium state. This is true regardless of whether the adaptive system is a technological system, a human operator or a team, or - more interesting for AMEBICA – a joint human-machine system. The joint system of human team and machine must be able to maintain performance at an acceptable level, despite potentially disrupting events. This is achieved by activating an appropriate response to the disturbance. During this response the goals of the system, hence the nature of its performance, may change, for instance from keeping the system running in a stable state, to establishing a safe shutdown state. In either case the performance must remain within acceptable levels. This happens, of course in conventional systems. However, it is the operators who have to do all the adapting. What we are seeking is a more equitable system where the interface system itself contributes to the adaptation process.

Consider the temperature regulation of the human body, which entails a kind of thermostat. Human beings can only function if the body temperature is kept within rather narrow limits, around 37.4 C. Whenever the body temperature becomes too high, e.g. during fever or vigorous exercise, a number of functions are activated to cool it down, such as sweating and increased blood circulation. Conversely, when the body temperature becomes too low, a different set of functions are activated, such as shivering, retraction of blood circulation from the extremities (cold fingers), etc. Both effects can be amplified by changing the environment, or moving to a different environment (moving to a cooler place or a warmer place).

The example shows two main characteristics of adaptation. Firstly, that there must be a set of conditions for starting and stopping the adaptation – specifically a set of initiating conditions or a triggering event. Secondly, there must be one or more appropriate responses or functions that can be used to respond to the disturbance. The initiating conditions mean that compensating functions are not activated all the time, but only when certain conditions are reached or a certain threshold has been passed. (Conversely, the functions cease when another set of conditions has been fulfilled.) Any adaptive system must therefore entail a definition of the initiating conditions and a specification of the functions that are activated when the initiating conditions are met. More generally, the initiating conditions can be seen as defining a specific goal, namely that the disturbance or deviation has been neutralised or counterbalanced. The compensating functions must be capable of achieving this goal – without at the same time introducing new disturbances in the system or the environment.

Adaptation will therefore not be activated all the time, but only when certain conditions are reached or certain thresholds have been passed. A set of initiating conditions or triggering events is thus needed to begin adaptation. Once this has been decided, one or more appropriate responses or functions are needed to activate the adaptation process, which will cease when some defined terminating set of conditions has been fulfilled.

There are only two main sources of information that will give a guide as to whether the system (man and machine) is deviating from equilibrium - the actions of the operators at the interface, and deviations in the process state. Whatever triggering mechanism is eventually decided, the source of adaptation information must be combinations of the operator state and process state. The adaptive response functions can only be a set of recommended modifications to the rendering system.

The Notion of an Adaptability Matrix

To help designers building an adaptive process control interface, an *Adaptability Matrix* has been developed that captures the above general mapping principles between triggers (operator and system state changes) and actions (modifications to the renderings). The dimensions of the matrix consist of the identifiable operator states on the one hand and the identifiable process states on the other. The current proposal (a very limited first attempt) has four operator states and three process states leading to a matrix with twelve cells.

The matrix has the following four operator triggers.

> **Normal response**: The responses of the operator are normal, i.e., the operator is capable of handling the situation. No loss of control is recognised.
>
> **Delayed response**: Some responses of the operator are delayed. Normal time responses for certain categories of events are exceeded.
>
> **Erratic response**: The operator sometimes fails to perform actions correctly enough to warrant attention but not enough to be considered disorganised.
>
> **Disorganised response**: The frequency of erratic responses is so high that performance is considered disorganised. The operator has clearly lost control of the process, and cannot maintain the overall goals. The loss of control may also be determined in terms of the strategies adopted, such as opportunistic search.

The matrix has the following three process state triggers.

> **Normal process state**: The process is in a normal state, as defined by key process parameters (e.g. critical functions or safety functions).
>
> **Disturbed process with high information rate**: Typically, the information rate is high when a disturbance occurs and for a limited period of time thereafter.

Disturbed process with low information rate: This case corresponds to the later stage of a disturbance, when the rate of information has gone down, but the process still has not been recovered.

These, of course, are not the only set of triggers possible. There could be a rich variety of triggers perhaps more fine grained than these. However, even these relatively simple triggers are difficult to measure (particularly the operator triggers), so we have tried in the first instance to keep them simple. The matrix (with its triggers filled in, but not the adaptive functions) is shown in Figure 21.

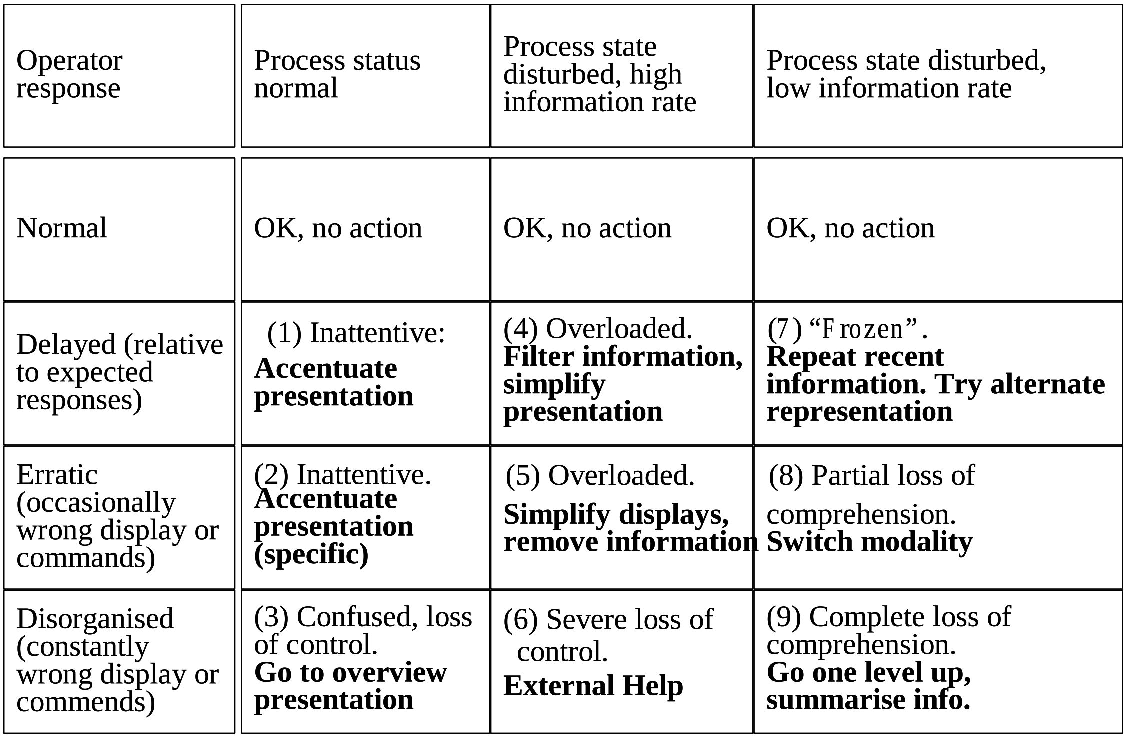
Operator response	Process status normal	Process state disturbed, high information rate	Process state disturbed, low information rate
Normal	OK, no action	OK, no action	OK, no action
Delayed (relative to expected responses)	(1) Inattentive:	(4) Overloaded.	(7) "Frozen".
Erratic (occasionally wrong display or commands)	(2) Inattentive.	(5) Overloaded.	(8) Partial loss of comprehension.
Disorganised (constantly wrong display or commands)	(3) Confused, loss of control.	(6) Severe loss of control.	(9) Complete loss of comprehension.

Figure 21 The Triggers in the Adaptation Matrix

Whilst process-overload and process-underload conditions are not too difficult to measure, measuring the operator states is more problematic. In this first raw cut approach, Delayed

Response will be indicated by significant delays in alarm answering, Erratic Response by inconsistent delays including wrong commands, and Disorganised Response by random delays and wrong commands.

As far as adaptive functions are concerned, no action is taken if the operator response is normal. The process may be in a disturbed state but the combined operator-process system is handling it. The rest of the function cells are currently filled in with a suggested description of the probable operator state, varying from inattention to loss of comprehension.



Operator response	Process status normal	Process state disturbed, high information rate	Process state disturbed, low information rate
Normal	OK, no action	OK, no action	OK, no action
Delayed (relative to expected responses)	(1) Inattentive: **Accentuate presentation**	(4) Overloaded. **Filter information, simplify presentation**	(7) "Frozen". **Repeat recent information. Try alternate representation**
Erratic (occasionally wrong display or commands)	(2) Inattentive. **Accentuate presentation (specific)**	(5) Overloaded. **Simplify displays, remove information**	(8) Partial loss of comprehension. **Switch modality**
Disorganised (constantly wrong display or commends)	(3) Confused, loss of control. **Go to overview presentation**	(6) Severe loss of control. **External Help**	(9) Complete loss of comprehension. **Go one level up, summarise info.**

Figure 22 Suggested adaptive functions in various conditions

The matrix is then populated with suggested adaptive functions. For example, in cases of information over-load some form of filtering might be appropriate, whereas in under-load situations moves to higher levels of representation might be more appropriate. The current suggestions are shown in Figure 22.

For each cell two descriptions are given. The first, in normal typeface, represents an assumption regarding the operator state. The second, in bold, is the response goal of the adaptive system. Some of them are clearly also of an illustrative nature, such as External Help – meaning that it is necessary in some way to communicate with the operators to order and get them back to the situation. Note that the normal situation is not confined to the normal state of the process, but may also include disturbed states where the operators have no problems in responding correctly and bringing the process back on the track. Here adaptation should be avoided, since the operators should not be needlessly subjected to unnecessary disturbance.

As an example, consider the cell in the middle, corresponding to erratic operator performance and a disturbed process state with high information rate. The "diagnosis" of the operator state in this situation is information or task overload. The situation is one where there is a large amount of information coming to the operator, and the fact that there are a number of incorrect actions suggests that he or she is not completely in control of the situation. Assuming that the "diagnosis" of operator overload is correct, a reasonable goal is to simplify the presented information by removing unnecessary information. This goal can be passed on to the specific adaptive system agents, who will (hopefully) have appropriate responses ready. In practice, this could be achieved by zooming out to a higher level of abstraction, removing detail information such as process measurements, removing unimportant alarms, or highlighting essential features.

The Adaptability Matrix tells the adaptive system what to do in general terms when the triggering conditions are met – HOW to trigger adaptation. All these triggers will then involve manipulating the interface in some way – in other words the renderings will be adapted. Adaptation can be of many forms – accentuation, switch of medium, switch of representation, higher or low level views, and alterations to presentation such as zooming or translation.

More Detailed Discussion of Possible Adaptations.

In this section we attempt to illustrate use of terms in the matrix. In each case, the symptom, state, and adaptation function will be presented.

Normal System State – Delayed Actions

Symptom: alarms are outstanding for longer periods than necessary. Acknowledgements are outstanding.

State: In this state there is no reason for operators to delay their responses. Thus the state implies that operators are *inattentive* (they may be on the telephone or talking amongst themselves or simply day-dreaming). The adaptive response therefore is to draw their attention to any delayed alarm responses – to accentuate the presentation of the relevant conditions in some way.

Adaptation: We call this *Accentuation Adaptation*. It could be of various types. One possibility is a switch to audio alarms. Another to highlight the display of the relevant information in some way. There could even be a hierarchy of actions if the operators do not respond. In our current architecture we have implemented such behaviour, though we did not drive it from the operator state.

The response is simply to draw the operator's attention to the suspected condition.

Normal System State – Erratic responses

Symptom: A succession of alarms and acknowledgements are outstanding, but some have been acknowledged on time.

State: This may imply a little more that inattentiveness. The operators may be attending to other ancillary jobs (recording information, shift switch over etc.). It might also be the result of inexperience. This requires more than accentuated adaptation. The attention of the operators needs to be achieved in a more focussed manner.

Adaptation: We call this *Focussed Accentuation Adaptation*. Not only should the alarms be highlighted, but also additional structure information may be needed (focus the process diagram on the affected area, or group the alarms together).

Normal System State – Confused Responses

Symptom: The operator pattern of activity cannot be related to likely goals. Wrong alarms are being acknowledged. Unnecessary actions are being carried out. Operator activity is not matching system activity.

State: Inexperience could be a major factor here. The operators are misunderstanding normal activity as abnormal activity. The only danger is that erratic responses might cause instability in the system.

Adaptation: We call this *Reassurance Adaptation*. Firstly the operators must be assured that all is normal. The priority of alarms could be restated and put in perspective. Then a higher level (higher abstraction level) presentation should be presented

Disturbed System State (High Information rate) – Delayed Responses

Symptom: The system is providing information at a higher rate than normal. There are many alarms and they are usually acknowledged but the delay to acknowledgement is getting longer. The operators are just not able to handle the quantity of information being presented.

State: High information rate in the process model. Delays for acknowledgement getting longer and longer. However the correct actions are being carried out.

Adaptation: We call this *Filtering Adaptation.* The need is to remove any extraneous information so that the operators can concentrate upon the issues being presented. We require simpler, more focussed displays, summaries of alarms (ordered by area).

Disturbed System State (High Information rate) – Erratic Responses

Symptom: The system is providing information at a higher rate than normal. There are many alarms and some are acknowledged incorrectly or not at all. The delay to some acknowledgements is getting longer but others are acknowledged promptly. The operator's judgment is now being affected by the quantity of information being presented

State: High information rate in the process model. Delays for acknowledgement getting longer and erratic. Some correct actions are being carried out in limited areas.

Adaptation: We call this *Focussed Filtering Adaptation*. There is an urgent need is to remove extraneous information so that the operators can concentrate upon the issues being presented. Furthermore information should be reorganised to facilitate understanding. We require simpler, more focussed displays, summaries of alarms (ordered by area). Overview diagrams may be needed and accelerated time repeats of how the situation developed could be presented.

Disturbed System State (High Information rate) – Confused Responses

Symptom: The system is providing information at a higher rate than normal. There are many alarms and few are acknowledged correctly or not at all. The delay to some acknowledgements is serious. Many inappropriate actions are occurring (often repeated). The operator's judgement is seriously affected by the quantity of information being presented. The system is beginning to take evasive action on its own.

State: This is a serious state. High information rate in the process model. Delays for acknowledgements are getting longer and more erratic. The system will begin taking automatic safety actions. The operators need outside help from Engineers or supervisors.

Adaptation: We call this *Confusion Adaptation*. The operators need to be focussed on remedial actions not on performance goals. The information must be simplified to present only the necessary actions to stabilise the system. Explanations of what remedial action the system is taking is needed and projections as to what will happen next if nothing is done (time to further remedial actions). Calls should be made to get additional help.

Disturbed System State (Low Information rate) – Delayed Responses

Symptom: Alarm states are reducing, but this may be because the system is now automatically responding by shutting down functions. The system is highly disturbed. It is now becoming disabled

State: Highly disturbed. Partially disabled**.**

Adaptation: We call this *Inaction Adaptation.* The operators are sometimes puzzled as to what to do and are spending long periods trying to work out what is wrong. Their thinking is not completely at variance with the System State, but they are slowly losing the battle. Information needs to be repeated in a focussed manner. Higher level views may be necessary. A change in representation might actually be helpful (say a Mass or Energy flow diagram). The operators need ton get a grip on the situation from a higher level

Disturbed System State (Low Information rate) – Erratic Responses

Symptom: Highly disturbed system state. Partially disabled. Operator actions sometimes correct sometimes wrong.

State: Highly disturbed. Lack of understanding at times of what is going on.

Adaptation: We call this *Comprehension Adaptation.* The operators are losing control. High level views are required to regain control at a high level. Then the information can gradually be brought to lower levels. Time replays may be useful here. The trick is to get back to basics. What is really basically wrong? Other representations may assist you, including media changes.

The rest of this chapter therefore deals with the question of HOW to adapt the interface given that a triggering has taken place. One set of issues concern the ways in which media may be manipulated. Another related issue concerns how to adapt when there are a number of choices available. A third set of issues will concern consistency. If adaptations are taking place over time, how can the system ensure that a consistent viewpoint is presented to the operators. Alty et al (Alty et al, 1992) have shown that there are no absolute rules about media usage. The appropriateness of a medium will depend on the state of the process, the task, and the operator. SO, as well as having access to the operator state and system state, the system will need to have access to a knowledge source which advises it on Human Factors issues – this is called the HUMAN FACTORS DATABASE.

The Development of a Human Factors Database

This section examines what rules and heuristics might be incorporated in the Human Factors Database to enable it to determine the most appropriate type of representation (and its associate parameters) for a particular context. It describes the general characteristics that the system should exhibit and develops a potentially useful set of rules that will govern the selection of representations. It is expected that the set of rules developed for the Human Factors Database, will be customisable at design time to match the nature of the particular process the adaptive system is being employed with.

User Interface Design Guidelines

Within the literature there are several examples of user interface design guidelines that provide a grounding for a set of potential rules for the Human Factors Database. (Smith & Mosier 1984) offer a set of general guidelines. For example, three of their guidelines are:

> At any steps in a transaction, ensure that whatever data a user needs will be available from the display

> Use short simple sentences
>
> For size coding a large symbol should be 1.5 times size of next smallest

These are useful but inevitably quite general. For example, the first principle is pertinent but simply moves the problem to defining what is needed. More detailed guidelines have been suggested by Nielsen (Nielsen 1992). The ideal is to present exactly the information the user required –and no more- at exactly the moment it is needed. Principles of graphic design and an adaptive interface can also help operators prioritise their attention to a screen by making the most important elements stand out.

> Minimise Users Memory Load: "Displaying to many objects and attributes will result in a loss of salience for the ones of interest to the user, so care should be taken to match object visibility as much as possible with the users need.
>
> Be consistent: "The same information should be presented on the screen in the same location on all screens and dialogues".

It can be argued that the latter point has less importance than the other two heuristics, since an adaptive interface can never be truly consistent otherwise it would static rather than dynamic. Although useful these guidelines are not always applicable in a process control context.

The remainder of this chapter will summarise specifics of how each media might best be used, and what rules could be implemented to make the Human Factors Rules Database as effective as possible. The rules were generated from several sources including (Faraday 1998, Pedersen 1999, Smith & Mosier 1984, AHCI 1998)

A number of "ad-hoc" rules have been gathered together that provide mechanisms for dealing with different types of media in different circumstances. The information types are divided into three groups – Descriptive, Operational and Organisational (Faraday 1998).

The following set of guidelines can be codified as a set of rules within the Human Factors Database, and customised at design time to match the nature of the particular process the adaptive system is being employed with.

Determining What General Form of Medium Is Appropriate for A Particular Context

It is necessary to determine what type of information the Human Factors Database is dealing with, because information types can be used to map different types of media dependant on the nature of the information. Task characteristics influence the modality of the media resource used, for instance, verbal media are more appropriate to language based and logical reasoning tasks: whereas visual media are suitable for physical actions involving moving, positioning and orienting objects (Faraday 1998). I

In Figure 23 (Faraday 1998) see an overview of how different information types are related. Each information selection rule (SR) links the required information type with appropriate media.

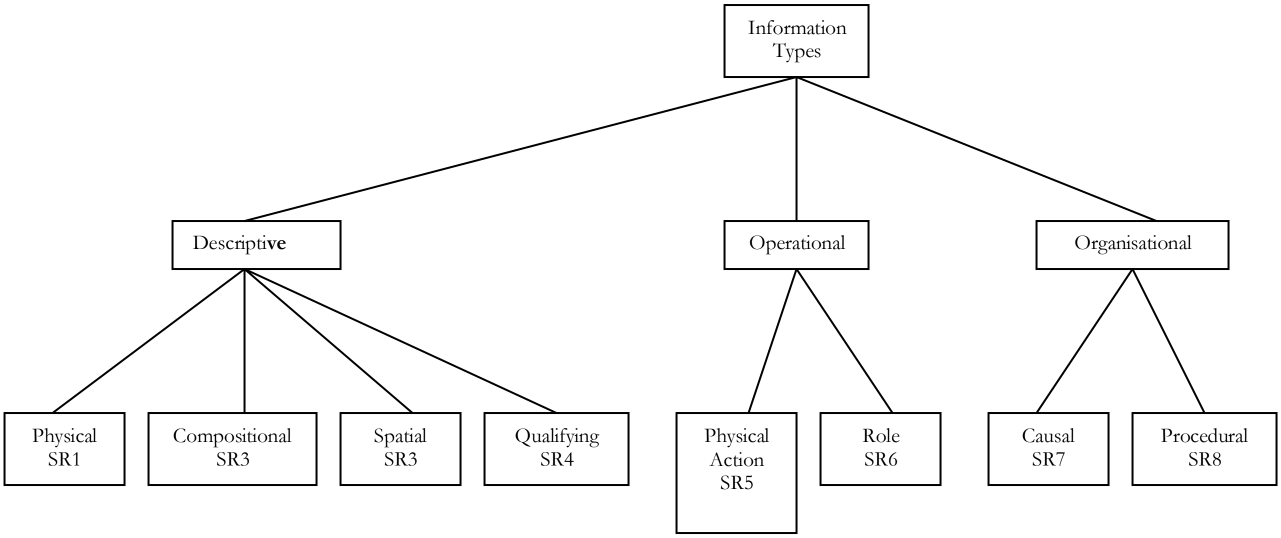


Figure 23 Overview of Information Types and Related Selection Rules (SR)

Descriptive:

Physical SR1: If the task sub-goal requires physical information then prefer a visual medium. Language is poor at describing object detail and appearance (Bieger & Glock 1984)

Caveat: If an object has to be identified which may be difficult for the user to recognise, use language-text to indicate identity (Jorg & Hormann 1978).

Composition Sr2: If the task sub-goal requires composition information then use a visual medium (Bieger & Glock 1984)

Caveat: Language can be used to set the granularity of the components in the image e.g. "Look as sub assembly X"

Spatial SR3: if the task sub goal requires spatial information then prefer a visual medium (Bieger & Glock 1984)

Caveat: Language- text captions can be used to identify landmarks and components for spatial information

Qualifying SR4: If the task sub-goal requires qualifying information then use linguistic media (Boohrer 1975*)*

Operational

Physical action SR5: If the task sub-goal requires physical action information then prefer a visual medium animation for complex actions e.g. those with complex paths or manipulations (Sweezy 1991) use still image for simple actions (Sweezy 1991)

Caveat: Language should be used to identify and amplify explanation of complex actions, or groups of action; or to qualify actions (Boohrer 1975*)*

Role SR6: if the task sub-goal requires role information to identify agents or objects involved in the action, then use a linguistic medium to add this information to the image or animation sequence.

Organisational:

Causal SR7: If the task sub-goal requires procedural information then prefer a linguistic medium to provide sequencing relations, particularly when the time frame is not constant (Burch 1973*);* animation or image sequences will be useful to support any underlying physical action information.

Procedural SR8: If the task sub-goal requires causal information prefer a linguistic medium to provide key causal relations *(*Mayer & Anderson 1991*),* with animation or images sequences to support any underlying actions being related.

The Visual Medium

Still Visual Image

Still Visual Image – Physical Information

If an unusual or complex image is to be displayed show the whole object unobscured.

If the task requires a specific object to be identified, or requires details of object properties, then use colour and texture in images to identify the object.

Still Visual Image – Compositional Information

Information is gleaned initially from an overview of the image, so the adaptive system needs to apply little descriptive detail to major objects. It should apply attentional effects (e.g. highlight) if the object is important.

By default objects focused on will be those which are: bright in colour, set apart from other objects, larger in size, shown in more detail, in sharp focus or nearer the front of the scene. These should be the important objects.

Still Visual Image – Spatial Information

If a particular object needs to be located accurately, then the Human Factors database should se it as a landmark. Landmarks should be perceptually salient objects with which the user is already familiar.

If several objects are to be located as landmarks, he Human Factors Rules Database should divide the image into sub areas, and select a landmark for each area.

Still Visual Image – Attentional Guidelines

To draw attention to a group of spatially distributed objects the Human Factors Rules Database should set a common visual attribute e.g. change to the same colour. To emphasise a group of co-located objects it should highlight the background or draw a box around the objects.

The Human Factors Rules Database can place icons by or on an object to draw attention to the object, this function can be used if highlighting may obscure details.

The Human Factors Rules Database can utilise labels linked to objects as a means of drawing attention and providing supplementary information. This method is particularly effective when labels are dynamically revealed to direct the users reading sequence.

The Human Factors Rules Database should avoid showing an object in motion or using a highlighting a technique when the user is extracting information from an image. It should allow at least a second before changing the image.

The Human Factors Rules Database should limit the use of too many highlighting techniques within an image at once; instead it can sequence highlights to move attention from one object to another.

Linguistic Media – Use Of Text

Linguistic Media – Procedure Information

The Human Factors Rules Database can use text for the purpose of displaying procedure sequencing information. It should utilise cue phrases for maximum effect. A cue (e.g. "at time x”) should be used to locate a particular important time point in the task, such as the start or end of a sequence

The Human Factors Rules Database should structure text to indicate procedures order by formatting the text into lists or frames

Linguistic Media – Causal Information

The Human Factors Rules Database should use cue words to indicate causal relationships: “because, in order to, resulting in”

Linguistic Media – Attentional Guideline

The Human Factors Rules Database should:

Display text before the image or make the text area larger, if it deems that the focus should be on the text prior to the image. Generally an image will be focused on before text.

Always set text a display time of at least 6 seconds. This allows the operator sufficient time to read the text. Simple words require 200 msecs each.

Use highlighting, bold or large fonts to make a particular word or clause of text stand out.

Use paragraphs and titles as entry points to direct attention to the required part of a text.

Choice of Static “v” Dynamic Media

The Human Factors Rules Database should:

Use text or still images for important information that must be attended to, as information may be lost from time varying media. Memory for the content of dynamic media is generally worse than for static medium.

Be aware that animation media will dominate over static image media, as attention is drawn by default to motion or stimuli that change.

Favour text if the content to be presented is complex or lengthy, if the content is simple or short then it should favour speech

Re-Enforcement.

If the Human Factors Rules Database deems that re-enforcement of a representation is necessary (perhaps the initial representation has not been acknowledged within a pre-defined time limit), then it should:

Present the same event in two (or more) modalities e.g. an animated demonstration of a procedure is accompanied by a voice description

Repeat the same content (if no further content is available) again but acknowledge that this is a less effective method than providing content with more information.

Colour Coding Applications

The above sections detail how the Human Factors Rules Database might decide upon the type of visual medium selected, it also must decide upon the parameters of the representation selected. An important parameter for visual representations is colour. Colour coding is most effective when used with some other display feature, such as symbology or actual text content, or with another coding method such as size. Therefore when selecting the parameters of a representation the Human Factors Rules Database should utilise one of the other main parameters as the primary code and colour as the secondary code.

The Human Factors Database should be loaded with a standardised colour palate and apply it consistently across all process applications to ensure that the operator can make the proper interpretations. Colour coding should also be consistent with the relationship of the label colour and the colour associations of the words in the label. The Human Factors Rules Database should use colour to:

Attach specific meaning to process information presented in the form of text or symbology. (for instance the use of industry standard colours for alarm priority (red, for instance, implying highest priority)).

Direct the operator's attention to the most important or time-critical information on the screen.

Enable an operator to rapidly differentiate among several types of information, especially when the information is dispersed on the display or contains complex computer-generated symbology.

Increase the amount of information portrayed on a graphic display by adding colour in addition to shape.

Indicate changes in the status of graphical data.

It is important that the Human Factors Rules Database use colour coding conservatively, so that only a few colours are used to designate critical categories of displayed data and only where it will help operator performance. When overused, colour may impede rather than enhance performance.

Presentation and Formatting of Auditory Information

The effectiveness of any auditory display is dependent upon the environment within which the display must operate. A spoken message may be easily obscured by other spoken messages; a particular tonal signal may be masked by or confused with other similar tonal signals; and any auditory signal or message may be masked by frequent, loud bursts of noise, such as would occur during an emergency. In general, periodic tones and non-periodic complex sounds are easily generated and are appropriate for various types of information display. Periodic tones are good for automatic communication of limited information, however, the meaning must be learned.

Complex sounds are useful when the sounds are easily identified and have an inherent common meaning to the user (e.g., fire alarm signals).

Voice warnings and messages are more flexible than simple sounds, because they can provide more information while alerting the operator to a problem. This may be especially important during high workload, when the meaning of a signal may be forgotten. Voice messages are most effective for the rapid communication of complex, multidimensional information. Because the meaning of speech is intrinsic in signal and context when standardised, there is a minimum of learning required of the user. Voice technology has been shown to offer some advantages in specific situations to both visual and other auditory methods of presenting information. These include: the display of alarm identification and location information; the display of process state and the display of warnings.

Selection Criteria

The following guidelines indicate how to select the type of auditory display that is most appropriate to a particular process application.

Usage

The Human Factors Rules Database should use auditory renderings when:

> The information to be processed is short, simple, and transitory, requiring an immediate or time-based response.
>
> The visual display is restricted by over-burdening, ambient light variability, environmental considerations, or anticipated operator inattention.
>
> The criticality of transmission response makes supplementary or redundant transmission desirable.
>
> It is desirable to warn, alert, or cue the operator to subsequent additional response;
>
> Custom or usage has created anticipation of an audio display

Voice communication is necessary or desirable.

Criteria for Auditory Tonal Presentation

Auditory tones should be used by the Human Factors Rules Database when:

The message is extremely simple;

The operator has special training in the meaning of coded signals;

A signal designates a point in time that has no absolute value such as a specific point in a sequence of events;

The message calls for immediate action;

Voice signals are overburdening the operator;

Conditions are unfavourable for receiving voice messages (tonal signals can be heard in noise that makes speech unintelligible);

Voice communication channels are overloaded.

Criteria for Voice/Speech Presentation

Voice messages should be used by the Human Factors Rules Database when:

Communication flexibility is necessary;

The identification of message source is necessary;

A simple coded signal cannot adequately give direction or instructions to the operator;

When the operator does not have special training in coded signals;

The message deals with a future time requiring some precision

Potentially stressful situations are occurring (alarm flood) that might cause the operator to “forget” the meaning of the auditory code

Ambient masking noise characteristics prevent the use of simple tonal signals

Other complex tonal signal possibilities have already been exhausted (i.e., have been assigned and cannot be duplicated).

Table 2 presents the advantages and disadvantages for different types of audio signals based on the function to be performed.

FUNCTION	**TYPE OF SIGNAL**		
	Tones (Periodic)	**Complex Sounds (Non-Periodic)**	**Speech**
Quantitative Indication	Poor Maximum of 5 to 6 tones recognizable when ideally spaced and sounded alone.	Poor Interpolation between signals inaccurate.	Good Minimum time and error in obtaining exact value is compatible with response.
Qualitative Indication	Poor to Fair Difficult to judge approximate value and valuation of deviation from null setting unless presented in close temporal sequence.	Poor Difficult to judge approximate deviation from desired value.	Good Information concerning the process presented in form compatible with response.
Status Indication	Good Start and stop timing. Continuous information where rate of change of input is low.	Good Especially suitable for irregularly occurring signals.	Poor Inefficient; more easily masked; problem of repeatability.
General	Good for automatic communication of limited information. Meaning must be learned. Easily generated.	Some sounds available with common meaning, e.g., “alarm ”. Easily generated.	Most effective for rapid (but not automatic) communication of complex, multi-dimensional information. Meaning intrinsic in signal and context when standardized. Minimum of new learning required.

Table 2 Functional Evaluation of Audio Signals

Speech & Sound Attentional Guidelines

The above sections detail when the Human Factors Rules Database should use the auditory medium, and what form of auditory representation is most appropriate for a certain context. This section shows under what circumstances the Human Factors Rules Database should alter the parameters of a representation, and how it should alter the representation. The Human Factors Rules Database should:

> Use speech and sound to alert users to important information. Sound will focus attention initially over visual information, but attention may shift to visual media after a short time.
>
> Only use a single strand of speech or sound at any one time multiple strands of speech or sound will interfere with each other and distract focus. Speech will usually gain focus over sound.
>
> Emphasise information in speech using loudness or rate; the louder or more slowly spoken words will be more salient.

Media Combination & Ordering

Combining media can be a very effective tool, however incorrectly used it can lead to confusion for the operator. It is therefore important that "Contact Points" are placed between the associated media. Contacts Points are places in the presentation where the verbal part of the presentation needs to be related with the visuals. The problems contact points bring are:

> How to provide linking references between language and visuals. A contact point references can either be direct in which the language explicitly references the visuals e.g., 'look at X'; or indirect in which the whole visual is referenced e.g. "see figure below"
>
> How to ensure the message thread can be followed between visual and verbal media.

The rules below attempt to answer these problems, the Human Factors Rules Database should try to:

Use a direct contact point if the connection between information is an image and language is important e.g. direct the users attention to the object in the image by highlighting the object which is being spoken about "look at alarm 5".

Use indirect contact points if the connection between information in an image and language is less important – therefore direct attention to the media resource.

Ensure that the two types of media are available when the contact point is made, and are in focus. The Human Factors Rules Database should reveal text and image elements together if they share a contact point. Use highlighting techniques in visual media where the contact point is auditory cues for speech.

Present language before visual media when the two are combined so that the language "sets the scene" and direct the user's attention to information within the image.

Allow time for contact points to be formed e.g. pace the presentation to allow inspection of image, or speech

Use speech if animation is to be combined with language, as reading text will compete with viewing animation.

Consistency Measures

Consistency Across Functions

Where possible, the Human Factors Rules Database should set representations for different functions to be as consistent and predictable in terms of the current format schemes, organizational schemes, control responses, interface dialogs, and feedback as possible. Abbreviations, acronyms, and symbol meanings should be consistent throughout the process interface system.

Format Consistency

Display formats should have a consistent structure and layout that support similar functions throughout the operator interface. Common elements within a given format type (e.g., alphanumeric formats, menu formats) or between similar formats should be, wherever possible, consistently depicted and located.

Dialog Consistency

Control procedure representations and dialog behaviour schemes should be consistent in form, means, and consequences from one transaction to another, from one task to another, and from one application to another, to support the operator in transitioning between tasks as well as multi-tasking.

Feedback Consistency

The process interface should have a reliable and consistent method of system response across process applications. Process interface transactions made by the operator should produce a consistent perceptual response whether it is in visual, tactile, or auditory form.

Conclusion

There are two main forms of possible adaptation, short or long term. Long term adaptation implies complete change of state of the adaptation system to match the disturbance. Clearly in a process control environment this is impossible. Therefore the form of adaptation required for this thesis is short term compensating adaptation.

To fully understand the nature of adaptation required for a complex process control environment, it is necessary to define triggers and actions. These triggers and actions can be defined in a matrix where, the two main triggers in process control (process and operator) run along the axis.

There are a large number of guidelines that can be used to guide the selection process in the adaptive system. Clearly implementing a Human Factors database taking into account all the above factors is well beyond the scope of this thesis. However, a limited implementation was attempted to be used as a guide for the adaptive system. The Human Factors database developed was fairly primitive, and was tailored to act specifically to the scenarios developed for the prototype. However, it could easily be scaled up and populated with a full range of heuristics.

Chapter 7

AGENT SOFTWARE DEVELOPMENT ISSUES: THE WAY FORWARD

Introduction

This chapter contains a review of appropriate software technologies and toolkits to be used for constructing and optimising agents. It begins by assessing which software language is the most suitable for writing applications within the multi-agent domain. Then it examines the arguments for and against the selected language, and looks at the various methods and application environments that will help solve the deficiencies of the chosen language. It then goes on to describe the best runtime environment within which it should run. The chapter examines the constraints imposed on the selection of the runtime environment when dealing with many threads.

An examination was then made of various Integrated Development Environments (IDEs) and a description was formed of which packages are most appropriate when dealing with agent technologies.

There then follows descriptions if various available agent construction and development tools. These are assessed on a set of criteria that suit the needs of the process control described in Chapter 3.

Choice of Software language

Introduction

This section first examines two prime candidate languages that can be used for developing agent applications, Java and C++. It begins by examining the recommended candidate – Java – and defines the intrinsic features that make it the language of choice for developing agent applications. Then it compares Java with C++ to highlight this selection. It then assesses technologies that can be used to improve Java run-time performance. Finally, it examines the various Java enabled, Integrated Development Environments and toolkits available.

Why Java?

What are the main features that make Java a suitable language for agent development?

Platform Independent: Agents, by nature, are inherently distributed. Thus an agent application should, in theory, be able to execute anywhere on the network on any Java compliant hardware/software platform. Java provides the developer with a language that is architecturally neutral and platform independent. Therefore Java provides an ideal solution to agent distribution, and allows the system to be rapidly updated as changes to platforms are introduced.

Object-Oriented: Although Agents are more than objects, a language that provides a rigorous object-oriented set-up is highly desirable. Whilst other languages are based on the object-oriented paradigm, Java is one of the few that *enforce* these principles.

Multi-Threaded: Agents are, by common consent, autonomous creatures and thus should run within individual threads. Unfortunately, in many other languages, support for multi-threading is limited and makes programming of this type very difficult. Java provides built-in support for threads and provides a way to obtain fast, lightweight concurrency within a single process space. In addition, Java provides a very powerful system called RMI (Remote Method Invocation) that allows an object running in one Java virtual machine (VM) to invoke methods on an object running in a different Java VM

Network Supportive: Java has a well supported and easily extensible set of API's (Application Programmer Interfaces) that deal easily with both UDP and TCP/IP, and thus allow both uni-cast and multi-cast calls across a network

Further Features: Java provides database connectivity through SQL interfaces and has native language support(C++) for legacy systems. In addition it has an IDL API, which allows Java to be incorporated with CORBA if necessary.

Problems With Java.

Much of the argument against Java has been directed at its runtime performance. This problem derives from the fact that Java is a platform independent language and is therefore, by nature, an interpreted language. As a consequence, Java operates not by running directly on a native machine, but by operating on a *Virtual Machine* (VM). This Virtual Machine receives instructions from the Java compiled classes in the form of *bytecodes*, which the Virtual Machine interprets into native system calls. This process, in early Java applications, slowed down its performance and scalability as well as increasing the load on system resources. However, this complaint has to be weighed heavily against the advantages listed above many of which are not possible to achieve with the native code generated by C/C++ and Pascal compilers.

Other examples of Java advantages are:

Platform Independence: Not possible with C++.

Object-Oriented: C++ is not rigorously object-oriented (unlike Smalltalk for instance), rather it had a tacked on pseudo object-oriented nature.

Multi-Threading: This facility is extremely difficult to implement well in C++.

Ease of Use: Java is an easy to learn, user-friendly and powerful language.

> Network Supportive: Once again in C++, accessing objects that are not directly in the local address space is a difficult procedure. One of Java's key features from its inception is the ability to be used in a Client-Server fashion, and accordingly is imbued with powerful communication API's.

In an attempt to alleviate the inherent performance problems associated with processor-independent Java executables, various companies offer just-in-time (JIT) Java compilers that compile Java bytecode executables into native programs just before execution. This considerably speeds up performance and allows Java to approach the executing speeds reached by C++.

Performance Assessment: Java vs. C++.

How does the performance of Java applications compare with similar fully optimised C++ programs in theory, in benchmarks, and in real-world applications? The benchmarks listed below were taken from (URL1)

Program Speed

Many of the performance problems associated with Java are due to the way in which it is compiled. Java executables contain collections of platform-independent bytecodes, which cannot be run on a target platform without translation into binary instructions suitable for each target platform's CPU. The Virtual Machine is responsible for performing this translation. There are two possible methods a Virtual Machine uses to do so: a *bytecode interpreter* or a *just-in-time (JIT) compiler*.

Bytecode interpreters perform many times slower than comparable C++ programs because each bytecode instruction must be interpreted every time it is executed, which can lead to a great deal of unnecessary overhead. A JIT compiler however passes over the entire class file and then stores it rather than executing it line by line. This eliminates the need for repeated translations of each bytecode instruction.

In Table 3 the benchmark test comprises:

Integer and *Floating-point* division tests a loop 10 million times calling a member method, which contains an Integer or Floating-point division.

Dead code tests a loop10 million times and performs an operation that is never used.

Dead code with Integer division tests a loop 10 million times and performs an operation that is never used and one that is.

Static method tests a loop10 million times calling a static method, which contains an Integer division.

Member method tests a loop 10 million times calling a member method, which contains an Integer division.

Test	**Time (sec's) C++**	**Time(sec's) Java (JIT)**	**Time (sec's) Java (Bytecode interpreter)**
Integer division	1.8	1.8	4.8
Dead code	3.7	3.7	9.5
Dead code with Integer division	5.4	5.7	20
Floating-point division	1.6	1.6	8.7
Static method	1.8	1.8	6.0
Member method	1.8	1.8	10

Table 3 Java Vs. C++ Performance Figures

Table 3 gives a flavour (admittedly not a comprehensive one) of the type of performance expected of Java and C++ at compile time. Without a JIT, Java performs three or four times worse than C++. However, it is clear that with a JIT, in most cases, Java match's C++.

Code Optimisers

C++ compilers are able to improve the performance of a piece of code by detecting and improving inefficiencies through a process called *code optimisation*. The calculation of most optimisations requires knowledge about a group of instructions and may require multiple passes over these instructions. Thus the compiler may well on the first pass gather information on the structure of programmatic loops and note down any variables used. On the second pass it may attempt to streamline these loops and eliminate any unnecessary overhead.

A Virtual Machine without a JIT works on the fly, and thus only sees each instruction as it is executed, and so is unable to perform this type of optimisation. A JIT can, however, perform code optimisation on the entire class file. As a result, the only significant performance difference between a Java program run with a JIT and a native C++ application will be the amount of time it takes to perform the initial translation of the class file and the types of optimisation that are performed.

This overhead will only be a significant proportion of the total execution time if a program is composed of a large number of Java classes that are not used a significant number of times by a program. Real-world programs use the same classes many times, so the proportion of the amount of time spent translating the class will be very low compared to the time spent actually running the code within the class.

Program Size

Windows NT executables that are written in C++ are significantly larger than similar Java executables. There are two main factors that account for this size difference.

First, the binary executable format for C++ programs can inflate code by as much as a factor of two over Java code.

The Java virtual machine provides a set of packages that perform various functions such as network services and collection classes. Programs want to access these types of functions (that usually exist outside of the core C++ API) in C++ the programmer must deliver the implementation within the program. This typically doubles or triples the code size.

These factors can be seen in Table 4 below where the size difference can be accounted for by the extra libraries that are required for the C++ program to perform the equivalent operation.

Program Name	Program Size: C++ *vs.* Java
Simple Loop	46K *vs.* 3.9K
Memory Allocation	34K *vs.* 1.4K

Table 4: Java Vs. C++ Program Size

Memory Allocation.

C++ and Java allocate memory in much the same manner. However, C++ programs must explicitly release memory back to the system. One major cause of bugs in C++ programs is that programmers forget to explicitly release memory back to the system. This memory is "leaked" and will not be available until the program terminates.

In Java environments, the in-built *garbage collector* detects when a program no longer needs a piece of memory, and consequently releases it. Since the garbage collector in Java has to be able to determine which pieces of memory are no longer in use, the overhead of memory management for simple tasks is much greater in Java than C++. However, there are two advantages to the Java garbage-collection model that C++ lacks. First, programs are virtually immune to memory leaks. Since memory leaks are a frequently occurring bug in large-scale systems, this greatly reduces development time, since the programmer does not have to spend lengthy debugging sessions finding and rectifying such leaks.

Second, memory fragmentation can be a major problem in large-scale systems. Memory fragmentation occurs when large numbers of memory allocations are made and released and can seriously hinder the long-term performance of an application. A well-written Java garbage collector can move allocated memory around and prevent fragmentation.

So, although there is an acknowledged trade-off between Java and C++ regarding memory allocation, the trade-off results in great benefits for Java at the price of the performance lost.

For example:

Allocating and freeing 10 million 32-bit integers took 0.812 seconds in C++.

Allocating and freeing 10 million 32-bit integers took 1.592 seconds in Java.

Conclusions.

This section has shows the relative advantages and disadvantages of Java, and explained methods for minimising the disadvantages. The next section gives a closer examination of the strategies used to increase Java performance is examined.

Just-In-Time and Native Compilers

Introduction

In this section an examination is made of some of the technologies available to improve Java runtime performance. The technology to do this is can be split into two groups, *Native* or *Static* compilers and *Just-In-Time* compilers.

First an examination is made of what each technology actually does and a brief outline is given of how it works. Next, this chapter looks at and describes existing products on the market for each technology as described. Finally a conclusion is reached, and a verdict is given on the reviewed technologies.

Just-In-Time Compilers.

What is a Just-In-Time Compiler

The Java Virtual Machine (VM) is a software abstraction for a generic hardware platform and is the primary component of the Java system responsible for portability. The purpose of the VM is to allow Java programs to compile to a uniform executable Figure 24) execute within the VM itself, and the VM is responsible for managing all the details of actually carrying out platform-specific functions.

It is through the VM that executable Java bytecode classes are executed and ultimately routed to appropriate native system calls (see . A Java program executing within the VM is executed a bytecode at a time. With each bytecode instruction, one or more underlying native system calls may be made by the VM to achieve the desired result. In this way, the VM is completely responsible for handling the routing of generic Java bytecodes to appropriate native calls on the underlying platform. Knowing this, it is clear that the VM itself is *highly* platform dependent.

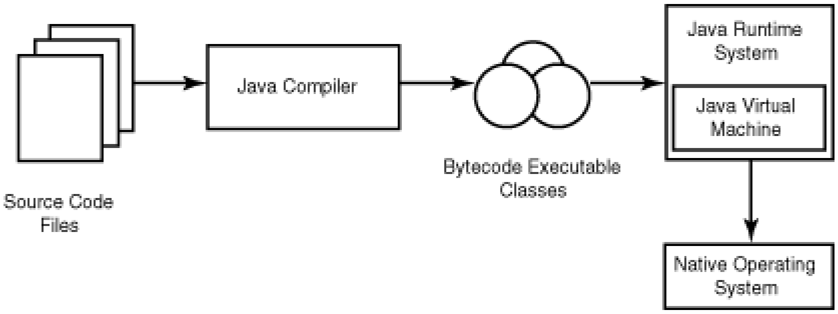


Figure 24: The Operation of the Java Virtual Machine

JIT compilers alter the role of the VM by *directly* compiling Java bytecode into native platform code, thereby relieving the VM of its need to manually call underlying native system services. By compiling bytecodes into native code, execution speed can be greatly improved because the native code can be executed directly on the underlying platform (see Figure 25).

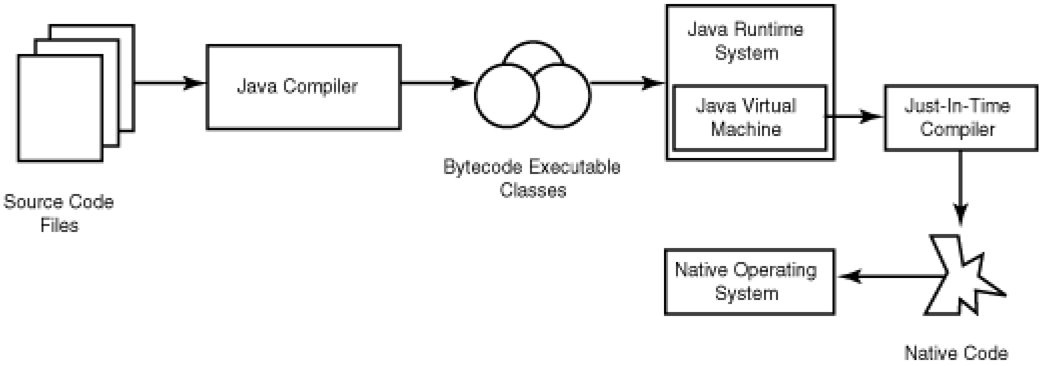


Figure 25: The Operation of the Just In Time Compiler

Notice that instead of the VM calling the underlying native operating system, it calls the JIT compiler. The JIT compiler in turn generates native code that can be passed on to the native operating system for execution.

Just-In-Time Compilers: The Reality

In theory there should be only a negligible difference between JIT-compiled Java bytecode and native C++. In practice, there are two factors that cause performance differences.

First, there will usually be several valid translations of platform-specific instructions when a bytecode instruction is translated into one or more platform-specific instructions. Each of these valid translations will produce the same result, but may have vastly different performance characteristics. If the programmers that create the JIT and C++ compiler are of the same calibre, the performance of both solutions should be similar.

Second, there is a significant trade-off between compilation time and the number or level of optimisations that are performed on a piece of code.

The common optimisations that compilers perform may be divided into groups based on performance gains and computational expense:

Primary and *secondary* optimisations typically afford a program 10 to 15 percent performance gains with minimal computational overhead.

Tertiary optimisations can add an additional 5 percent performance gain, but at much greater expense.

However, the current crops of JIT compilers are part of a rapidly maturing technology, and the balance between degree of optimisation and performance seems to have been well defined.

What is a Native/Static Compiler?

Java Native or Static compilers operate in a similar way to conventional C++ compilers. They operate upon the Java source bytecodes, and translate them into native platform dependant code. Unlike dynamic JIT compilers this happens "off-line" and outside of the Java VM. The compilation only happens once and produces system dependent executables. The generation of a native code program eliminates the performance and footprint overhead of a virtual machine.

The work necessary to translate bytecodes into an executable form is only performed once, before the program is delivered to its users. Because the translation from bytecodes to native code need not be accomplished in real time, computationally intensive, but very effective, optimisation techniques can be used to improve the quality and performance of the generated code. Whereas, JIT compilers can only perform a minimal set of optimisations due to runtime constraints.

Contrary to popular perception, deploying Java applications as native executables does not necessarily preclude the benefits of the "write-once-run-anywhere" paradigm. Most native compilers preserve the original Java class files, which can always be moved to a new platform at any given point. Needless to say, it is absolutely imperative to select a native compiler that not only supports the complete Java specification, but also does not require the developer to make any changes to the Java source code.

A Review Of Just In Time and Native Compilers

In this section an overview of current technology is presented and an assessment of their usefulness is made. Various assessment criteria were used:

Runtime Speed.

Ability to run on various platforms.

Dynamic Class loading ability.

Reputation and reliability.

Just-In-Time Compilers.

Symantec Corp. Just-In-Time Compiler.

The Symantec JIT (URL2) is widely regarded as the best on the market. In most current benchmarking exercises the Symantec JITs has been found to offer the best all round performance. It is so effective that Sun has now bundled it free with it's latest Runtime Environment and is included in the new JDK1.3. This decision by Sun has underlined the importance of this VM.

Conclusion

A fine stopgap solution when assessing initial performance in the prototyping stage as it is free and readily available. Symantec has more expertise in JIT technology than any other vendor in the Java market. Its technology has steadily improved and matured, and it offers a stable, well tested and well used technology.

SunSoft's HOTSPOT Compiler

The only real competitor, in performance terms, to Symantec's JIT is the HOTSPOT (URL3) optimiser/JIT from Sun. It delivers Native Compiler performance with all the other advantages of JIT technology. Fundamentally, HotSpot is an extension of JIT compiler technology, and uses a technique called *adaptive optimisation* to generate the performance boost. The HotSpot VM constantly monitors the performance of the executing bytecode on a per-method basis, and strives to identify critical regions within the application. Once this is done, Java methods in the critical region are "in-lined" (statically included directly within the code) and optimised for maximum performance.

Because HotSpot learns from the execution of an application at runtime, its speed really picks up after it has been running for a few minutes. It also analyses the application's environment; optimising for a multiple-processor system differently than it would for single processor systems. Because it is more sensitive to the underlying machine, HotSpot could produce code that is more optimised for a specific machine than a C++ program compiled specifically for an particular Operating System.

HotSpot also features an advanced garbage collector and a revolutionary thread synchronisation implementation that significantly improves the performance of multithreaded applications.

Conclusion

HotSpot almost matches anything produced by C++, it should be highly reliable as it operates within Sun's own VM. In addition, many of the performance problems encountered within agent projects are due to the multi-threading aspects inherent when creating agents, which HotSpot addresses using its threading technology.

Native/Static Compilers.

IBM High Performance Compiler for Java

The HPJ Native compiler (URL4) is one of the best known and respected native compilers available, and is packaged as part of IBM's Visual Age's Java Development Environment. The following benchmark test (Table 5)was taken from PC Week (URL5) for the following applications written in Java.

Jobe: A Java obfuscator, by Eron Jokipii.

Toba: A Java to C translator, University of Arizona.

Javac: Sun's Java 1.0.2 source to bytecode compiler.

Benchmark	Interpreted	JIT	Compiled	Compiled (no check)
Javac	40.2s	21.1s	3.9s	3.7s
Toba	67.2s	51.7s	5.7s	5.5s
Jobe	18.1s	13.6s	4.6s	4.2s

Table 5: Performance Figures for HPJ Compiler

Conclusion.

Although it is fast, HPJ has significant limitations. IBM's online white paper describes the beta version as "a Java 1.0.2-level implementation with some missing features". Unfortunately, these missing features are critical to any agent project that interacts with a user as they include the java.awt(Java's graphics API) and some of Java's basic facilities, such as text-mode output from the println. Also absent from the HPJ beta version is one of Java's vital features for software life cycle cost reduction: dynamic classes. It does not appear that IBM has created any further versions of HPJ, and the beta version is rather old now.

Tower J High Speed Native Compiler,

Tower Technologies is currently the leading player in the field of specialised native Java compilers. Their TowerJ (URL6) Java compiler not only makes use of sophisticated global optimisation, but also structural and flow analysis to deliver C++ execution speed to Java applications. Additionally, TowerJ claims significantly to reduce the memory footprint of Java applications.

TowerJ employs a unique hybrid architecture that potentially solves many of the problems associated with native compilers. It allows users to choose at deployment time which portions of a system are to be compiled directly to native shared libraries and which will remain as bytecode until runtime. This architecture could potentially offer an agent developer excellent performance whilst still maintaining flexibility.

The TowerJ compiler was an early entrant into the native compiler field, the compiler is not only Windows compliant (most competitors are *only* windows compliant), but also can operate on Solaris and many other platforms.

The new release of TowerJ also adds a partitioning option to Tower's whole system optimising native compiler allowing developers to create optimised native DLL's from Java bytecode classes. Thus TowerJ should allow for dynamic functions such as Remote Method Invocation (RMI).

In terms of performance it by far the best product on the market. In a recent benchmarking exercise it set a new record among Java VM and deployment compiler vendors.

Conclusion.

It must be expected that in the near future other technologies will emerge that might match TowerJ's performance, however at this time, this is the best on the market.

JOVE

JOVE (URL7) is a "whole-program" optimising compiler offered by *Instantiations Corp.* It builds a model of an entire Java program including any standard or third party class libraries that are used by the program. JOVE uses this model to analyse the structure of the entire program before performing optimisations or code generations.

Jove is unique in that it not only performs standard optimisation techniques but also applies specialised optimisations, which work on Java's inherent object-oriented inefficiencies.

In addition, unlike most standard native compilers Jove fully supports many of Java's dynamic features such as dynamic object creation, garbage collection, reflection, Beans and RMI. Additionally, it includes a high-performance runtime environment, which packages applications as single executable files.

JOVE claims that:

JOVE removes over 90% of the dynamic calls in a typical Java application

In addition, it produces a 50-75% overall reduction in call sites

JOVE aggressively seeks out and eliminates unused code and unnecessary generalisation in programs

JOVEs precise, multi-generational garbage collection system outperforms competitive systems by a factor of up to 15 times. All while maintaining or reducing the runtime footprint of the program

In addition JOVE's runtime system includes:

Low memory overhead objects.

Precise, multi-generational, multi-threaded garbage collection

Low overhead polymorphism

Native multi-threading or single threaded

Fast method dispatch and type checks

J/Direct native method support

Conclusion

Jove appears to be significant quicker than all current JIT and Native compilers, the only real problem lies in the fact that it produces Native code.

Future Options: Jove vs. HotSpot.

These two technologies offer the agent developer the brightest hopes of best performance whilst still maintaining Java's advantages.

Hotspot attempts to extend JIT (Just-In-Time compiler) technology to include aggressive optimisations. The JIT approach is inherently limited by the requirement that it perform its optimisations in real time as the program is executing.

JOVE, sometimes called an "*ahead of time compiler*", does its optimisations during the deployment phase of a project when developers are preparing the application for implementation on the target system. Because of this, JOVE, unlike Hotspot is not time-constrained with respect to the optimisations that it can use. Relatively time-expensive, but very effective techniques can, and are, used to produce the highest performance possible.

JOVE has several major advantages over Sun's Hotspot technology:

It produces significantly higher performance programs

Optimisations are not limited by real-time requirements

JOVE produces significantly smaller runtime footprint for applications

It provides the familiar "executable file" deployment model

JOVE will support both JNI and Microsoft's J/Direct extensions

However these performance gains apply mostly to applications that are largely static in nature. Whereas in programs with a degree of dynamism HotSpot is likely to offer a better approach. So in the case of our system, it would appear Jove would offer the better performance gains.

JITCache Technology: Red Shift.

JITCache technology works by observing the code that goes through the compiler as it is compiled (like any normal JIT). However once a method is compiled a copy is stored, so that the next time that method is loaded the JITCache recalls it from the store without having to recompile. If a class file changes, its methods do get recompiled and the old code in the database is thrown away.

JITCache technology can be combined with native, ahead-of-time compilation to pre-load the cache with methods that will be used in a deployed environment. The result: the speed of a natively compiled application and the flexibility of a Just-in-Time compiler.

Redshift (URL8) make uses of JITCache technologies and is a hybrid approach with the advantages of both JIT and native compilers. It is however intended for use with embedded systems. Its approach to increasing performance is novel though, and is a promising avenue for future tools.

Conclusion

It can be seen that, in theory at least, of the current crop of Java performance tools Redshift should offer the best performance. However, at the time of writing the author was unable to retrieve any performance figures from the makers, and is therefore unable to make any judgement on its real performance. In addition there appears to be no testimonial evidence of any real applications that use this technology.

Conclusion on JITs and Natives

Performance issues pertaining to Java should not present a problem when developing a multi-agent adaptive system, with each passing month performance barriers are being broken with SunSoft alone having boosted its Virtual Machine tenfold in a matter of months.

It is clear that each technology has its relative advantages and disadvantages. JIT offers much improved performance with fully dynamic performance, and offers developers all of Java's inherent advantages. However, JIT leaves large memory overhead and it's performance in terms of speed is a *far* below that of native compilers. JIT compilers (including HotSpot) are expected to continue their dominance at the client, but it seems that in all likelihood, native compilers are here to stay. Companies will find it impossible to ignore the prospect of running their natively compiled Java applications many times faster than by using JITs alone. Thus a split can be seen in technologies where JITs are employed at the Client end, and Native compilers are in use on the server.

To summarise JIT offers:

Portability: Fulfils all of Java's inherent portability aspects.

Security: It generally is not possible to analyse a program's machine code before execution and determine whether it does anything malicious. Tricks like writing self-modifying code mean that the malicious operations may not even *exist* until later. But Java byte code was designed for this kind of validation: it does not have the instructions a malicious programmer would use to hide their assault.

Dynamic Loading: A dynamic Java application can continue to load new classes during execution. However when a native static compiler turns a set of Java classes into a single executable, only those predetermined classes are available when the program is run. This dynamic capability is particularly useful for modifying the behaviour of an application while it is running, rather than having to recompile or reboot before the changes take effect.

It is clear then that JIT offers many important features key of which is dynamic loading, of classes using RMI, which could well be a useful mode of inter-agent communication. Once again though, JIT performance just isn't quick enough for use in a real-time system (Perhaps HotSpot could disprove this stance when it arrives).

The relative advantages and disadvantages of each technology can be summarised in Table 6.

Compiling pros and cons		
Product	**Pros**	**Cons**
Virtual machine interpreter	Full awareness of object properties	Slow run-time performance
JIT (just-in-time) compiler	Object awareness plus faster execution	Longer load times due to compilation, optimisations limited by need to minimise start-up delay
Static compiler	Fastest possible execution due to extensive (but one-time) optimisations	Large executable files, platform specificity, difficulties in using updated modules without recompilation

Table 6: Compiling Pros and Cons

Ideally the agent developer requires an environment with the speed of native compilation, but with the dynamic features that a JIT provides. This is a difficult problem and one that has no clear solution at this time. However, the available technologies will improve as the market for native Java compilers is only now developing and should witness severe competition in the near future. Native compilers should show some sophistication in performing the global optimisations but also support provided for key Java features like dynamic class loading, multithreading, and security.

Conclusion

At this moment in time it appears that TowerJ is the leading candidate within the optimisation technology field, as it offers the best possible speed whilst maintaining many critical aspects lost on other compilers such as Dynamic Class loading. HotSpot offers major performance increases to any project that involves a large number of threads. HotSpot offers improved thread performance in two key areas: *garbage collection* and *synchronisation.* Both are critical to most agent projects, but synchronisation problems could be significant, especially at the GUI where threads are in contention, not only for CPU attention, but also for access to the graphics on-screen which must be synchronised in order for the threads draw.

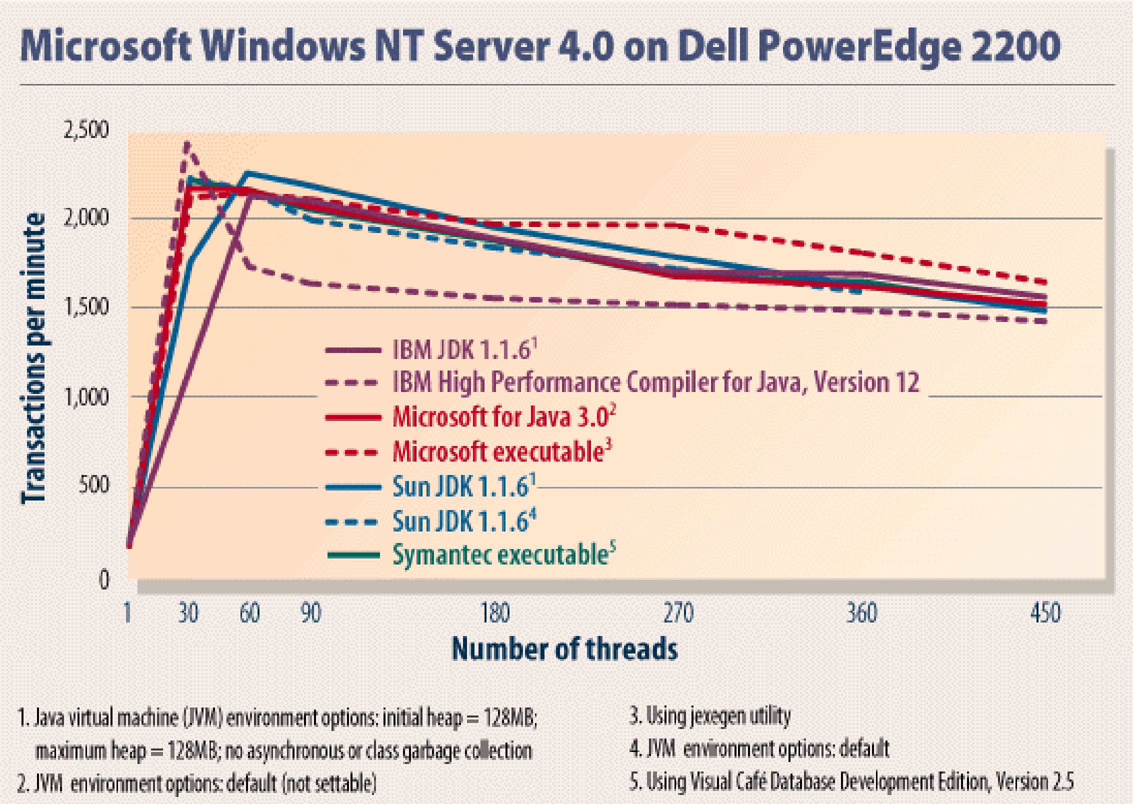
Other Performance Issues:

The Threading Problem

What is the problem with Threads?

It is envisaged that any typical multi-agent project will encounter performance problems not only due to Java's inherent problems, but also due to the potentially large number of threads in use and the apparent load this will place on the Java VM. However, in Figure 26 (URL9) it can be seen that most Java VM's can handle large number of thread without highly significant performance loss. (For detailed figures used in Figure 26 see Appendix B)

It may appear from Figure 26 that Native compilers offer no greater performance gains than dynamic compilers. However, Figure 26 only includes Symantec's crude native compiler (without optimisation) and IBM's HPC, which is less than perfect. It fails to include the performance leaders such *TowerJ*, *Supercede*, or the best of breed JIT compiler by Symantec. It does however offer us a general flavour of Java performance with a large number of threads.


Microsoft Windows NT Server 4.0 on Dell PowerEdge 2200

1. Java virtual machine (JVM) environment options: initial heap = 128MB; maximum heap = 128MB; no asynchronous or class garbage collection
2. JVM environment options: default (not settable)
3. Using jexegen utility
4. JVM environment options: default
5. Using Visual Café Database Development Edition, Version 2.5

Figure 26 Java Thread Performance

Influence Of Choice Of Platform On Thread Performance.

Thread performance is highly dependent on the choice of platform used at runtime. Under Windows 95 and NT, it is now routine for Java threads to be run as native Win32 operating system threads that, in theory, should be scheduled by the Operating System itself. In a test by PC *World* (URL10) much better results were achieved across the board under Windows NT 4.0 than Windows 95. Windows NT offers more robust scheduling and synchronisation than does Windows 95.

The reason for this is Windows 95's dubious Win16Mutex. Win16Mutex is a mechanism for ensuring that only one thread in the system at a time enters the non-re-entrant 16-bit portions of Windows 95's kernel. Under Windows 95, the threads fall completely out of synch with one another, some ending long before others do. Under Windows NT, the 16 threads all proceed in lockstep and finish in synch.

The author has compared a Windows NT system with a Solaris Spark system and found that it is 4 times faster.

Conclusion.

For optimum performance one should use **Windows NT4.0** on a **Pentium processor** of a least 350Mhz.

Which Virtual Machine is appropriate?

For Speed.

Table 7 shows some benchmarking results using the near industry standard *Volcano Mark* tests (URL11). The TowerJ compile beats all comers including the ultra-fast Microsoft VM and the new JDK1.2, which include the Symantec JIT. TowerJ, however, is not free, and could be potentially be bettered by Jove and HotSpot.

Java virtual machine	Scores	Average (best 2 of 3)
Tower TowerJ 2.1.2	1715, 1761, 1749	1755
Microsoft SDK 3.0 P1	1398, 1408, 1414	1411
Novell JDK 1.1.5	1319, 1325, 1320	1323
JavaSoft JDK 1.2	1260, 1234, 1260	1260
IBM JDK 1.1.6	1217, 1214, 1207	1216
JavaSoft JDK 1.1.6	1119, 1117, 1111	1118
Microsoft SDK 2.02	1109, 1108, 1109	1109
SunSoft JDK 1.2 Dev 3	839, 837, 838	839
SunSoft JDK 1.1.5	546, 548, 546	547
Apple MRJ 2.0	319, 319, 323	321
Linux JDK 1.1.6	230, 234, 233	234
FreeBSD JDK 1.1.5	175, 175, 174	175

Table 7: Comparison of Virtual Machines

For GUIs.

The new JDK1.3 offers an ultra-stable and reasonably quick Virtual Machine. In addition it includes *Swing* libraries, which are a radical rewrite of the widget set in AWT. *Swing* offers widgets which are double-buffered, all can have tooltips, are extensible, track the tab key for focus, support keyboard shortcuts ("accelerators"), and are internationalisable. Buttons and Labels can contain icons, built from GIF files, in any orientation. JPanels can have standard borders. JMenus or JMenuBars can be added to any container. In addition the new Swing widgets are not *slow* (They operate quicker than Windows 98 system menus.)

It also contains several other important features such as:

JavaIDL is now a core package, and JDK 1.2 contains a pure Java ORB. That means any Java VM can act as a CORBA client or a CORBA server, and any Java object can become accessible through CORBA.

RMI—Custom sockets can be used, sending RMI requests over SSL or IIOP. Also, a remote object can remain dormant until created by a client request.

Transactions—Java now supports an implementation of the OMG/CORBA Object Transaction Service (OTS/JTS) with an alternate Java API on top (JTA). This allows a single transaction to comprise actions occurring on multiple VMs and multiple databases; all those actions will either succeed or fail as one.

JDBC—Database access has been improved, with support for scrollable and updateable result sets, batch updates, connection pooling, rowsets (sort of a Bean-enabled database view), distributed transactions, extra data types, and so on.

Conclusion.

For development purposes TowerJ is recommended. However, for prototyping purposes the JDK1.3 VM is highly recommended. It is quick, and although it is not as quick as the Microsoft SDK VM, it includes several in-built features that allow it to better use the new API's in the JDK1.3 libraries. In addition, HotSpot is only usable with JDK1.3, thus when it arrives seamless integration with existing development environments is possible.

Integrated Development Environments (IDE)

Various industry standard IDE's were examined and assessed. It was found when examining the RAD approach (Rapid Application Development) to GUI building, that a great deal of the IDE's tested utilised the component-event and AWT aspects of JDK1.3. However, now JDK1.3 is in existence it is to be expected that most application developers will use Sun's improved graphical widget set "Swing" instead of AWT. Many IDEs were examined and only Borland's JBuilder had full support for Swing. Visual J++ was excluded from assessment because J++ is tailored towards applications that are to be deployed on only on the Windows platform. To this end it does not provide an open enough environment for developers producing agents in an heterogeneous environment, and its proprietary use of WFC (Windows Foundation Classes) is not necessary now Swing has arrived. A more detailed review of available IDE's along with conclusions can be found in Appendix A.

Multi-Agent Toolkits.

Criteria

There are a number of commercial/research agent development toolkits in the public domain, with each catering to a certain type of system. In the context of this thesis, the toolkits examined were largely KQML compliant and offered a lightweight communications platform suitable for a distributed system. It was found that a large number of the toolkits on the market cater for developers seeking to produce mobile solutions. These were, inherently, unsuitable for multi-agent static systems, although some did provide good communication facilities.

It is reasonably difficult at this stage to select *the* most appropriate toolkit, as each toolkit has been designed for a certain architectures/paradigm. Further, each toolkit is designed to be used not only within a certain conceptual architecture, but also with one of several pragmatic solutions to the implementation of this architecture. Thus it is fairly hard to select the best solution if one does not yet know the problem. Until the agent designer has rigorously defined a conceptual architecture, it becomes difficult to select a toolkit that is tailored to the individual problem.

To give an example of this problem, if the agent designer decides to have a central router running (Agent Name Server, ORB, and Kernel) to which all the agents are attached and communicate through, then toolkits like JATLite and Via are appropriate. If, however, they intend their agents to have some inherent reflection mechanism, in which the agents have an internal model of the agent system and know exactly which agent they intend to communicate with directly, then toolkits such as JAFMAS, KAFKA or Voyager are appropriate.

Thus the approach taken here was to examine the most appropriate general toolkits on the market and assess their architectural independent features. The following are some of the major requirement assessments:

Lightweight Communications Platform: To remove some of the overhead associated with developing distributed applications.

KQML compliant: Useful if toolkit supports some form of agent communication language such as KQML.

Speed: The delivery of intra-agent messages should be timely

Ease of Use: Agent developers need a toolkit that is intuitive to use and that requires the least amount of time to learn.

Future Integration: Should try to be FIPA (Federation of Intelligent Physical Agents – agent standard) compliant.

A full review of the multi-agent toolkits examined can be found in *Appendix B*.

Conclusion

The selection of the tools used by an agent designer is of vital importance as they help define the system architecture. This section has outlined the problem of Java performance and has recommended the use of the TowerJ compiler to increase performance to an acceptable standard.

Development times can be reduced by the use of a suitable IDE. This chapter has concluded that JBuilder 2 offers the best functionality and development tools. Further, it appears to be the only IDE that supports JDK1.3 libraries such as *Swing*, which is of key importance to GUI implementation.

Lastly an examination of agent toolkits was made. Once again, as in the case of the Java compilers, it is rather difficult to select the most appropriate toolkit as new tools appear on a frequent basis. Each toolkit offers functionality that best suits a certain type of agent architecture. For instance toolkits such as *Concordia* and *Grasshopper* and *Aglets* are suitable for use with Mobile agent applications.

With this in mind it seems best to utilise the best aspects of various toolkits, each which offers a functionality most appropriate to an agent projects general requirements. Thus for an agent environment for use in a real time system, as described in this thesis, the recommendation of this chapter is to use *Voyager* as the platform at the transport level. Voyager allowed the core architecture to work with third party APIs, because it can utilise existing objects without having to change them to exist in a distributed system..

One might argue that Voyager does not offer KQML compliance, however it is the opinion of this author, after much consideration, that KQML compliance is not essential. KQML's forte is in providing a simple transport syntax - an interlingua - for heterogeneous systems, with each agent querying others for services. In a closed system there is no real need for the syntactic form of KQML: in tightly defined closed agent system one can simply encode messages in objects and focus on what to do with them, not on how to represent them. For instance there will be little use in a closed system for the *advertise, broadcast, broker, recommend* etc. type performatives. All agents in a closed system should know exactly which other agent they intend to communicate with. Thus the use of KQML may very well be restricted to *ask-one* and *tell* type performatives, which are easy enough to encode without third party libraries.

Thus the tools used in the construction of the AMEBICA agent architecture will be:

1: Jdk1.2 beta version:

2: TowerJ

3: JBuilder 2

4: Voyager

5: Windows NT4.0

This chapter has shown what tools are needed to construct the adaptive system, the next chapter details the conceptual architecture, and the fundamental tenets on which it was built.

Chapter 8

CONCEPTUAL ARCHITECTURE

Introduction

Having decided to adopt a multi-agent approach in the design of an adaptive presentation system, the next step is to design a multi-agent architecture to support an implementation using the chosen software development environment. Any chosen architectural approach needs to fit into existing process control environments and be applicable across a number of domains. This chapter therefore discusses the proposed architecture and the reasons for any imposed constraints.

Conceptual Design

At the heart of the design is the notion of a multi-agent information presentation system that is capable of adapting the information presented to the operators, dependent upon the context of the state of the operators and of the process. The presentation system (which includes both visual and auditory elements) will automatically adapt the presented information in order to enable the operators to attain their current goals in the most effective manner.

A number of general architectural constraints have been adopted in the design (Khalil 1999a), as presented in the following sections.

Design Principles Adopted In AMEBICA

The Stream Concept

Process values arrive in continuous streams and these are rendered at the interface usually in some standard graphical form (for example as a P & I Diagram). The usual graphical (or auditory) form at the interface is often constrained by relevant industry standards. At any instant in time, therefore, the interface will consists of a set of representation objects continuously being updated by the process.

It is a key aspect of the design of the architecture of the AMEBICA adaptive system that it does not operate on these data streams as they pass through to the rendering system, but operates via its reasoning processes only on the set of representations at the interface. In other words, adaptation occurs only at the rendering interface, not in the streams themselves.

This indirect coupling approach eliminates many problems that would otherwise occur – particularly time constraints. Because of the stream approach, the operator receives information on-the-fly in a timely manner, only then does AMEBICA modify the renderings as a result of its reasoning processes. The concept is illustrated in (Figure 27). Adaptation only occurs after the Process Model Agent has decided that adaptation may be required. This is signalled to the AMEBICA system, which eventually adjusts the rendering

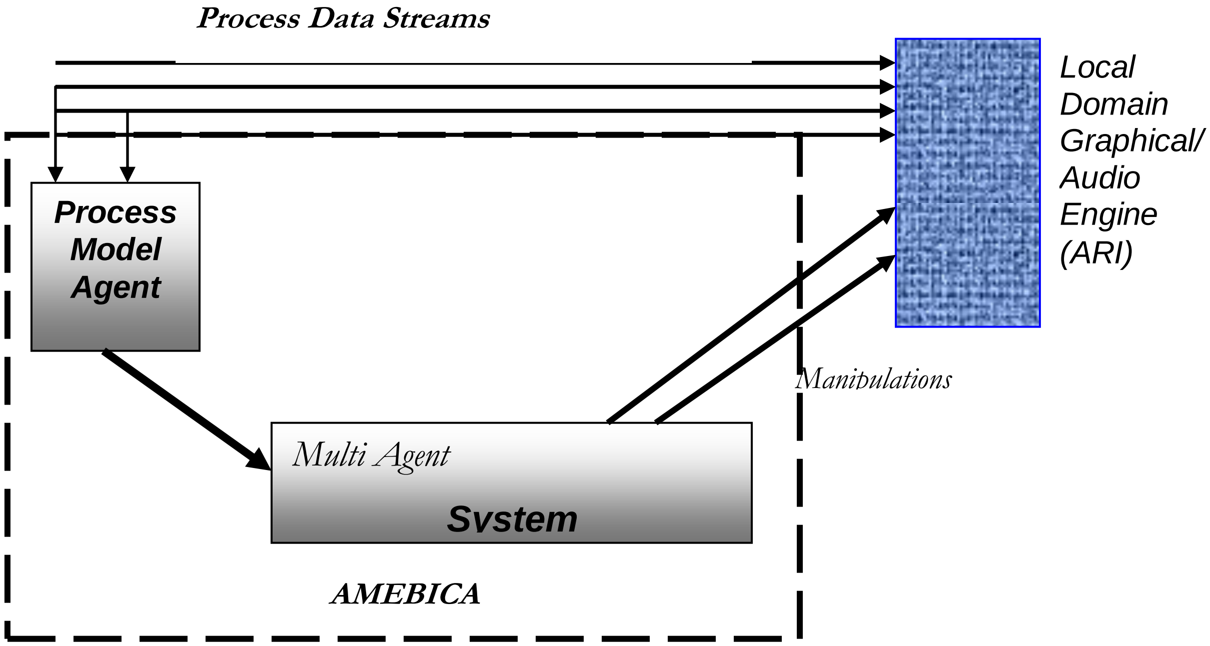


Figure 27 Stream Architecture

The Generality Principle

A second constraint on the architectural design is to make the AMEBICA system, as far as possible, a generic adaptation system that maps events of discrete levels of significance - at the input - to appropriate rendering characteristics at the output. To achieve this, no direct process knowledge is embedded within AMEBICA, rather, AMEBICA has two domain dependent interfaces – the Process Model Agent and the Abstract Rendering Interface. These two interfaces allow it to be domain relevant. The Operator Agent informs the Process Model Agent about operator interactions and the Process Model Agent is continuously monitoring the process output streams. Thus the Process Model Agent can identify domain specific occurrences in the operator or system environments and can send triggers to AMEBICA for adaptation. The Abstract Rendering Interface takes general adaptation commands from AMEBICA and renders them in domain dependent representations (Khalil 1999b).

AMEBICA therefore operates in a similar manner to a Java class file in that such a class file is generic and applicable among many systems. However, to make this possible, the class file requires a Virtual Machine (VM) that is highly platform dependent. The VM operates as a translator, changing generic Java calls to system-dependant calls. The Process Model and the Abstract Rendering Interface are the AMEBICA equivalent of the VM. Thus if an alarm of relatively high importance occurs from a non-critical sub-system, the Process Model might translate this to a low priority AMEBICA alarm. This is then processed and the resultant output of the AMEBICA system is passed to the Abstract Rendering Interface, which realises the rendering at the interface as a domain specific alarm. Thus to apply AMEBICA in different processing domains, an appropriate Process Model Agent and Abstract Rendering Interface need to be constructed.

Internally, therefore, AMEBICA operates in a domain independent way passing information in what we call "AMEBICA-speak". The process model converts from "process-speak" (the real process world) to "AMEBICA-speak", and eventually AMEBICA commands (in AMEBICA-speak) are converted back into domain dependent representations through the Abstract Rendering Interface.

Spatial Adaptation Principle

There is some empirical evidence that operators have certain cognitive traits that are resistant to change, and hence are particularly important for adaptive systems. These traits affect the way in which an operator interacts with an interface and the requirements they need of an interface (Van der Veer 1990)

If the operator cannot cognitively adapt to interactions within the interface, then the characteristics that prevent them doing so are suitable for system adaptation (Van der Veer 1990). Spatial ability is one such characteristic and one for which AMEBICA aims to provide adaptation, especially in terms of making it easier for the operator to realise where salient information is. However it is also important that during adaptation, the spatial placement of information should be maintained as far as possible. Thus we have bounded adaptation so that adaptation usually occurs before information is placed on the screen. Generally speaking, once upon the screen, information should not be moved. AMEBICA, will place relevant information as near as it can to other information of that type, it will also try to select the best representation for that information in context (see flexible mapping), and use the best representation parameters (size, colour, behaviour and so on).

Other general adaptation principles we have tried to follow are:

> *All rules should be simple/straightforward*: Complex rules often lead to unexpected emergent behaviour, usually when separate rules are contradicting each other. One example of a straightforward rule is giving certain general process events a priority. These priorities are used later in the adaptation framework to help organise how AMEBICA decides which renderings it should alter and which it should not. It makes this decision based on the relative importance of each rendering, assigned to it by the Process Model Agent. Thus, a simple integer priority is assigned for each type of event. So because alarms are of greater importance than measurements, the following priorities are assigned for events of these types: MEASUREMENT = 5, ALARM =8. The Process Model Agent also assigns sub-priorities, which are specific to a type of event. For instance a *temperature* alarm may be far more important alarm than a *maintenance valve* alarm, so: temperature=9, maintenance=9. In this way the system determines not only the general importance of an event (alarm say) but also its relative importance when compared with events of the same type. So it might decide it can shrink a rendering that has priority 5,7 (in other words: measurement=5, priority =7), over one that is 8,9 (alarm=8, priority =9)
>
> *Limiting screen Adaptation:* it is clear we do not want AMEBICA to adapt at the wrong time, or to make too many adaptations of the wrong sort. These unwanted adaptations, reduce user trust in the system, and thus make the system less efficient and liked.
>
> *Operator Induced Renderings Should Not be Adapted:* If the operator has requested a specific representation then that representation should not be altered until the operator has

finished with it. AMEBICA must not alter it once it is on screen and whilst the user is working on it.

Active moving of windows is limited. The operator has a spatial understanding of where unserviced renderings are. If windows are moved around the screen it disrupts operator effectiveness by altering their spatial map.

Window re-sizing allowed: Re-sizing a window only partially affects the operators spatial map, the operator still maintains an understanding of the renderings location, even if its dimension has changed.

Changing existing representations for alternative should only be used in limited circumstances. Changing a representation when it is already being rendered at the interface can disrupt the consistency of interface and therefore operator effectiveness. An operator does not wish a rendering they are using, to change its representation form during manipulation. Doing so may cause the operator to believe it to be a completely new rendering. However the *parameters* of that rendering (colour etc) CAN be changed. Alternative representations may be used when the system informs the operator of an imminent change (in an attempt to draw their attention to the representation perhaps).

Adaptation Principles - Flexible Mappings and On-the-Fly Adaptation

In traditional interfaces, a mapping is made at *design time* between the process parameters and appropriate renderings at the interface. This mapping is usually the best all-purpose mapping under a set of general constraints. Once selected, this mapping is then rigidly applied independent of context even though in real life, mappings can change with context.

There are two alternative approaches to implementing adaptation. Firstly, the designer can provide alternative representations at design time. The adaptation process is then concerned with decisions between the provided alternatives. We call this approach a *Flexible Mapping Approach*.

Secondly, Artificial Intelligence techniques could be applied (using a Knowledge Base of appropriate representation types and rules for generating them) such that representations would be generated at run-time. This latter approach we call *On-the-Fly generation*. Figure 28 illustrates both approaches to adaptation. The latter approach is often difficult to implement because of a lack of appropriate knowledge about the effectiveness of particular presentation mechanisms in particular contexts.

In AMEBICA both approaches are used. The flexible mapping approach is used when there is incomplete knowledge about alternative representations and their usefulness. In such cases, the expertise of the interface designer is used to provide alternatives and guidelines for their use. The on-the-fly generative approach is used for application areas where reasonable knowledge about possible representations and rules for their manipulation exist. A good example is space or screen management. In this case there are well-known rules about cluttering, zooming, translation and

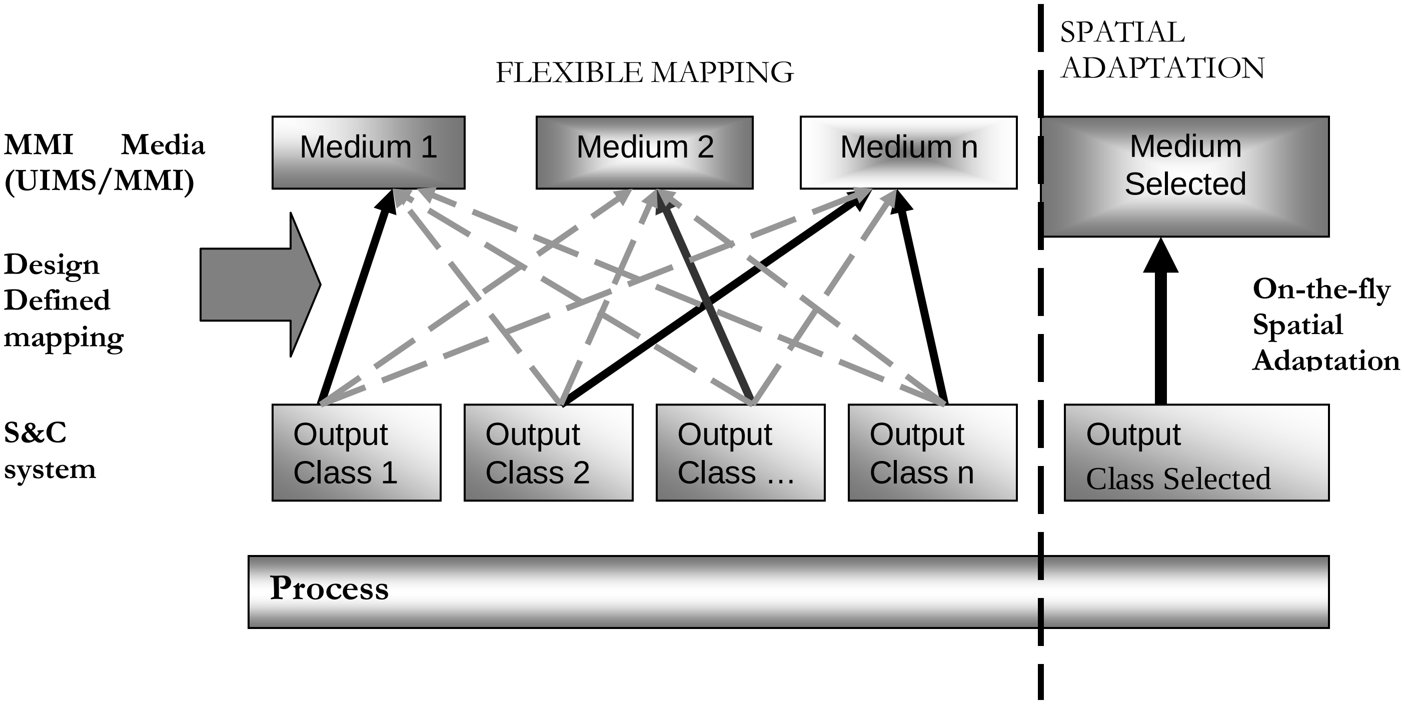


Figure 28 Flexible Mapping

highlighting.

The Agents and their Functions

Stream Rendering – The Role of the Media Agent

Once the Process Model Agent has identified that a stream needs to be adapted, the responsibility for adapting the streams of data from the process to the operators is vested in a set of Media Agents. Each Media Agent (Figure 29) is responsible for a set of substreams of the total process stream and renders them according to instructions from the main AMEBICA system. An appropriate set of components of the stream is defined as a logically related domain set. For example, in Electrical Network Management, there already exist well-defined collections of objects (e.g. Transformer sub-stations, High level Transmission Networks), which always exist together and are driven by a set of data values. Graphical representations also already exist for these as industry standards. Initially therefore, the Media Agent will have these representations as a default, and this set will be loaded when the system is initiated. So at initiation, the Media Agent implements a default system. However, the Media agent will also have access to a set of alternative representations for these streams (the Flexible mapping set) to enable adaptation to take place when required.

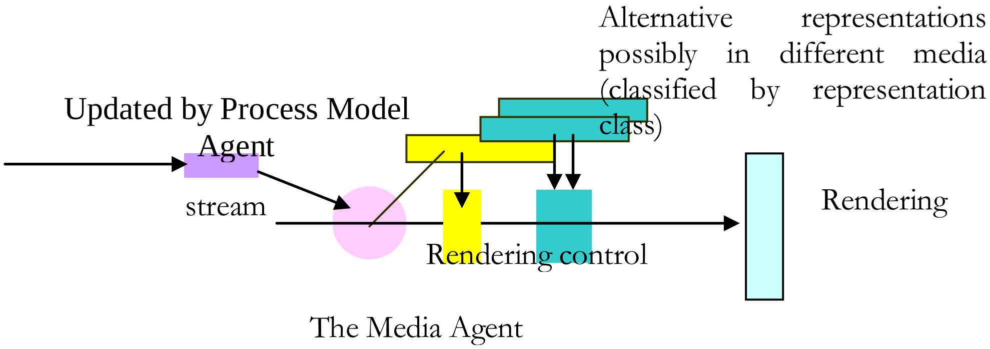

Alternative representations possibly in different media (classified by representation class)
Updated by Process Model Agent
stream
Rendering
Rendering control
The Media Agent

Figure 29 Media Agent Streaming

Adaptation occurs in AMEBICA when the Media Agents are directed to alter the nature of the rendering by other agents in the system. The basic source of all adaptation triggering is the Process Model Agent, which initiates adaptation as a result of a defined set of conditions in the process/operator input or both. As a result a Media Agent will receive notification of an adaptation requirement from the Process Model Agent and then it must negotiate with the rest of the AMEBICA system to decide how these special conditions will be implemented and, after negotiating with other AMEBICA agents, allocates a particular rendering mechanism for it.

For example, the Process Model Agent might detect that a temperature value in a particular set of stream components has exceeded a defined limit. It notifies the appropriate Media Agent controlling this sub-stream, that special notification status (this is an example of AMEBICA-speak) now exists for the temperature rendering. The Media Agent negotiates with the rest of the system, and eventually is told (say) to colour the value in red. It then modifies the parameters of the graphical rendering to achieve this.

All Media Agents have at their disposal a number of pre-defined rendering representations for a particular sub-stream (Graphics, Text, Audio) and each of these media has a set of pre-allocated possible representations, one of which will be a default. The Media Agent decides upon a new rendering (with help from other agents) and manipulates the rendering control for each medium. Thus, the position of a graphic may be changed, the sound of an Audio channel may be altered, or an entirely new representation loaded.

Each Media Agent therefore has access to a large set of Graphics, Text, Voice, Sound, Animated Diagrams or Video media, there may also be special effect Media such as "Fish Eye" views.

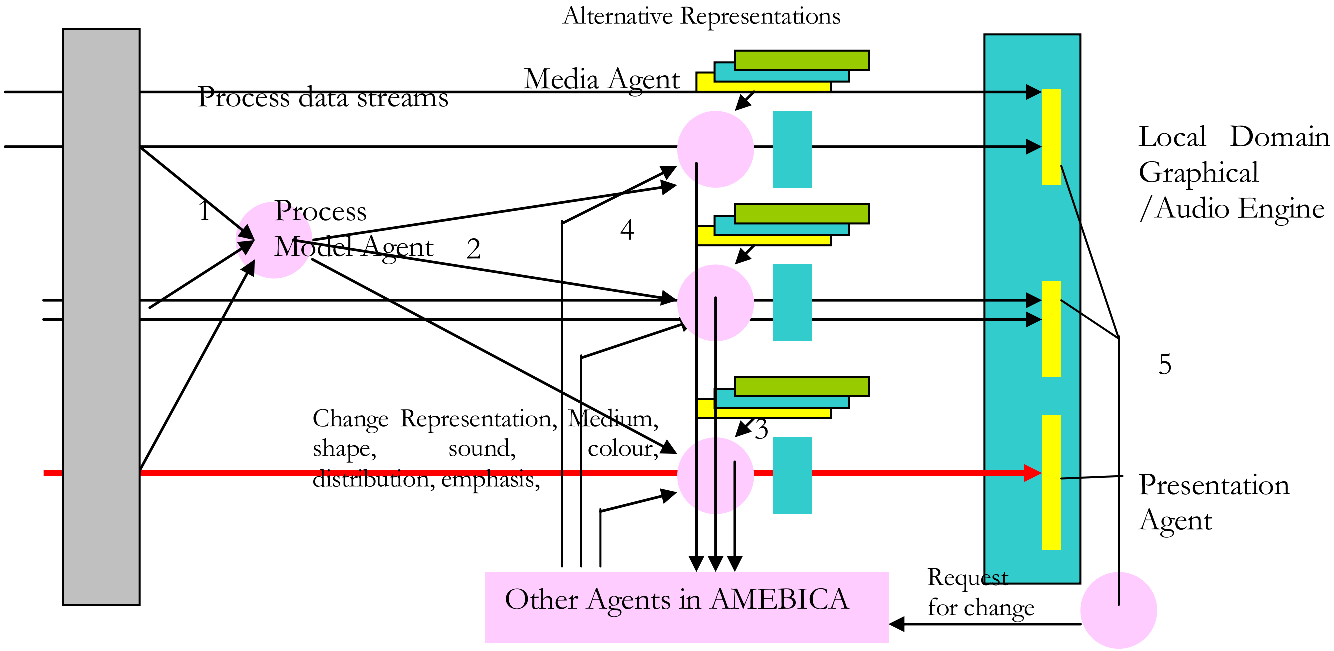

Alternative Representations
Media Agent
Process data streams
Local Domain Graphical /Audio Engine
1
Process Model Agent
2
4
5
Change Representation, Medium, shape, sound, colour, distribution, emphasis,
3
Presentation Agent
Other Agents in AMEBICA
Request for change

Figure 30 The Agent Streaming Architecture

In addition, a Presentation Agent, monitors current usage of the rendering facilities at the interface and offers summary information about unused rendering resources to the main AMEBICA system. In doing this, the Presentation Agent offers on-the-fly generation facilities. The information it provides can result in modifications to the spatial layout characteristics (zooming, translation, uncluttering etc.)

The way in which Media Agents operate is illustrated in Figure 30 above:

1. Data is currently being rendered as a set of sub-streams in some manner by a variety of Media Agents. The Process Model Agent detects conditions in some sub-stream that requires special action (the decision may also depend upon operator conditions as well).

2. The Process Model Agent communicates this in a domain free form (AMEBICA-speak) to the Media Agent involved.

3. The Media Agent continues rendering as before, but requests assistance from the AMEBICA system to decide on a new rendering from its set.

4. The AMEBICA system eventually decides on a new rendering and communicates this to the Media Agent.

5. The Media Agent then modifies the rendering at the interface and adaptation is complete. The Presentation Agent updates its view of the usage of resources at the interface and offers new configurations to the system. It also provides a history of media usage that will be later used to maintain consistency. The Presentation Agent also provides the other AMEBICA agents with information about Resource Usage in the Interface.

In the above viewpoint, the Process Model Agent, the Media Agents and the Presentation Agent are mainly reactive in nature. The Process Model Agent reacts to the current state of the streams and informs the Media Agents of special requirements. The Presentation Agent reacts to changes in the screen and audio resources used, and informs the main AMEBICA system of the current state of renderings. The Media Agents react to instructions from AMEBICA to adjust the renderings or to changes suggested by the Process Model Agent.

In a future architecture these agents could be more deliberative, but in the first version of the architecture they are largely reactive. There are other agents in AMEBICA that are, from the outset, more deliberative in nature and these will be described shortly.

The Role of Media Agents

One difficulty with *Media Agents* has been defining their exact role. They are, of course, responsible for passing streams to an appropriate rendering at the interface, but a number of issues were examined in deciding at what level of granularity they should operate.

> Are the *Media Agents* hardwired to particular streams, which they permanently represent?
>
> Is there a *Media Agent* for every stream?
>
> Is there a *Media Agent* for every type of event? (alarm)
>
> How does a *Media Agent* operate on non-stream variables?
>
> If a *Media Agent* is representing a schematic diagram representation, composed of a variety of different types of renderings, how should they be decomposed? Should other Media Agents be instantiated dynamically, or is the decomposition within the existing *Media Agent?*

These are some of the questions that were argued over in the initial implementation. The final approach adopted was as follows:

> Each *Media Agent* should represent an *event*.
>
> An *event* is an occurrence at the *Process Model Agent* that has been stipulated at design time, as worthy of its own *Media Agent*.
>
> This means that ANY *event* at the process that – at design time – had several possible mappings that could represent it, must be associated with a Media Agent.

From these decisions we can formally define some terms:

EVENT: An occurrence in the system either by the Operator or in the Process that warrants the system undertaking some action. Generally, if an Event warrants the system action, it is the type of situation, which at design time would warrant the designer having several different forms of representation to represent it. So for instance an alarm condition in the process is a situation that would require system adaptation. It is also the sort of situation where at design time the designer might create several different ways of showing an alarm situation. An Event is the initiator of a Media Agent (which contains all the representations for that event).

An Event can not only be a direct occurrence in the Process that warrants adaptation, it can also take more abstract forms. Later in the chapter *Measurement* and *Schematic* events are used as examples. These simply mean that some initiating event in the System has warranted the system displaying a Measurement or a Schematic type representation. So, if the Process Model has been constructed at design time to display a Measurement Representation of a Valve, if the pressure in a certain Valve in the Process has moved into a pre-defined range, the AMEBICA system would see this as a Measurement Event. If several sub-stations in an Electricity Grid reach the same voltage level, the Process Model Agent might issue a Schematic Event, ensuring the system displays a Schematic of these substations, whose representation will be dictated by the system context. Also, if an Alarm is being rendered at the interface, and has not been serviced for a set amount of time, the system might issue an *Alarm Event* to the Media Agent controlling that alarm, stating that it has not been serviced. Finally, Events can be operator derived, so if an operator wishes to view a Measurement value, they will issue a Measurement Event from the interface.

REPRESENTATION: The actual composition displayed at the interface for a particular event. Typically a Representation is a higher level view of an event, so the representations for an alarm might be audible tone, flashing symbol, text, dynamic moving slider whose position represents the severity of the alarm. A Representation can often be decomposed into smaller sub-parts. For example, a schematic diagram might have a boiler (which can have temperature and pressure sub representations) and a flow line with a valve, each of which are sub-components, which can be rendered in different ways. These sub-components are called *Rendering Objects*.

RENDERING OBJECT: A single entity, typically part of a representation, but which can be displayed independent of an initial representation. For instance a DIODE Rendering Object might be part of a circuit diagram representation, but may also be viewed independently of the circuit diagram.

RENDERINGS: These are the physical manifestation of the conceptual classes of Representations and Rendering Objects. A Rendering is the actual interface element that is displayed and that can be clicked on by the Operator to request further functions.

An Example of Media Agent Decomposition

To give an example using the terms given above, there might exist a *Schematic Media Agent*, used to render a schematic diagram. This diagram will consist of a number of elements that are driven by events. At design time, several possible ways may be defined for representing how this schematic can be presented at the interface. For instance, one representation could be a *grid format.* This might take the form of a typical 2-D lined grid layout with transformers at set places on an electricity grid. This representation has two types of Rendering Object – the grid and the transformers on it as can be seen in Figure 31.

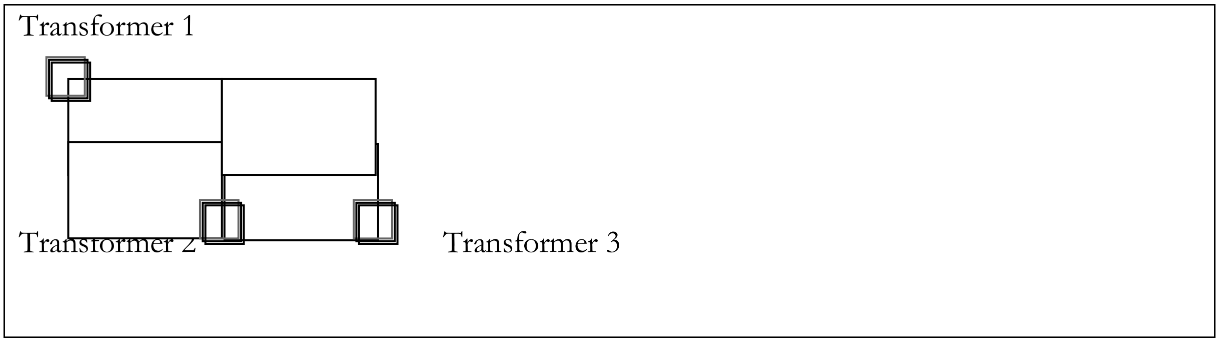


Figure 31*: A SCHEMATIC, with representation type GRID*

Another type of representation might be a *Table Format* representation, which contains within it a simple list, that's shows the relevant values for each transformer. (Table 8).

Table 8 A SCHEMATIC, with representation type LIST

GRID LOCATION	TRANSFORMER NUMBER	NAME
0.2	*1*	*Willowby One*
1,0	*2*	*Acacia Avenue*
2,0	*3*	*Station Rd.*

When AMEBICA is running, only ONE of the representations can be selected at any one time for the Schematic Media Agent. It is the job of other agents in AMEBICA to advise the Media Agent on the MOST appropriate representation to display.

Each representation is composed of one or more *Rendering Objects*. For example in Figure 31 a *GRID* representation can be seen which consists of a **grid** *Rendering Object* and three **transformer** *Rendering Objects*. Each of these *Rendering Objects* can be displayed independently and *represented* in different ways. Thus if the operator wishes to check on the value of the voltage of an individual transformer, they might select the transformer they want to examine and change its type of Rendering Object from a Transformer *symbol* to a Transformer *measurement.*

To operate correctly it is necessary for the system to know how different representations or different Rendering Objects are related together. Thus each Media Agent contains within it a Media Selection Table. This table contains a pre-ordered list of representations deemed at design time to be suitable for that particular type of Media Agent (so the Schematic Media Agent would contain a table stipulating available representations as *Grid Format* and *Table* Format representations). The order the list is set depends on the current Process Conditions. So if the Process Conditions are normal it might set the highest-ranking representation as *Grid Format* since it allows the operator to easily understand the topological relations between the transformers. If however there is a disturbance and it is critical that the operator views the current voltage levels the *Table Format* representation might be ranked highest.

To relate which representation is most appropriate for the process conditions the Media Agent must contain meta knowledge which details this relationship. These are contained within an objected called *Representation Data.* So the *Table Format* representation would have meta-data contained within it that stipulates that it should be ordered higher than a *Grid Format* representation, *IF* process conditions are disturbed.

Each *Media Agent* also contains meta data on the relationships that the TYPE of *Media Agent/Representation/Rendering Object* can have with other types of *Media Agent/Representation/Rendering Object*. For instance a:

RENDERING OBJECT (TRANSFORMER) of **TYPE**: Symbol, Within a **REPRESENTATION** of type GRID FORMAT Of **EVENT**: SCHEMATIC

Could be DISPLAYED as a

RENDERING OBJECT of type HORIZONTAL SLIDER, Within a **REPRESENTATION** of type GRID FORMAT of **EVENT:** MEASUREMENT, by a *Measurement Media Agent*.

By this we mean that a transformer symbol Rendering Object on a schematic diagram could be clicked on and it would change to a Horizontal Slider measurement Rendering Object. So instead of seeing a static symbol representing the Transformer we would see a dynamic measurement Rendering Object which represents the values of the voltage running through that transformer, such as those seen in Figure 32. The difference is that the *Schematic Media Agent* is generating the TRANSFORMER symbol, and *a Measurement Media Agent* will generate the appropriate measurement representation (i.e. a Slider). A Transformer can therefore be represented by a set of different Rendering Objects such as those in Figure 32.

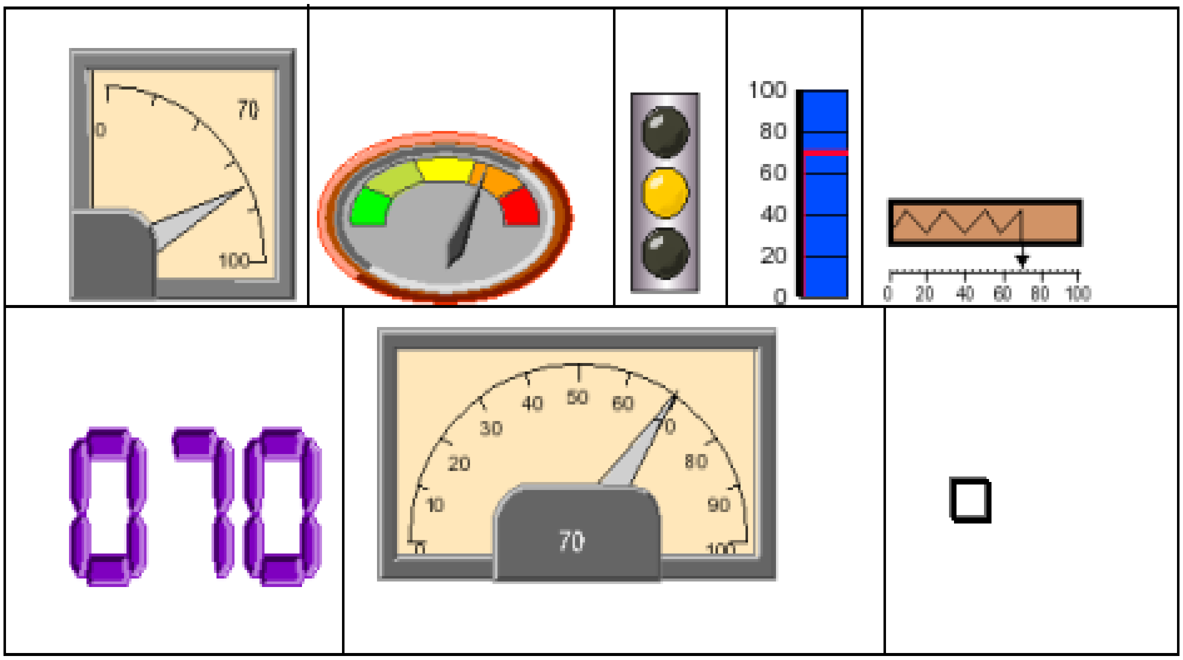


Figure 32 Different Available Representations For A Transformer

Thus a Transformer has a variety of different forms it can take within a Measurement Media Agent. The choice of which Rendering Object the Media Agent does use can be made directly by the operator or adaptively selected by the AMEBICA framework. Each *Rendering Object* that can take on more than one form contains within it a list of all its *allowable* transformations. So a TRANSFORMER Rendering Object will contain within it the knowledge that it can be viewed as either as a *Measurement* or as a *Symbol*. This sort of knowledge will allow AMEBICA to know which *Media Agent* to use for rendering alternate views.

An Example Case

The scenario described here expands on the example given earlier of a Transformer changing form from a symbol to a measurement. The operator would click the right mouse button over the Transformer symbol. Right clicking over the object accesses the relationship table within the *Schematic Media Agent* and displays all the different forms that the Transformer can be displayed as. The operator can then either SELECT whatever Rendering Object they desire OR let AMEBICA chose an appropriate Rendering Object.

The way in which the two approaches are implemented is as follows:

a) AMEBICA adaptation to an operator request

The operator is currently viewing a Schematic diagram generated by the *Schematic Media Agent* and wishes to observe the measurement values of gas passing through a valve Rendering Object on the schematic.

To do this the operator moves the mouse pointer over the appropriate valve and presses the *right mouse button*.

The mouse button click causes AMEBICA to check the relationship table for the VALVE Rendering Object. The table tells AMEBICA that the valve can be displayed as a MEASUREMENT or as a SYMBOL, and generates a list of the allowable event types (Measurements and Symbols) and displays it over the mouse pointer to the operator. The operator passes the mouse over the Measurement option and selects it. The operator then releases the right button. The Measurement event in the representation table stipulates that it requires a *Measurement Media Agent* to display it.

The *Operator Agent* takes this information and passes it to the *Process Model Agent*. These instruction contain a TAG indicating that the data came from the user and other associated information

RENDERING CATEGORY: Measurement, **FORM SELECTED:** Not Stipulated, **RENDERING OBJECT:** Valve Number: **111 RENDERED CURRENTLY BY:** *Schematic Media Agent* **. FROM:** Operator

The *Process Model Agent* receives the operator request and, if not already in existence, launches a *Measurement Media Agent* to display the request.

The *Process Model Agent then* passes the TAG information on to the *Measurement Media Agent*. Since the user has not stipulated WHICH exact presentation form of Measurement they require (Slider, Dial, Text Values for instance) AMEBICA must decide upon the most appropriate form for the circumstances.

The normal cycle of the *Media agent* contacting the *multi-agent* system begins and appropriate representation and parameter information is eventually selected and returned to the *Measurement Media Agent*.

The *Measurement Media Agent* finally renders the AMEBICA selected representation of the VALVE Rendering Object.

b) Non-AMEBICA Adaptation to a User Request

The operator selects the VALVE, and not only selects MEASUREMENT, but also specifies the TYPE of Rendering Object they require, say a DIAL.

This is done by the operator clicking the RIGHT mouse button over the Valve symbol, which creates a menu with the allowable options for that symbol (Symbol and Measurement).

The operator scrolls down to the Measurement option, which displays a sub-menu showing the available Rendering Options available for that event (much like navigating a bookmark menu), and the operator selects the DIAL representation

This information is passed to the PROCESS MODEL AGENT by the OPERATOR agent and is TAGGED as being of type:

RENDERING CATEGORY: Measurement, **FORM SELECTED:** Dial, **RENDERING OBJECT:** Valve Number: **111 RENDERED CURRENTLY BY:** *Schematic Media Agent* . **FROM:** Operator

Because the operator has chosen this, AMEBICA's selection of the best REPRESENTATION is not required and the relevant *Measurement Media Agent* is launched and the Rendering Object for DIAL is selected.

As the USER selects an event and representation then there is no need to contact the multi-agent system. Thus the *Media Agent* can display the operator chosen representation immediately in the default location.

The Process Model Agent

The Process Model Agent connects the real domain dependent world of the process with the "pure" rendering world of the adaptive system. Once this transition has been defined we need only discuss the AMEBICA world, represented as the multi-agent system. Eventually, the output of the multi agent system manipulates the local graphics, video, and audio systems and since these systems deal with "real" domain objects the manipulations are relevant.

Some readers may find this idea of Domain Independence for AMEBICA hard to understand. To understand the idea, consider a Pharmacist in the High Street who develops photographs for clients (the example is a good one because they are "rendering" data for clients). How a photograph is developed will depend on the use for which it was made. For example, if a client wants a Passport Photograph, it will need to be Black and White and 2.5cm square. A wedding photograph will be larger , and in colour. The counter assistants, who talk with the customer will, like our Process Agents, convert the domain knowledge (Passport Photo) to pure photographic knowledge (B&W, 2.5cm square). When the processing actually takes place, the domain knowledge is forgotten and all reasoning takes place in the Photographic domain.

The process model acts on streams of data that come from the process. Each stream is grouped with other related streams, and their values monitored. The Process Model Agent (Process Model Agent) has pre-set trigger values for each of these logically grouped streams. If the current stream value exceeds these pre-set values then the Process Model Agent may instantiate an appropriate Media Agent to adapt a rendering. Depending on the nature of the stream, it sets an Adaptive System variable called *Evidence Levels.* The severity of the Evidence levels (an equivalent of priority) determines the nature of the representation that will eventually be chosen. So if a stream representing the temperature in a boiler reaches a critical point, the Process Model Agent might set the domain independent variable *Evidence Level* to (say) a high level of 8 for the request to the Media Agent responsible for displaying the temperature.

The Process Model Agent (Process Model Agent) is therefore customized at design time and populated with pre-defined rules for triggering adaptation in incoming streams. When the conditions for these rules be are met, the relevant data streams are grouped and sent as a logical stream to the appropriate Media Agent. The Process Model Agent also has *rendering categories* linked to an event. Thus for an alarm event (that is the Process Model Agent has found conditions occurring on certain input streams that match conditions dictated by the user customised *rules.*), there are several rendering categories such as Measurement, Alarm etc. Therefore the Process Model Agent utilises *logical rules* for the inspection of the state of the process/alarms and, based on the results, **manipulates the data streams** in order to change the associate rendering category when particular emphasis must be given to a particular stream.

As an example of this, consider a situation where a schematic layout of a sub-system is currently displayed on screen, and is being handled by a *Schematic Media Agent*. Then a situation occurs whereby one data stream feeding a boiler symbol on the schematic exceeds a rule-value threshold in the Process Model Agent (for instance the value for temperature has exceeded a pre-defined limit), and its *rendering category* can now be classed as alarm.

This situation would need to be highlighted on the schematic diagram AND additionally may also require the display of a separate alarm window containing temperature and pressure information. In our scenario, a rule in the Process Model Agent would be triggered and would send information on which boiler has gone into an alarm state to the relevant *Schematic Media Agent*. In this case the adaptation would not occur WITHIN the representation of that schematic. The Media Agent for that schematic would simply look up an internal table to determine what it needs to do when one of the Rendering Objects within it reached an ALARM condition. This lookup it may result in the boiler symbol being switched to RED and caused to flash.

A second alarm window would be created if the Process Model Agent determines from the rule firing that an *alarm media agent* was also required. The Process Model Agent would then instantiate a new *alarm media agent*, and would tap off a separate data stream to the one feeding the *schematic media agent* and pass it to the *alarm media agent*. The *alarm media agent* contains within it knowledge of all the possible representations of an event of type "alarm" and will pass this information to other agents in AMEBICA, for them to select the most appropriate representation.

Data Streams

The Process Model Agent divides input data into Streams. Each stream represents a set of information that should be rendered using the same rendering resource.

Each Media Agent is in charge of the management of one stream. It controls the resource switch to redirect the stream to the proper rendering resource.

What kind of criteria may be used by the Process Model Agent to create streams? Three simple types of mapping methods could be used in AMEBICA:

Static per-component mapping

Static per-media mapping

Dynamic on filter basis mapping

Static per-component mapping

This solution consists in having a stream, and a Media Agent, for each component that the alarms/events/measure may belong to. If, for example, designers want to group signals on geographical basis they must define a different stream for each geographical area present in the system.

This results in having a great number of streams and Media Agents running at any one time. It is reasonable to assume that the number of possible streams can become a great deal larger than the amount of rendering resources that may be active at a given moment. Therefore, this solution implies a considerable waste of system resources. Moreover this method is inflexible, as the designer has to strictly define all the possible sets of aggregation in the configuration phase.

Static per-media mapping

This solution consists in having a stream and a Media Agent for every rendering resource defined by the user. This implies that the user has to detect all the possible significant situations during the configuration phase.

As in the scenario mentioned above, this also results in having a great number of streams and Media Agents running at any given moment, regardless of whether the predefined rendering resources are available or not at that moment. This method of stream generation may involve resources wastage if the user defines many rendering resources. On the other hand, if the number of rendering resources is quite small, this solution becomes too rigid a representation of the system status. In fact, in the case of a disturbed status, where many alarms arise on a certain system component, if the user has not previously defined a specific rendering resource for such a component, this quite serious situation cannot be exploited by the adaptive system due to lack of system flexibility to display it in a highlighted way.

Dynamic on filter basis mapping

This type of mapping method does not involve a rigid definition of all the possible streams in the adaptive system rather it involves the dynamic creation of the data streams and their related Media Agents. The user, during customisation phase, may define several rules, based on predefined or customised system data. Using such rules the Process Model Agent can filter the process input data to obtain a set of different streams.

For example the user may define a rule to create a stream if more than five alarms arise on the system components belonging to the same geographical area. Using this rule a stream is created only if in that moment it is really necessary for information rendering, and it is dynamically destroyed when it is no longer needed, therefore ensuring efficient usage of resources. The user can define many different types of grouping criteria, to cover all their specific needs, using logical rules to decide what kind of information to render depending on the process status and incoming alarms/events/measures.

Approach comparison

All three approaches described can be used for mapping process data to Process Model Agent streams. The first two are based on static mapping criteria while the last adopts a parametric generation approach. In this case streams are generated at run-time on the basis of triggering rules introduced during customisation.

Static approaches require defining all streams and Media Agents at design time. A number of Media Agents are always running even when they have no role in the representation. A fixed association of plant component and streams / Media Agents is both impractical and inflexible. These types of mappings seem suitable only for simple applications with small numbers of streams, components and Rendering Objects.

A dynamic approach is needed to match the requirements of a complex application with large numbers of potential input configurations. In this case the resources actually allocated to the system should be optimised; and a static approach cannot do that.

Rendering Category

As *streams* represent WHAT a designer wants to be rendered, and *evidence levels* their priority, so *rendering categories* determine HOW designers want the rendering to be displayed. When the Process Model Agent receives a signal, it uses logical rules to reach two goals:

Find out all the streams the signal belongs to

Determine the rendering categories of each stream

Each stream is managed by a Media Agent. The Media Agent receives the *rendering category* for the stream from the Process Model Agent and then must only use an appropriate rendering.

Rendering categories represent the degree of evidence and importance of the stream for representation. They are a conceptually independent of the process knowledge but are the result of the process-dependent reasoning of the Process Model Agent. When a process data stream carries important information it needs to be highlighted (i.e. a process issue) and it therefore has to be given a higher evidence rendering category (i.e. an AMEBICA issue).

Rendering categories are associated with integers; the higher the integer, the higher the evidence level of the stream. The Media Agent contains a table, which may be customised by the user, containing, for each rendering category, a list of all the resource types suitable for stream rendering, ordered by preference criteria. This table may be different for each type of Media Agent, and therefore for each different type of stream.

The Process Media Agent can, therefore, assign a rendering category to a certain stream to define the level of evidence the information should have, regardless of the specific type of media the Media Agent will actually use for rendering.

Overview of the AMEBICA Reasoning Process

The Media Agents actually instruct the Rendering System to display a particular representation. To do this, however, they need advice from other agents in AMEBICA. There are two agents that reason about rendering issues and enable instructions to be given to the Media Agents to allow them to make a choice between representations. These are the *Rendering Resolution Agent* and the *Media Allocator Agent*. The job of the Rendering Resolution Agent is to interpret a request for change from a Media Agent and provide an ordered set of high level information representation classes for further processing. This set is then passed to the Media Allocator Agent, which is responsible for taking the final decisions about representation taking into account spatial factors and constraints at the actual interface. It can do this because it is also provided with relevant spatial information by another agent in the AMEBICA system – the *Presentation Agent*. The Media Allocator Agent will try to fit the best representation suggested by the Rendering Resolution

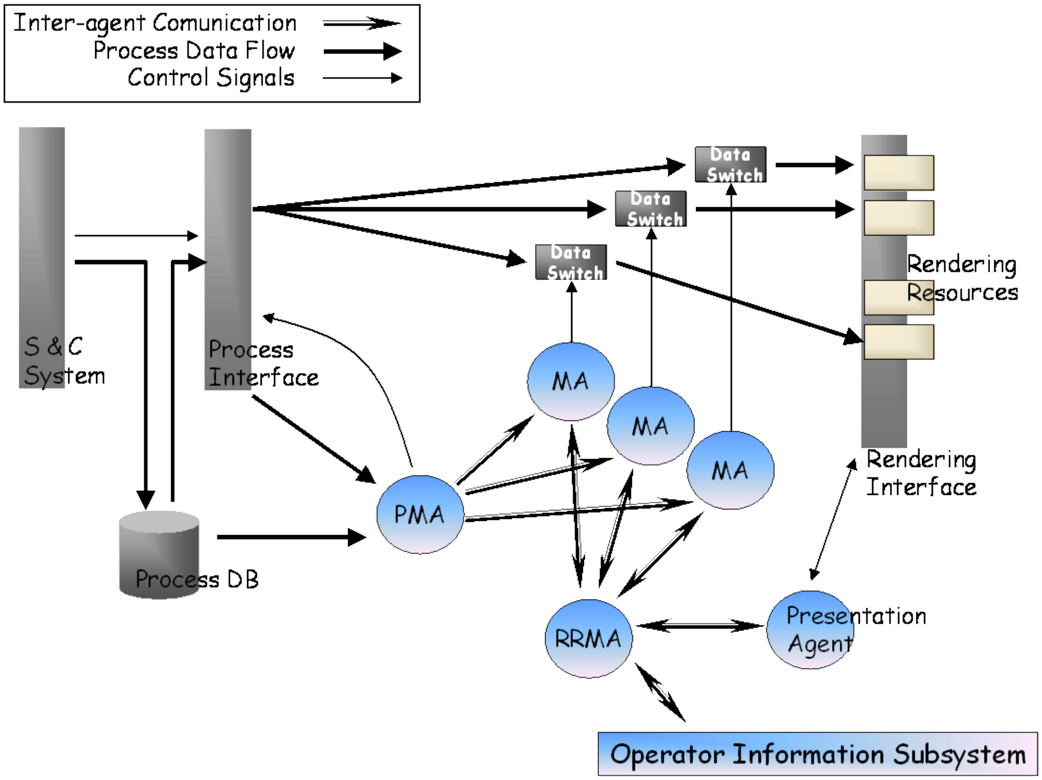


Figure 33 Overall System Context

Agent into the rendering interface. If this cannot be done, it will try to reorganize the output renderings in some way.

Rendering Resolution Agent (RRA)

The task of the Rendering Resolution Agent (as seen in Figure 33) is to determine an ordered list of acceptable representation classes when requested to do so by a Media Agent. It arrives at the ordered list after consultation with two other knowledge sources in AMEBICA – the *Human Factors Database* and the previously mentioned *Operator Agent*. The Rendering Resolution Agent's goal is to gather information from the Operator and Media Agents and send context knowledge (including a list of appropriate representations) based on this information to the Human Factors Rules Database. The Human Factors Rules Database uses its knowledge of the representations and the context information (including implicit process knowledge passed on by the Media Agent) to return a possibly re-ordered list of representations along with some parameter information. Such parameter information might include colours of components or background, sizes, use of text and so forth. It passes its priority ordered list to the Media Allocator Agent, which compares the list against available interface resources (provided by the Presentation Agent) and makes a final decision. Subsequent negotiation may be required if the Media Allocator Agent cannot find a candidate within the list which meets the resource constraints.

To understand its process it is necessary to understand how the other actors in the reasoning process operate.

The Operator Agent

The Operator Agent plays several roles. It monitors mouse and keyboard clicks and any interaction the operator has with the system. From this, it attempts to deduce, what state the operator is in. This information is then passed on to both the Process Model Agent and the Rendering Resolution Agent. The Process Model Agent uses the information to make adaptation decisions. The Operator Agent also logs all user requested events that pass through it. It then passes this information to the Rendering Resolution Agent, which may append it to the Representation List passed on to the Media Allocator Agent.

The Human Factors Rules Data Base

It is not possible to apply domain independent HCI rules to select the best type of representation without implicit knowledge of the context. It is therefore necessary at design time for the designers to indicate the constraints that make one type of representation better than another and in what type of situation. From this, a list of possible representations is generated for each event situation, and these are stored within the Media Agent.. This might appear to contradict the *generic* nature of the system, but this is not true, as the system needs to be *customized* before it is ready for deployment.

On initiation the Media Agent takes the requirements from the process, accesses its Representation Table and extracts a pre-set (design time) list of appropriate representations that are rendered. This pre-ordered list is then passed (via the Rendering Resolution Agent) to the Human Factors Rules Database that takes into account operator conditions and uses the generic HCI rules to alter the parameters of the representations (such as size, colour, content) and the order. It is to be expected that in some cases this list will be a set of representations that *all* are suitable and are *all* of equal priority. The Human Factors Rules Database will then re-order the list based on its context knowledge and HCI rules. For instance for a measurement value, the Human Factors Rules Database might re-order the list such that the highest order representation is *text* rather than a graphical medium. It may do this because it deduces that the current value is within certain boundaries, such that the operator requires high accuracy in the information. Text will provide this accuracy in comparison to (say) a Graphical Slider giving a better view of the overall value in the context of several boundaries. Context is therefore very important and the Human Factors Rules Database can only select the most appropriate representation and parameters when it understands the full context (including the state of the operator and the event history).

The initial set of representations are chosen and their default order is determined by *domain dependent* rules customized within the Media Agent, and selected according to the nature of the event the Media Agent receives. The Media Agent, therefore, delivers an ordered Representation

List to the Human Factors Rules Database (via the Rendering Resolution Agent). Once the order has been set, the *domain independent HCI rules* can re-order the list, and set the display parameters for each representation (size, colour, positioning, flash, media). The advantage of this approach over placing all representation choices within the Human Factors Rules Database, is predominantly one of speed. Because there are multiple instances of Media Agents each selecting its own pre-ordered list, a great deal of the workload for selection is taken off the Rendering Resolution agent system. This is an important factor when alarm flooding is taking place.

The Media Allocator Agent

The Media Allocator Agent's job is to work out how to actually fit a representation into the rendering system. It receives an ordered list of acceptable representation classes from the Rendering Resolution Agent. It will obtain the appropriate list of representations for these classes from the Media Agent, It will request information about current resource usage (visual and aural) from the Presentation Agent, and will try to develop a solution for the most appropriate representation class. In doing this it may have to modify other Media Agent representations in order to spatially accommodate the new representation. The Media Allocator Agent has several strategies to deal with the spatial adaptation problem. If it can find the best representation in an existing space, it does so. Otherwise, it will try and reduce the size of existing renderings to expand a large enough space for the current representation to be displayed. If it cannot expand the free space, it may *overlap* existing representations, dependent on their respective priorities. If the requesting representation cannot be placed using *overlap/expansion* techniques, the Media Allocator Agent has the power to move existing representations, or queue the representation (assuming it has a low enough a priority) until space is available on screen. If the problem is too difficult to solve it may send a request for a revised list from the Rendering Resolution Agent.

The Presentation Agent

The Presentation Agent has a continuously updated view of resource usage on the interface for all media. It also keeps a historical record of past representations and media use. It provides the Media Allocator Agent with under utilisation data and over utilisation aspects.

A Simplified Example Set of Interactions

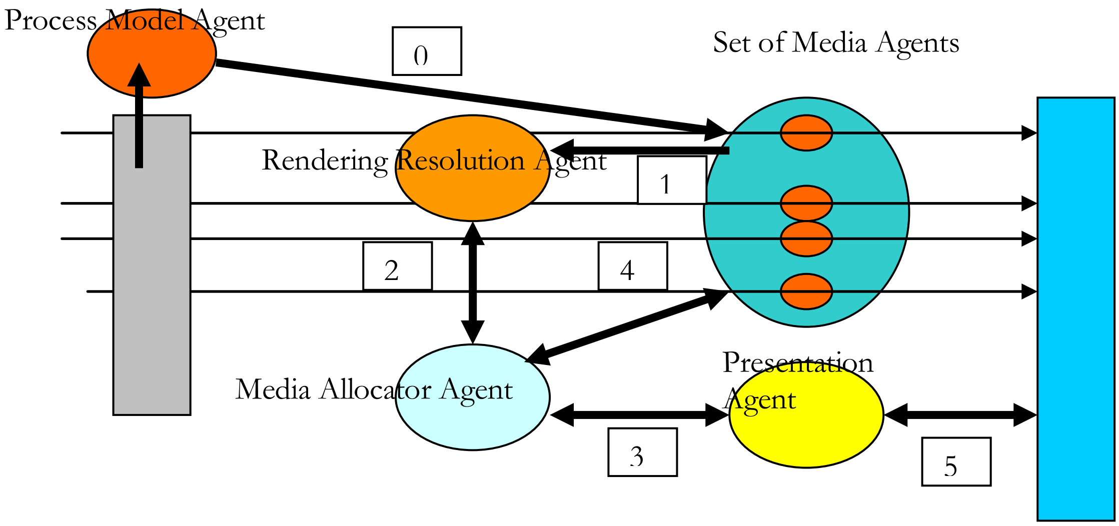


Figure 34 An Example Set of Agent Interactions

As an example, consider the actions following a message from the Process Model Agent to a Media Agent in Figure 34. Let us assume that a condition has occurred which requires the information rendered by the Media Agent to be given a much higher priority

0: The Process Model informs the Media Agent of a problem in the Process that may require adaptation to the current rendering.

1: The Media Agent informs the Rendering Resolution Agent that it has a problem, and that the problem is one of increasing priority for its object (it would probably also describe this as an alarm condition).

2: The Rendering Resolution Agent decides upon a prioritised list of possible representation classes. It decides this after consulting other agents such as the Operator Agent and the Human Factors Database. It concatenates this *context information* with the process independent AMEBICA proprietary lingua-franca protocol. This protocol is in effect a translation of process dependent terms to general AMEBICA terms, this is called AMEBICA SPEAK. This AMEBICA SPEAK knowledge of the process condition that spawned this event, and with the Representation List. This concatenated information is formed into one request for the Human Factors Rules Database. The list is passed to the Media Allocator Agent, which determines possible candidates from the list of the requesting Media Agent.

The Human Factors Rules Database receives the request, and returns its recommendations. The Rendering Resolution Agent, takes these recommendations, resolves any conflicting answers and forms a best-fit list. This best-fit list is composed of the recommendations offered back by the Human Factors Rules Database.

The appropriate representations are marked as being the ones the Media Allocator Agent should chose from, the *best-fit representations*. A second set of representations are also passed on with crude re-ordering applied, and if necessary some general parameter information. This second list, the *normal representation list,* is used by the Media Allocator Agent as a backup set of representations in case none of the representations on the *best-fit list* can be rendered with the current available interface resources. Typically, the Human Factors Rules Database will place the more inappropriate Representations on the *normal list*. This generation of two separate lists increases the reasoning speed within the Media Allocator Agent, as it has a smaller set of representations to reason about.

Both lists are examined by the Rendering Resolution Agent, which has an in-built consistency checker. Upon the Media Allocator Agent making a final decision on which representation is rendered it returns the result to the Rendering Resolution Agent. The Rendering Resolution Agent therefore has a *history* database. Using this database, it can determine what previous results of *Process Events* of this type and under what *Operator Conditions* the Media Allocator Agent has rendered. The Representations that have been rendered most often for a particular circumstance are given increased priority on the *best-fit* representation list. This gives them a higher chance of being rendered and therefore maintaining a crude form of consistency

3: The Media Allocator Agent then checks with the Presentation Agent to determine if the operation can be carried out (from a resource point of view). This may result in removal of representations from the list.

4:

(a) The best remaining list candidate is chosen and the Media Agent sends the appropriate controls to change the representation.

(b) If the resources are not available, the Media Allocator Agent interrogates the other Media Agents using the same resources to see if changes to their representation can free up the required resources. If so, the changes are made.

(c) If not the Media Allocator Agent may have to renegotiate with the Rendering Resolution Agent (as in 3) to get a new list of solutions.

5: The updated representation displayed at the interface informs the Presentation Agent of its new size and position, so that the Presentation Agent can update its view of interface resource usage.

Conclusion

This chapter has detailed the main principles, *stream* and *generality,* on which the conceptual architecture was constructed. These ensure that the reasoning process the system must make to decide on an appropriate adaptation does not delay important data reaching the operator. They also ensure that the system is generic enough to apply to a variety of process control domains, by simply configuring the system at design time.

The chapter has introduced the main agents of the architecture and given an overview of their function and goal. System properties such as domain independence, rendering category, data stream selection were explained.

Lastly, an example set of intra-agent interactions were given which demonstrated how the adaptive system operates.

The next chapter gives a more in-depth explanation of the operation and mechanisms behind the key adaptation agents.

Chapter 9

THE ADAPTATION AGENTS IN DETAIL: MECHANISMS AND RATIONALE

Introduction

This chapter describes, in more detail, the internal workings of the key agents in the AMEBICA system. In particular, it examines the Media Agent, whose job is to offer up lists of possible representations to AMEBICA and, on receipt of a priority list, actually place the rendering in the interface. It also maintains a record of the position of every object in the interface, and this information is used by the Presentation Agent to maintain a view of current resource usage on the interface. The chapter then describes in detail how the Presentation Agent monitors interface real estate and provides information to the Media Allocator Agent to enable it to decide which objects best fit into available spaces.

The chapter further describes the negotiation strategy adopted between the various agents for selecting a best-fit representation. More sophisticated relationships between representation objects (such as Satellite/Source and Parent/Child) are introduced to enable complex interrelationships to be represented. Finally, timing and consistency considerations are discussed.

The Internal Architecture Of The Media Agents

This section describes the internal architecture of the Media Agent, and more specifically how the Media Agent informs the other reasoning agents about the rendering situation at the interface. It is important for the correct operation of the spatial adaptation mechanism that the reasoning agents have a correct and precise view of current interface resource usage (through the Presentation Agent). Therefore, whenever the adaptation system (or indeed the operator) moves a rendering (via its Media Agent) the new position must be registered.

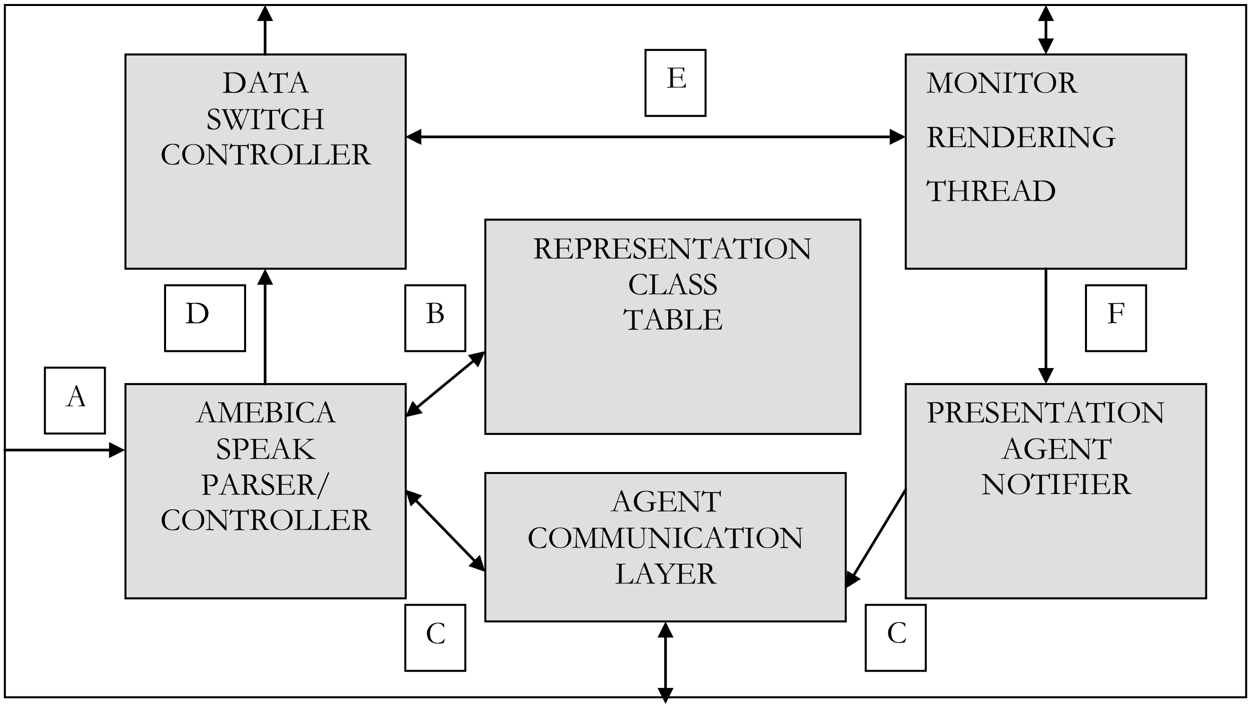

DATA SWITCH CONTROLLER
E
MONITOR RENDERING THREAD
REPRESENTATION CLASS TABLE
D
B
F
A
AMEBICA SPEAK PARSER/ CONTROLLER
AGENT COMMUNICATION LAYER
PRESENTATION AGENT NOTIFIER
C
C

Figure 35 The Media Agent Internal Mechanism

The internal Media Agent architecture shown in Figure 35 demonstrates how a Media Agent keeps the Presentation Agent informed of changes to the spatial status of its rendering. An important component of the architecture is the Monitoring Thread, which is linked directly to the rendered object. It returns the size and positional parameters of its rendering. Upon receiving update information from its rendering, the *Monitoring Thread* then passes on the raw data to the *Presentation Agent Notifier*. The Presentation Agent Notifier's job is to format the data into a readable format that the Presentation Agent can understand. The best way to illustrate how the architecture works is to demonstrate it through a series of scenarios.

Scenario 1: Initial Instantiation of a rendering and its later modification

This scenario first illustrates how a default representation is established by a Media Agent (when a new rendering is required). It then shows how AMEBICA modifies such a rendering.

When the Media Agent is instantiated by the Process Model Agent, it is passed the appropriate contextual information to identify the appropriate representations to be driven by the event that caused it.

1. The contextual information is passed in AMEBICA Speak form (A) from the Process Model Agent to the *AMEBICA Speak Parser/Controller*. This component interprets the AMEBICA speak request and *Evidence* Level and, from these, deduces what it should do with it. It additionally translates the information into appropriate *Media Agent* internal calls.
2. After translating the request the *AMEBICA Speak Parser/Controller* now knows the type of representation needed to satisfy that request. It then submits a request for the type of representation it requires to the *Representation Class Table* (B) The *Representation Class Table* holds knowledge of the appropriate representations for pre-defined *Evidence Levels* and AMEBICA speak information. All representations appropriate for that request are returned to the *AMEBICA Speak Parser/Controller*,
3. The *AMEBICA Speak Parser/Controller* now has a list of appropriate representations.
4. It selects the default representation from the list and the default location
5. The *AMEBICA Speak Parser/Controller* now knows the default representation and the *minimum* size it can take (part of the parameter information associated with each

representation). Its next job is to find out the best location to place the default representation. At this stage it is important that the *default representation* get rendered as quickly as possible. However, whether it immediately gets put on the screen, or not, is wholly dependent on the Media Agent's priority. If it is above a certain threshold (alarm value say) then it will be rendered immediately, in the pre-defined default location. If the priority is relatively low, the Media Agent will query the Presentation Agent to establish the location of appropriate space at the interface.

6. It asks the *Presentation Agent* (C) via the *Agent Communication* Layer, for the location of the largest possible available space that can fit the *minimum representation size* (a parameter common to all representations, this is stored with the representation itself).
7. The *Presentation Agent* returns information on the largest possible representation space available (C).
8. The *AMEBICA Speak Parser/Controller* then has information on the default representation, its size, location and the data streams the representation must connect to. It must now pass all this information (D) to the *Data Switch Controller* which is responsible for instantiating the rendering that corresponds to the representation, and connecting the appropriate data streams to it
9. When it has finished establishing the rendering at the interface it starts (E) the *Monitor Rendering Thread*. It is the job of the *Monitor Rendering Thread* to inform the *Presentation Agent* of any changes in status to the rendering.
10. When the rendering has been fully instantiated it will return its status to the *Monitor Rendering Thread* which (E) tells the *Data Switch Controller* that the rendering is successfully being displayed at the interface. The *Monitor Rendering Thread* then informs the Presentation Agent of the new position (F). This then completes the Media Agent's instantiation of the <u>default</u> representation at the interface.
11. On receiving a modification request, the *AMEBICA Speak Parser/Controller* retrieves the representation list and sends it to the AMEBICA framework, and awaits its suggested

rendering. To do this it first formats these Representation List into the appropriate AMEBICA speak format. It now has to ask the AMEBICA agent framework to return the most appropriate representation, with parameter and location information. It thus passes (C) the representation list to the *Agent Communication Layer*. The *Agent Communication Layer's* job is to route requests to the appropriate agent (Rendering Resolution Agent) using the correct communications strategy (uni-cast, point to point in this case)

12. The AMEBICA framework then returns the most appropriate representation with parameter, size and location information. The Agent Communications Layer routes the request back to the *AMEBICA Speak Parser/Controller*. It then formats the returned information and adds other information about the connecting streams to the (D) *Data Switch Controller*.

The process then repeats steps 9 to 12.

Presentation Agent: Maintaining an Updated Knowledge Base of Interface Usage.

The previous section has shown how Media Agents can initiate adaptation and communicate the present status of their renderings to the adaptive agents, and send updated rendering size/position information to the *Presentation Agent*. Later in this chapter an examination will be made of how the reasoning agents (Rendering Resolution/Media Allocator Agents) then select the appropriate type of adaptation. Before this examination, however, it is necessary to understand the operation of the Presentation Agent. Without an understanding of its workings, it is difficult to grasp the nature of its interaction with the Media Allocator Agent.

User induced changes of any sort at the interface (p 258) as well as AMEBICA adaptations, have a significant effect on how AMEBICA apportions screen space to new events. It is the *Presentation Agent's* job to keep track all these interface changes and to maintain an updated view of the rendering space. It responds to Media Agents when they ask for available space in which to place their renderings. It therefore plays a very important role and has three views of the interface.

Amount of spare, unused screen space.

Where Media Agents renderings are currently located and their size.

What priority each Media Agents rendering has.

The Presentation Agent knows where every rendering is and its current size, and it can use this knowledge together with its knowledge of screen size to work out areas of free screen real estate. The priority of a rendering is used by the reasoning agents to deduce the nature of the representation, and how dominant it should be over available screen real estate, and how much the current representation can overlap/expand on existing (lower priority) representations.

The *Presentation Agent* views every available discrete space at the interface as a *Representation* Space, a special class that represents the space on the interface but also contains meta-data regarding the renderings adjacent to it, and their minimum size and priority. The *Presentation Agent* maintains a table of all the available representation spaces. It also holds information for each representation space such as its location, the name of its Media Agent, its size, its priority and the minimum size it can take. The information is placed in lookup tables as this allows quick lookup and increased performance. The reason the Presentation Agent keeps information on mean and constituent priorities is because this information is used by the Presentation Agent to deem whether a representation space is suitable for adaptation or not.

The Presentation Agent therefore monitors current interface usage and the priorities of representations occupying screen real estate. If a representation space is of irregular shape the

Presentation Agent will split the representation space into several sub representation spaces of regular shape. Each suitable representation space is entered into a lookup table.

When the Media Allocator Agent requests available space it sends the Presentation Agent information on the *priority* and *minimium size* of the calling Representation. The Presentation Agent uses this information to derive a specialist *Repesentation Space class*. This class contains all the normal size/position information, but also contains the Presentation Agent generated *expand/overlap* space. This is set up in terms of methods that can be called to find out the amount of space that can be expanded/overlapped to the top/left/bottom/right of the calling representation space.

Choosing The Best Representation Space

So, how does the Presentation Agent deduce which representation spaces are suitable for the requested representation? It first iteratively cycles through its list of representation spaces and determines whether the size of each is large enough, and if it is not whether it is a candidate for expansion or overlap.

Figure 36 shows an interface which contains six visual spaces (A to F) with priorities assigned (for example A = 3). For each representation space, the Presentation Agent defines a set of edges (Figure 36) calling the top of the representaion space *Horizontal 1*, and the bottom of the representation space *Horizontal 2*. It does the same for the vertical edges taking *Vertical 1* to represent the left hand side of the representation space, and *Vertical 2* to represent the right hand side. It makes no difference how the edges are defined as only one edge from the horizontal and one from the vertical are used to make changes, regardless of actual position.

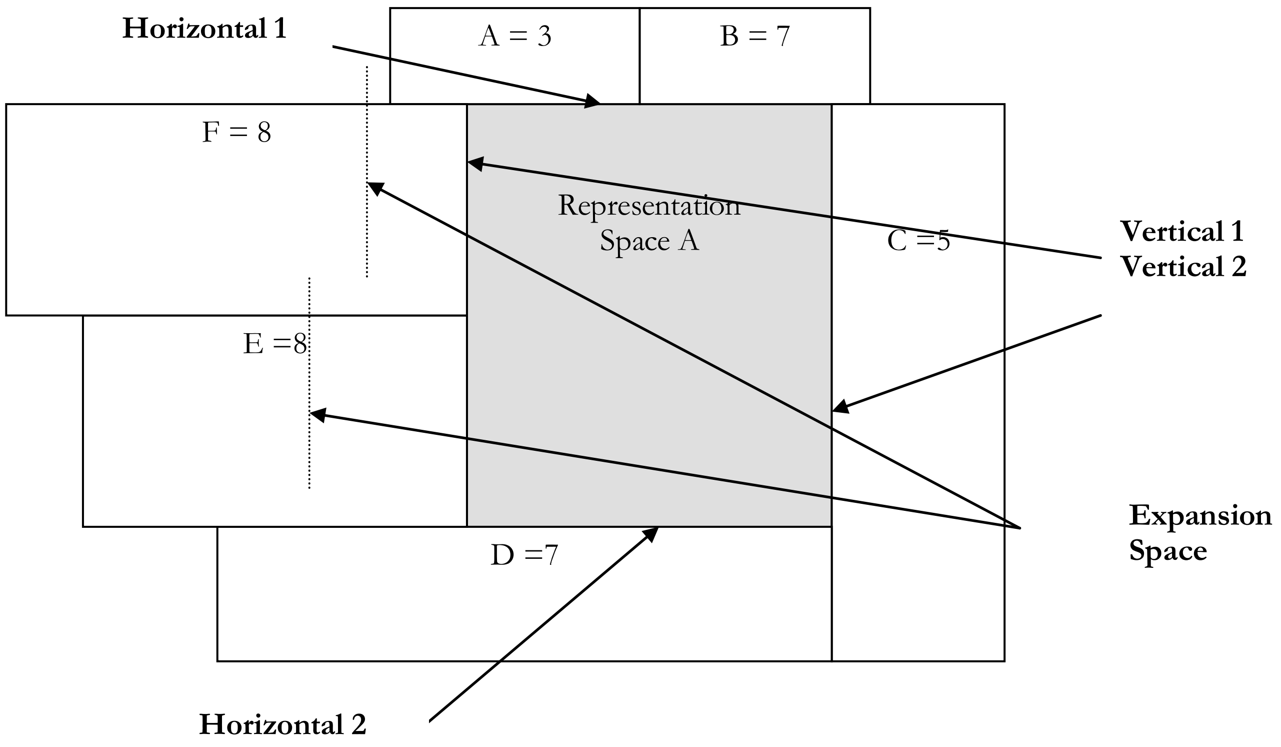


Figure 36 Representation space Illustration

The *Presentation Agent* keeps a table of these values (Table 9) as well as tabulating the average priority for each side thus in Figure 36 above we can see a representation space, surrounded by windows. Each adjacent window is represented by a letter marking its position and a number indicating its priority.

Representation A Space (X,Y)	Horizontal1Priorities	Horizontal1 Mean	Horizontal2 Priorities	Horizontal2 Mean
150,200	3,7	5	7	7
	Vertical1 Priorities	Vertical1 Mean	Vertical2 Priorities	Vertical2 Mean
	8,8	8	5	5

Table 9 *Showing Horizontal and Vertical Priorities for Representation space A*

The *Presentation Agent* receives information on priority and available space from Media Agents as they are updated or initialised. In particular, it knows what the *Minimum Representation Size* is for each rendering, and uses this information to deduce *how much* a representation space can be expanded or overlapped. The *Expansion Space* is worked out by observing the renderings on the side of the representation space that is being examined (say Vertical 1 in this example). The Presentation Agent will look at (E,F) and can *only* expand as far as the *minimum representation size* (shown in Figure 36 by a dotted line) of that rendering. Thus it knows that maximum expansion size of vertical 2 is F. It can only expand if the priorities of the side to be expanded into are *all* less than the calling representation priority. This information is added to Table 10.

The Agent also stores overlap figures. To work out these *Overlap Figures* (Table 10) the Presentation Agent compares the priority of the *incoming representation* with the renderings along the horizontal and vertical axis of each available representation space. It will then check each representation space that has one horizontal and vertical axis whose *average priorities* are lower than the incoming representation (information held in tables 8). For each suitable representation space it will look up the available *Overlap Space* to deduce whether any *Representation space* can be overlapped onto other renderings succesfully. In contrast to the expansion example given above, if (E,F)'s average priority is *less* than or *equal to* the incoming representation then the *Overlap Space* is defined as (E). That is, the incoming representation is of greater average importance than the renderings bordering that side. Thus since the agent is only performing overlapping (thus keeping the original representations size, it just overlaps them). It takes E as being the greatest distance that those representations can be overlapped.

Representation A Space (X,Y)	Horizontal 1 Expansion Space	Horizontal 2 Expansion Space	Vertical 1 Expansion Space	Vertical 2 Expansion Space
150,200	50	20	100	40
	Horizontal 1 Overlap Space	Horizontal 2 Overlap Space	Vertical 1 Overlap Space	Vertical 2 Overlap Space
	70	50	130	70

Table 10 *Showing Expansion Spaces for Representation space A*

Tables of this sort are stored in the *Presentation Agent*, and are used by the Media Allocator Agent to decide where and how to render a representation in an appropriate place in the rendering space.

The Presentation Agent uses the tables in the following way to inform the Media Allocator Agent of the suitability of a representation space.

1. It examines the list of representation spaces (in the tables) and compares them with the minimum size of a representation requested by the Media Allocator Agent
2. If no Representation Space is suitable it attempts to finds a Representation Space that the incoming requested minimum size representation can expand into or overlap. First it checks the priorities for each representation space, starting with the largest and working its way down, to determine which, if any, of the representations spaces can be expanded into. If some representation spaces are found which can be expanded into, the Presentation Agent now works out if any Expansion space is large enough by adding the expansion space size to the representation space size. If one or more is found, it knows the expand operation can be performed.
3. If no space is found that is suitably large, then the Presentation Agent will determine if any representation space is large enough to be overlapped onto. Using the process mentioned above (2) the Presentation Agent determines if any representation space is

large enough and has suitable priorities to overlap into. Then each suitable representation space is checked to see whether the overlap space is large enough for the smallest of the best representations. If one is found, we then know that at least one of the best representations will fit.

4. The *Presentation Agent* finally gathers all the suitable representations, whether they are normal,.expand or overlap candidates and places them in a Vector which is returned to the calling Media Allocator Agent.

If there are insufficient resources to allow either expansion or overlap to occur, the *Presentation Agent* will return a *no available space* flag to the Media Allocator Agent.

Specific Examples.

Window Re-Sizing Case 1: Representation space adjacent to Screen Edge.

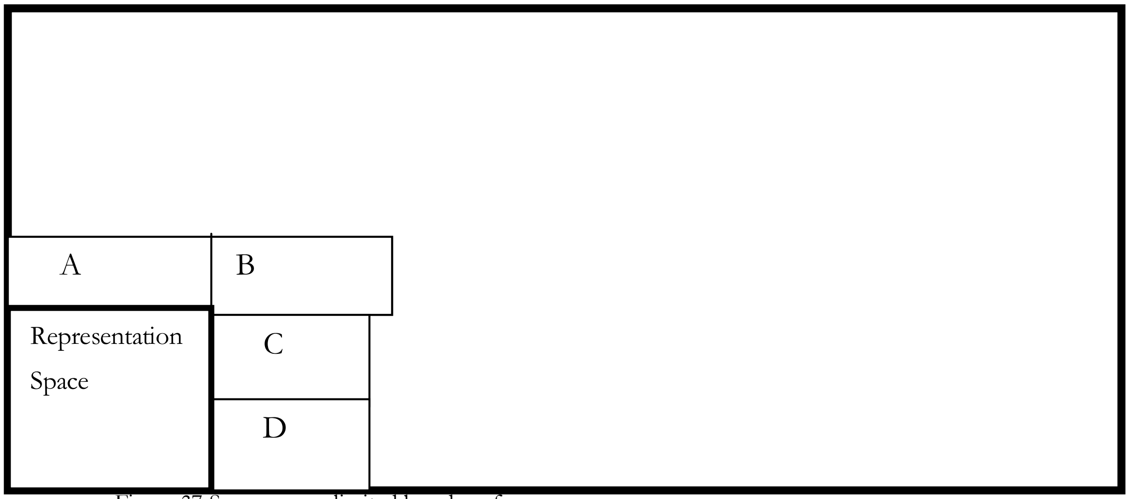


Figure 37 Screen usage limited by edge of screen

In the case where the *Representation space* is adjacent to the edge of the screen, (Figure 37) the *Presentation Agent* takes the screen edges as being of *maximum* priority. The Presentation Agent will then place the priority values and average for A,B into its internal table as Horizontal 1. It does the same for C and D for Vertical 1. Clearly this representation can only expand along 2 sides. So if a representation cannot fit into the representation space, then its priority value must be higher than A,B,C and D to expand into A,B,C,D. To overlap, the representation space priority must be higher than A+B/2 to expand vertically, and higher than C+D/2 to expand horizontally.

Window Re-Sizing Case 2: Representation space with Renderings on all Sides.

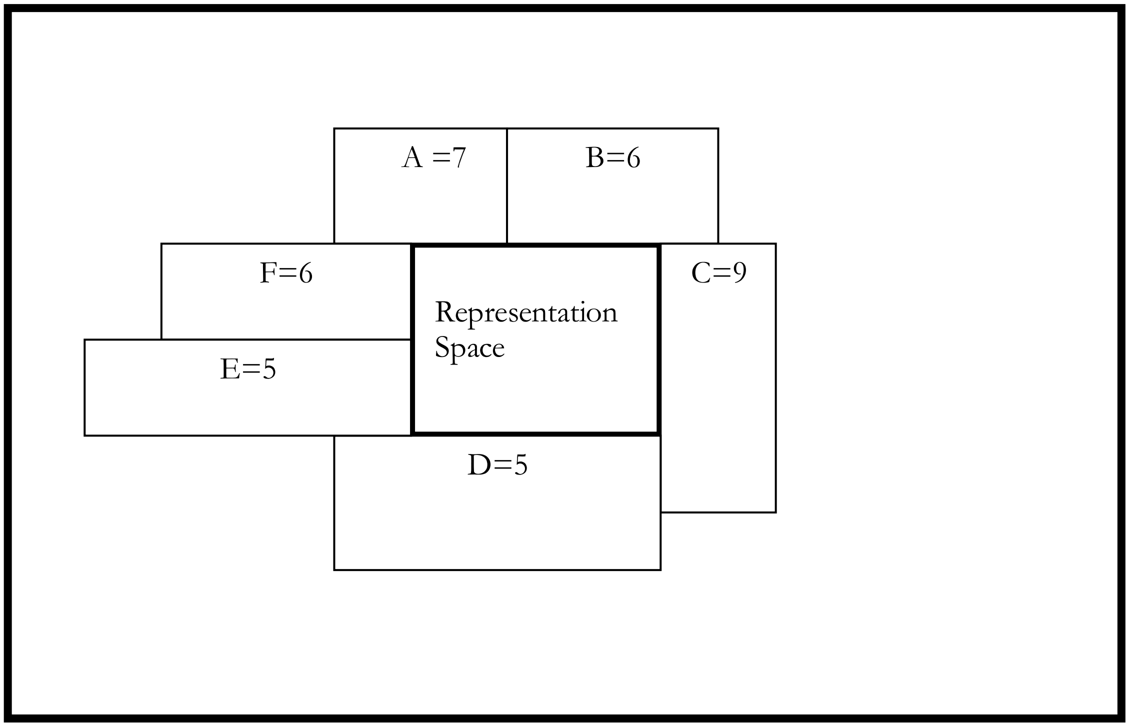


Figure 38 Showing Representation space when surrounded on 4 sides

In the case above (Figure 38), if the incoming representation has a priority value of 7, and is too big both horizontally and vertically to fit into the representation space., the *Presentation Agent* will look at internal priority tables before deciding whether the representation should overlap, expand or do anything else. In this particular case it can expand into (A,B).(D) and (E,F) as all have priorities equal to or less than the incoming priority. C having a priority of 9 cannot be overlapped by a representation with priority 7. In fact it would expand into (E, F) and (D) because they have the lowest priorities. If we argue the case for overlap (which would not be necessary here) then the same would be true with the incoming representation overlapping (E,F) and (D) as they have lowest average priorities.

Window Re-Sizing Case 3: Representation space with Irregular Renderings on all Sides

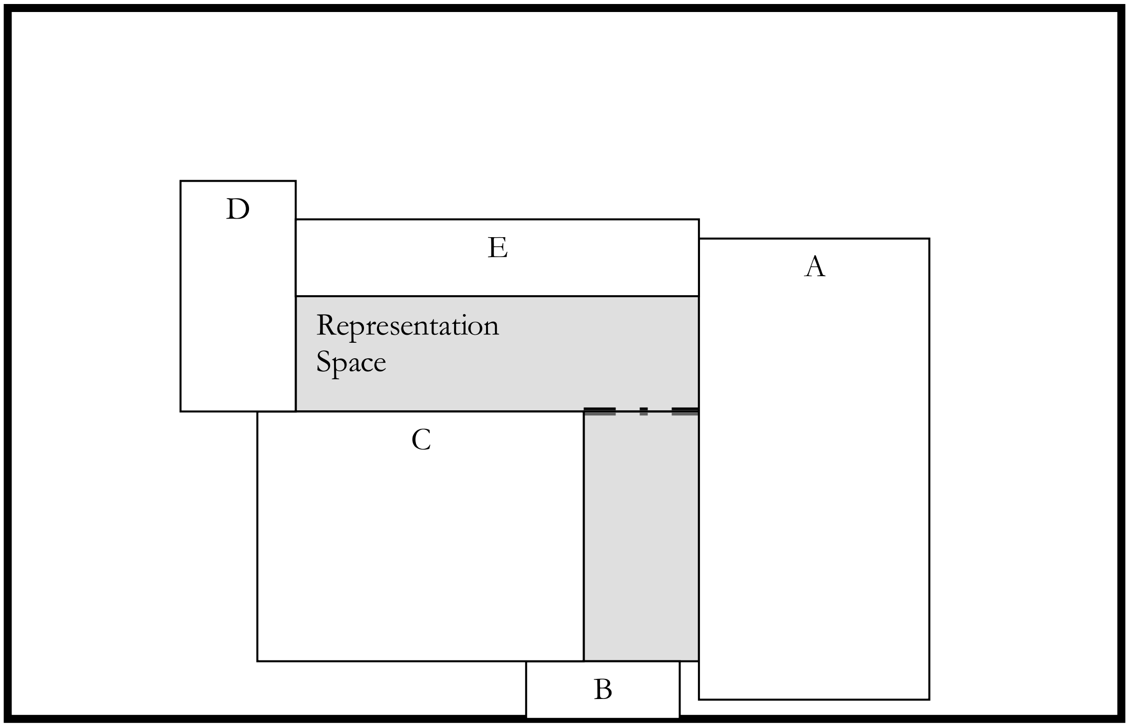


Figure 39 The case of an irregular shape.

In the case of an irregular shaped representation space (Figure 39), the Presentation Agent will take and use the space with the greatest area. Thus, in the above L shaped representation space, the Presentation Agent splits the space into two (marked by the jagged line), and only uses the top part of the L. It then takes (D,E,A,C) as the representations surrounding the representation space, and uses the same logic shown in the previous example to deduce whether expand or overlap are possible.

The Media Allocator Agent: Finding The Best Locations And Representation

It is the job of the Media Allocator Agent to try and make the best possible choice for a representation of an event at the interface. It therefore has to take into account resource usage at the interface (both visual and audio). Additionally, it attempts to keep its choice of representation as consistent as possible with other representations of that type.

One of the major problems faced by the Media Allocator Agent is that of balancing the need to provide the best representation (that is the representation with the highest priority) against available interface space. For instance, consider the case of the Media Allocator Agent being passed a list of representations. Of these representations, only one low priority representation is able to fit onto the screen. Does the Media Allocator Agent select that representation? If it does select the low priority representation, it has disregarded the best choice solution, which negates somewhat the Rendering Resolution Agent 's job. Alternatively, the Media Allocator Agent could take the first choice representation and attempt to adapt current screen usage to make space for it.

There are several different strategies the Media Allocator Agent might adopt to cope with the trade off between best choice representation and available interface space. The strategy adopted in this thesis is *best-fit list*. Since the best-fit list consists of representations which are all suitable for rendering then the Media Allocator Agent can utilise any representation. The real problem is finding suitable interface resources to accommodate them. The mechanism for dealing with this involves the Presentation Agent sending the Media Allocator agent a list of available *Representation* *spaces*. Upon receiving the list the Media Allocator Agent attempts to fit the highest ranking representation on the list (the best representation as determined in the rankings set by the Rendering Resolution Agent) onto the screen. To do this the Media Allocator Agent sends the Presentation Agent the priority value of the representation along with the minimum size of the smallest representation on the best-fit list. The Presentation Agent then uses this to send back a list of appropriate representation spaces the Media Allocator can use to decide upon its rendering strategy. It will initially try and display a representation from the *normal* representation spaces. If this is not possible it has other available options such as re-sizing windows at the interface to create another space.

Media Allocation Principles for Rendering Selections

To control how the *Media Allocator Agents* determines which spaces are suitable for what representation there must be some general guiding principles. These principles are used to work out some lower granularity rules later which will govern the window sizing operation.

> No active moving of windows is allowed. The operator has a spatial understanding of where unserviced alarms are, if windows are moved around, it disrupts the operators effectiveness.
>
> Window re-sizing is allowed, as it only partially affects the operator's spatial map.
>
> No selection of alterative representations is allowed for *existing* windows. This disrupts consistency of interface and therefore operator effectiveness.
>
> Any representation can be selected an run-time as long as it fits interface resource conditions.
>
> Although a representation being rendered on the screen cannot change its form (type of representation) it can change the parameters of that rendering (colours, size).
>
> The time taken to select a representation that can fit onto the screen should be as minimal as possible.

Media Allocator Agent should first select any representation on its best-representation list that fits into current interface usage.

These general principles were used to arrive at the rules given below for the Media Allocator Agents actual behaviour.

Media Allocation Rules

The Media Allocator Agent has several strategies to deal with the problem of fitting an appropriate representation onto the current interface configuration. The basic strategies are listed below in the reasoning order of the Media Allocator Agent. It will first try and fit a representation into a normal space at the interface, without recourse to re-configuration.

Normal Fit: The Presentation Agent will return to the Media Allocator Agent a set of representation spaces that are greater in size than the smallest configuration of the smallest representation on the best fit list. It examines the priorities and also returns representation spaces that are available to be expanded or overlapped if necessary. From this general list, the Media Allocator tries to determine whether each representation on the best-fit list can be fit onto the screen. If it can, they are placed along with the representation space into a Composite object and stored on a sub list. The Media Allocator Agent then builds this list, and, if it can place the highest ranking on the list on the screen without re-configuring the interface, it does so. If not it examines the Expansion representation spaces.

Expansion: If it is not possible to fit high ranking representation of the best-list representations into the available representation space, then the Media Allocator Agent may attempt to reduce the size of the representations surrounding a representation space.

Overlap: If none of the representations spaces are suitable for expansion, then the Media Allocator Agent will determine whether any are suitable for overlapping on X, Y. It is preferable to try and expand a space, to allow all the involved windows to be fully visible, rather than to overlap. By overlapping only partial visibility of some windows is experienced. However in many situations, screen real estate is not sufficient for the information needed to be displayed.

No Appropriate Space Can Be Found: If the Media Allocator Agent cannot **overlap, expand** or **fit into the representation space** then it will try the representations on the *normal representation list.* This occurrence will usually occur because the piority of the requesting event is of a lower value than any of the surrounding windows, and the minimum size representation is STILL larger than the available space.

This is ALWAYS the case UNLESS the event is operator requested. If the event is operator requested then an attempt will be made to fit the representation for that event into the available space. Otherwise the representation will overlap where the EVENT was requested, i.e. mouse pressed, or overlap in minimum space. In fact it tries to first fit, then overlap. If neither case can be rendered then it overlaps where the operator requested. operator requested events are *always* of the highest priority.

Some Typical Examples Of Media Allocation Activity

In the next section a number of typical cases are examined, and a discussion is provided of how the adaptive system reacts.

The Case Of Finding A Large Enough Representation Space To Fit A Representation In.

The Media Allocator Agent passes the *Minimum Size* parameter along with priority information to the *Presentation Agent*. The *Presentation Agent* first finds the largest available representation space, and checks whether the smallest best-fit representation will fit. If so, it knows it can fit in at least one of the best-fit representations. It then iteratively repeats this for all the available representations spaces.

If more than one representation space is appropriate, the *Presentation Agent* returns the appropriate representation spaces with certain information (X,Y position of top left hand corner). The Media Allocator then examines all the spaces and, starting with the highest ranking representation in the best-fit list, attempts to find a suitable space. It will then return the selected representation along with parameter information to the relevant Media Agent.

The Case Of Reduction Of Existing Renderings (Expansion Of A Representation Space) To Accommodate A New Rendering

The following is how the adaptive system deduces expansion:

1. Presentation Agent checks to see whether expansion is possible by attempting to fit the *minimum size* of the smallest representation into the largest available representation space of lower priority. It checks whether it can expand by checking the Expansion size of that representation space. It then checks every available suitable representation space (that is of lower priority) and see whether any have a large enough Expansion Space to accommodate the smallest representation. If none are found then skip to Overlap procedure.
2. It gathers the list of representation spaces by compiling a list of suitable representation spaces by finding representation spaces with more than 2 edges whose bordering renderings are all of lower priority.
3. For each of these suitable representation spaces, the Presentation Agent deduces the available expansion space. It does this by referencing its table of *minimum sizes* for the bordering renderings. From this it notes whether expansion into each Media Agent generated representation will reduce the targeted rendering to a size smaller than its minimum allowable size. In this way it discovers which of the representation spaces can be expanded into without making the surrounding representation unreadable, because they have been shrunk too much.

4. From this it works out the *expansion space* available, and hence the maximum size each representation space can expand to.
5. This list of representation spaces which can expanded is returned to the Media Allocator Agent. The Media Allocator Agent then attempts to render the highest ranking rendering. If none can be found with the size parameters given by the Rendering Resolution Agent then the *minimum size* for each representation is attempted.

The representation space composite object contains information the Media Allocator Agent needs to re-configure the interface. This includes X,Y position of the top left hand corner of that space and the two Media Agents which are to be expanded into.

To enlarge on the overview given above, the Presentation Agent first takes the priority of the incoming representation, and for each Horizontal and Vertical of each representation space will compare priorities of adjacent renderings. If it finds a Horizontal and Vertical axis, where the bordering representations are ALL LOWER priority than the incoming representation, then the representation space is deemed initially ready for expansion. It will not consider an axis as suitable for expansion if one rendering is of higher priority.

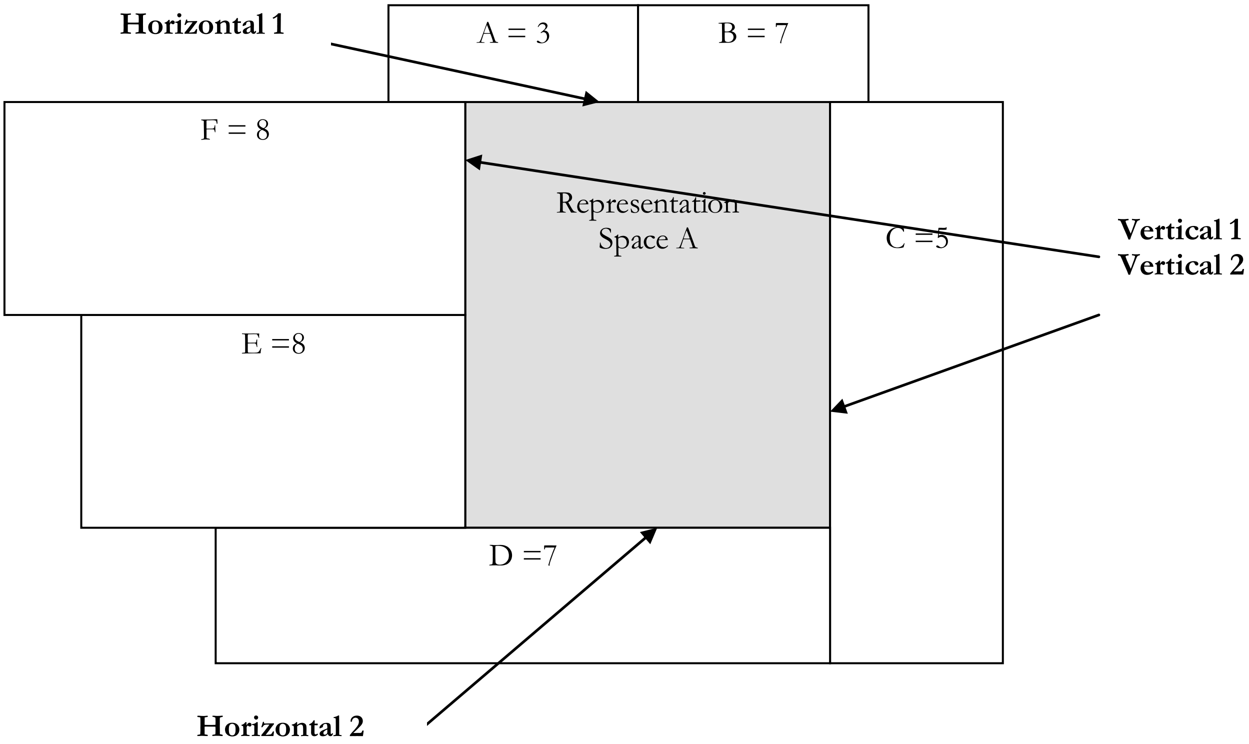


Figure 40 Representation Space Example

Once it knows two axis of a representaion space are *ready for expansion*, it will check to see whether there is enough room on those axes to expand by checking the Expansion space. To deduce this the Presentation Agent, uses its lookup table for expansion (Table 10) to deduce whether that axis offers enough *Expansion Space*.

To give an example of why this system was chosen assume there is an incoming representation of priority 6 as seen in Figure 40. Then rendering A can be reduced on Horizontal 1, but rendering B cannot because it has a higher priority. We cannot reduce A and not B, because the representation must have a straight side, and so A and B must be reduced equally. The same is true of Horizontal 2, where rendering D is of priority 7 and therefore higher than the incoming representation.. AMEBICA would not be able to reduce renderings E and F either, because both have higher priority. It *can* however expand C on *Vertical 2*, however that gives us just one side to expand into and we need two. Therefore in the case of the representation being of priority 6, this representation space is unavailable. If we chose an incoming priority of 8, then this representation space is suitable as ALL sides can be expanded into.

This provides the following additional rules for the Media Allocator:

RULE: Examine Horizontal/Vertical Priorities in table, if INCOMING priority is BETTER THAN or EQUAL TO ALL representations on that side, then it can expand there.

RULE: For a Representation space to be suitable, there must be at least 1 horizontal and 1 vertical edge found ready for expansion.

The Case Of Being Unable To Expand, Therefore Representation Must Overlap.

Because this strategy overlaps existing renderings, then the need to have ALL renderings above the incoming priority is negated. However, it is preferable to overlap renderings of lower importance. To ascertain whether a representation space is suitable for overlapping, the Presentation Agent attempts to discover whether the *average priorites* of renderings bordering one of the vertical and one of the horizontal axes are of lower priority than an incoming representation.

The Presentation Agent deduces the average priorities for each represenation on-the-fly as renderings are altered at the interface and the information is stored internally in a table. If two sides are found of a single representation space that can fit the smallest representation on the best-fit representation list then it knows it can overlap and an Overlap Representation Space list is generated which is returned to the Media Allocator Agent.

The Media Allocator Agent then looks at each representation on the representation list, and attempts to find a suitable space is found that can accommodate it. If no space can be found for any of the representations with standard sizes then the *minimum size* for each representation is used until a space is found. If no representations spaces can be found, then the course the Media Allocator takes is wholly dependent on the priority of he incoming event. If the event is of a very high importance (urgent alarm) then the Media Allocather will *force* overlapping (so that the Representation is forced on screen), otherwise other strategies are adopted (see next section).

The way *forced overlapping* works is by the Media Allocator Agent checking the *normal representation list* (as opposed to the best-fit list) and attempting to find a representation that fits into the largest available representation space. There will usually be a representation that will fit on this list, even if it is not the best possible way to render an event. For instance, if text is one of the representations on the *normal representation list* then it requires very little representation space to accommodate it. Even if this space is still too great, overlap still takes place. The Media Allocator simply displays the *smallest possible* Representation in the largest available space. The reason for this is that although forced overlapping covers the underlying renderings with the new representation, the underlying renderings are still there. If they are unserviced for a long time, *because they have been covered,* AMEBICA will detect this and send an update event. AMEBICA will then re-locate them if necessary. In this way the overlap case is quite different from the Expansion case where the renderings surrounding the representation space are altered. In that case they cannot expand past their minimum size because they become unreadable.

Alternative Strategies For The Case Of Being Unable To Expand Or Overlap.

If there are insufficient resources available to expand or overlap succesfully then the strategy used is that of overlapping the minimum size of the smallest representation into the largest available representation space. However this is but one appropach, and others can be considered.

There are two ways to deal with a representation list that cannot fit into a representation space:

Put up dialog box asking the operator what they want to do with windows and options available

Have a queuing system which is utilised for events that cannot be displayed. When the *Media Allocator Agent Quorom* notices that screen real estate has been made available or that a representation space has increased in size, it will try and service the first event in the queue.

The latter approach is adopted. The calling event that cannot be rendered is held in a queue located in the *Media Allocator Agent*. The queue is formed and serviced in order of priority. The queue is not serviced in order of time because if an event is held in the queue for some time, we can expect AMEBICA to send an update event for it. The *Media Allocator Agent* services this update event by *increasing* the priority of that event in the queue. By doing this the event will skip several places in the queue, and should get serviced quicker.

Periodically the *Presentation Agent* notifies the Media Allocator Agent of current on-screen priorities and space. The Media Allocator Agent will then check the queue status. If the first event in the queue can be rendered with the new interface data it is put on to the screen as a minimum size representation. The *Presentation Agent* will then send back to the Media Allocator Agent the new space allocation including the updated event. The *Media Allocator Agent* will keep moving down the list until all the queued events have been serviced.

Co-Ordination Process Between The Media Allocator Agent, The Presentation Agent And The Rendering Resolution Agent

The Rendering Resolution Agent, Media Allocator Agent and Presentation Agent interact as follows:

1. The Rendering Resolution Agent sends a list of appropriate representations to the Media Allocator Agent. The Media Allocator Agent has a buffering system, and the incoming

requests are time stamped and queued according to priority first then are arranged according to a First-In-First-Out (time stamped) system.

2. The *Media Allocator Agent* receives two appended lists from the Rendering Resolution Agent. These two lists are the *best-fit* list and the *normal representation list.* The Media Allocator Agent must now try and find suitable screen resources for the representations on the best-fit list first.
3. The *Media Allocator Agent* will then examine the current Representation list and extract the minimum size of the smallest representation. This information along with the spawning events priority is passed on to the Presentation Agent.
4. The *Presentation Agent* will return a list of representation spaces in which the representations can be displayed. Each representation space object also contains information on the amount of expansion/overlap space on each vertex and the name of the Media Agents in the expansion/overlap space.
5. The Media Allocator Agent will decide upon its strategy based upon the rankings of the representation and the available space.
6. If the representation selected needs to placed into an expanded representation space, then the Media Allocator Agent will contact the appropriate Media Agents and inform them of the axis and the degree of shrinkage required. The Media Agents will then shrink their renderings, thus increasing the size of the representation space. The Media Agents then inform the Presentation Agent which updates its view of the interface as necessary.
7. If the necessary space has been made then the Media Allocator Agent will contact the appropriate Media Agent (cross-referenced by Name, and passed to it as one of the arguments in the event), and inform it of relevant positional, parameter, representation and size information. The Media Agent will then render the selected representation and load it with the appropriate parameters (size, colour, details and so on). The Media Agent will then return the new updated information to the Presentation Agent.

Intelligent Placement Of Related Windows: The Satellite-Source Relation

The adaptive spatial reasoning process mentioned above places windows in available spaces without constraint. However, in certain situations it is preferable for related windows to be grouped together.

This capability is enabled by determining at design time the relation between certain representations. For instance, there may be a representation of a sub-station that shows an overview of the relations between certain transformers within that substation. To determine the values of specific transformer voltage levels separate lower level representations are required, each showing a single value. The higher level overview representation and the lower level specific representations are related, and should therefore be placed close together so that the operator can cross reference and switch between the information easily. In non-adaptive systems, each window would be launched on the screen at a pre-determined point and would not adaptively locate itself near to related windows. In our nomenclature we call the *main* window that *Source* and the related windows *Satellites*.

The relationship between sources and satellites are determined at design time and placed as a *constraint* within the representation data for the appropriate representations. Thus, when the Media Allocator Agent processes the representation data it can determine whether the representation is a source/satellite and decide upon an appropriate strategy.

This general relationship does not specify the exact nature of the spatial relationship between representations. To ascertain this certain preferences can be contained as constraints within the representation data for each representation. So, for instance, a satellite representation may have constraints informing the Media Allocator that its prefered spatial position is *below* and *near* its *Source* representation. The Media Allocator would then try and organize the interface as best it can to accommodate this constraint. The constraints of this type contained within the system are:

Above

Below

Right Of

Left Of

Near (can be applied as an AND condition to other constraints)

With the exception of the *Near* constraint, all the position constraints do not imply proximity to source, they simply indicate relative position. Thus, if the positional constraint is *"LEFT_OF"* the Media Allocator Agent only has an obligation to place the requesting Satellite representation somewhere to the left of the source. Its main priority remains attempting to place the highest ranking representation on the representation list somewhere on screen – in this case to the left of the *Source*. Therefore, the Media Allocator Agent would rather place the *best choice* (but larger say) representation some distance away from the source (if that was the only place it could accommodate it) than place a smaller and lower ranking representation near to the *source*.

Not all representations have a Source/Satellite relationship, others are merely *Related_To*. This constraint indicates which other representations a certain representation is related to, and allows the Media Allocator Agent to attempt to place related representations in positions determined by the spatial constraints listed above.

Example Source/Satellite Interaction For *Location* Position Constraints (Below, Above, Right, Left)

1. *A Source* representation named "BOILER 1A" arrives at the Media Allocator Agent. After some interaction with the *Presentation Agent* the Media Allocator places the Source in the most appropriate location.
2. A *Satellite* representation arrives at the Media Allocator Agent. The Media Allocator recognises it as such by examining the Satellite representations constraint meta-data. The meta-

data informs the Media Allocator Agent that representation is of type *Satellie*, whose source is named “BOILER 1A”, and whose placement constraint is “TO_LEFT_OF”.

3. The Media Allocator Agent sends a request to the Presentation Agent for a list of currently available Representation spaces. The Media Allocator Agent sends a request to the Presentation Agent for the original source representation data, including its current position and size.

4. The Media Allocator Agent then derives a new list of available representation spaces, that represent all the available spaces in the area set by the positional constraint (for instance to the *left* of the source). To do this it extracts the (x,y) position of the axis (left axis here), and deduces the (x,y) position of the appropriate axis of each represenation space (in this case the *right hand axis* of the representation space). From these two co-ordinates the sub-list is derived.

5. Once the derived list has been obtained the Media Allocator Agent performs its normal operation of examining each space to deduce whether any can fit the *Preferred Size* of the highest ranking representation on the *best-fit list*. If none can be found it tries the other representations on the list and attempts to match their *Prefered* size with the available space. If none can fit, it starts again with the highest ranking representation and examines the spaces, any space that can fit it in, and is greater in size than its *Minimum Size* constraint is suitable. Otherwise it repeats this procedure with lower ranking representations until a possible sutiable solution is found.

6. If no appropriate space can be found, the Media Allocator Agent will first attempt to shrink the source on the appropriate axis, *if* there are adjacent representations and *if* the shrinking allows one of the *best-fit* representations to be placed in the freed up space.

7. If no space can still be found, then the Media Allocator defaults to an alternative *Near* strategy. It then attempts to place the representation somewhere near the *Source*.

Example Source/Satellite Interaction For *Near* Position Constraints (Below, Above, Right, Left)

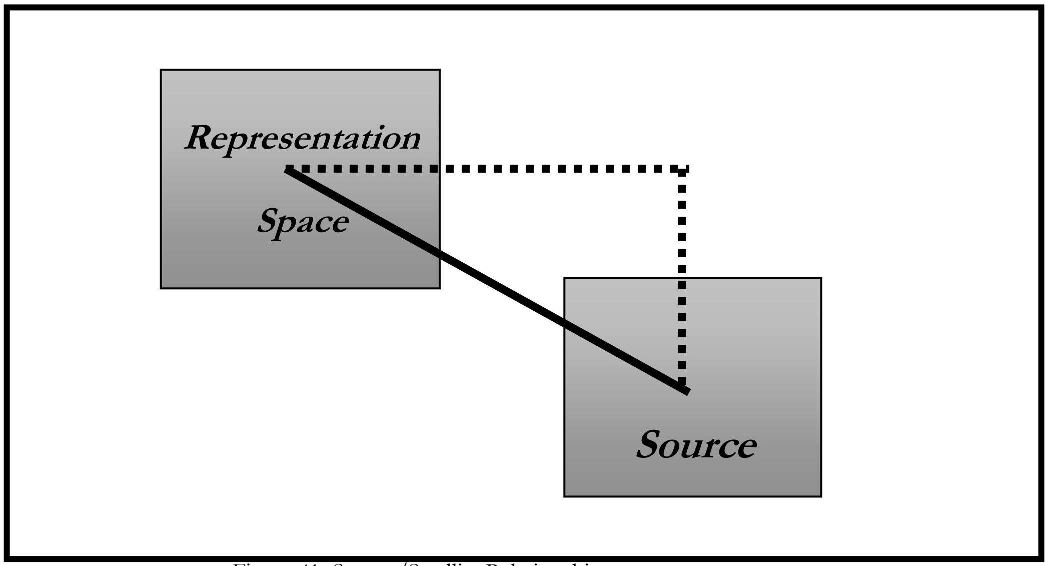


Figure 41 Source/Satellite Relationship

1. The procedure for *Near is* identical to the positional constraints mentioned above up until the point where the available representation spaces are retrieved along with source data. (Steps 1-3 above).

2. The Media Allocator Agent then needs to deduce the centre of the *Source* representation, and the centre for each of the available representation spaces.

3. It therefore cycles through the representation space list deriving the centre for each representation. It then can use this to deduce the distance each space is away from the source.

4. The method used to deduce the distance is rather simple and can be seen in Figure 41. From deducing the two centre points, the hypotenuse (distance in other words) can be simply deduced using Pythagoras' theorem. The distance for each representation space is then added to its meta-data

5. The Media Allocator Agent then cycles through the representation list and sorts the list by order of distance.

6. Each *Source* contains within it a parameter called *Near Distance*. This stipulates the distance the designers class as being near. Thus it may be that anything within 200 pixels is classed as near. The Media Allocator Agent uses this parameter to try and see whether any of the representations that fall within the *Near Distance* can hold the highest ranking (best) representation of the Satellite to be placed. If there are more than one it selects the nearest.

7. If it cannot place the highest ranking representation of the satellite within the *Near Distance*, it calls upon another parameter in the source called *Suitable Distance*. The *Suitable Distance* represents a distance further out than *Near* but still allowable (Figure 42). It then follows the same procedure seen in part 6. If the highest ranking representation cannot be placed within the *Suitable Distance,* it goes back to part 6 and tries to place the *second* highest ranking representation within the *Near Distance*. The process of 6-7 is then repeated until one of the representations in the *best-fit* list can be placed in either a *near* or *suitable* distance.

8. If none of the best-fit representations can be placed at a *Near* or *Suitable* distance, the Media Allocator Agent examines the remainder of the representation spaces and simply attempts to place the highest ranking representation as close as possible to the source. If it cannot it cycles through others on the best-fit list.

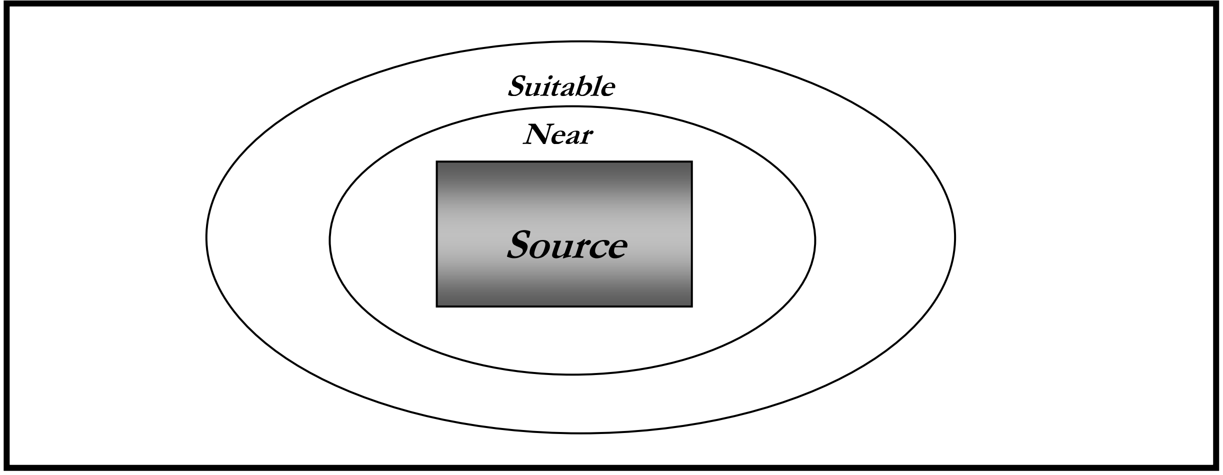


Figure 42 Source/Satellite Relationship For Near Constraint

Media Allocator Agent: Other Interface Manipulations - The Parent/Child Relantionship.

The spatial adaptation so far described in this chapter relate to single representations. However, in process control it is normal to have large representations (Electricity Network Grid say) - termed here the *Parent* - composed of many smaller composite representations (Substations on the grid) which are termed *Children*. In such cases the normal form of free spatial adaptation is not possible on the lower granular representations since they are fixed in their position on the larger representation. This does not mean, however, that the Adaptive System cannot act on these representations, it is still free to alter the form of the representation as long as it does not alter the configuration of the large representation. Any adaptive action taken on the *Children* Representations is performed through their respective *Parent Representation*

The parent representations can have normal adaptive functions performed on them, such as being changed in form (type of representation) and position/size (spatial adaptation). However, by virtue of being a composite object, other adaptive functions are applicable such as:

Zoom: The Media Allocator Agent can set the zoom factor on a composite object, so that certain areas of the Parent object are magnified or reduced as necessary to produce the best possible view of the representation for a certain context.

Translate: The Media Allocator Agent can move the representation around to the area required to be viewed. Thus, if the parent representation is an electricity network grid, which is much larger in size than the limited viewing window (container), the Media Allocator Agent has the capability to move the underlying larger representation around so that the area in view is the most appropriate in a certain condition.

Update: The Media Allocator Agent can send an update request, which informs the parent representation that it should update all its children. In some cases, the children may have changed representation, but since the Parent controls what is viewed this change may have not been performed by the Parent. The update control ensures all Children displayed are in their most current context.

Example Of Parent/Child Adaptive Behaviour

1. The Parent Representation is initialised and processed by the Media Allocator Agent.

2. A Process condition occurs that requires a Child Representation to update its priority from a low value to a high value.

3. The Media Allocator Agent receives the Child Representation, and obtains from it the name of its Parent Representation.

4. The Media Allocator Agent then sends a request for a list of all the current children of that Parent Representation from the Presentation Agent.

5. The Presentation Agent returns the list of current Children. The Media Allocator Agent then has to decide whether the priority of the incoming Child Representation warrants re-configuration of the Parent Representation. It does this by comparing all the priorities of the children.

6. It will then decide that it is important that the two highest priority children should be visible. It takes the relative co-ordinates of the two children (relative because they are only applicable within the Parent Representation, and are quite different from the absolute co-ordinates used for normal window adaptation) and deduces the area that should be visible to include those two children.

7. It forms a visible rectangle area, and sends this to the Parent Media Agent, which sets the Zoom/Translate settings so that these two Children are visible.

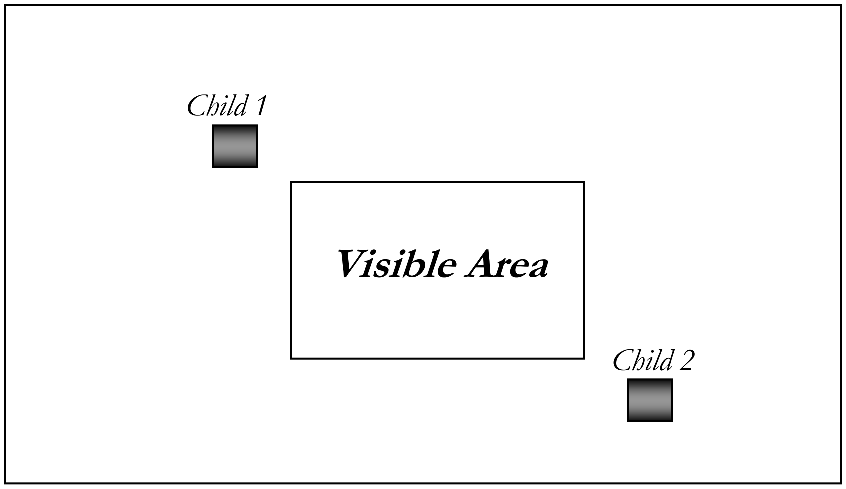


Figure 43 Children Representations Outside Of Visible Area

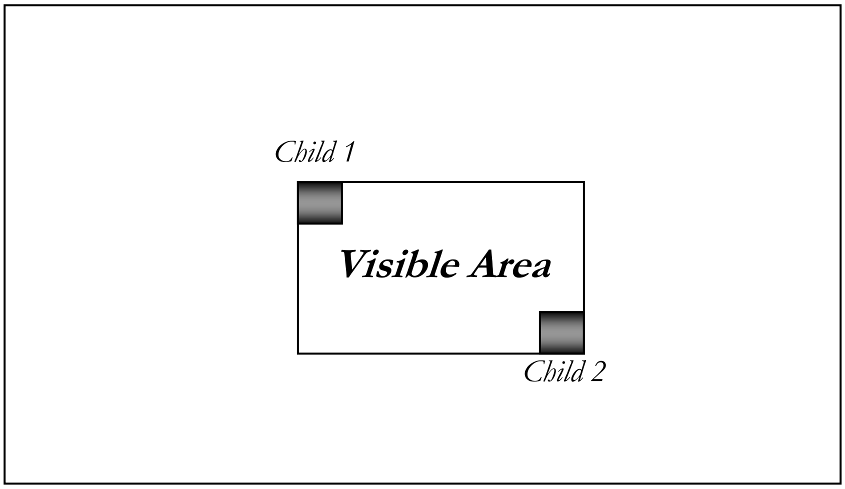


Figure 44 Children Inside of Visible Area

The After Zoom/Translate Effects

Once a zoom or translate has been activated, the visible screen real estate changes. This means that the currently visible sources may be different to the original configuration. If this is the case, the system tries to maintain the source/satellite relationship. Therefore after each zoom the Media Allocator Agent queries the Presentation Agent for an updated current view of which *Sources* and currently visible. From this it requests the Presentation Agent for the sources attached to the specified *Sources*.

Upon receiving a list, it orders the list according to:

1. The Priority of each Source.
2. The Priority of each Satellite

Where at the top of the list is the *highest* priority Satellite of the *highest* priority Source, and the bottom of the list is the *lowest* priority Satellite of the *lowest* priority Source.

Once the list has been ordered the Media Allocator proceeds to service each *positional* constraint of the Satellites (*to_left,near etc)* and attempts to place them as required.

AMEBICA Consistency

Consistency is always a difficult issue within an adaptive interface. Having too many different representations of the same type of event rendered at the interface might confuse the operator as to their purpose. This can sometimes contradict the adaptive system's primary goal of dynamically selecting the *best* representation for the *context* in which that event occurred. By performing this goal without taking into account ther issue of consistency, it is conceivable that there may exist, at the interface, different representations of the same type event. For example, if there are several *measurement* events represented by several *dial* representations on screen, and another *urgent* measurement event occurs in a *boiler*, AMEBICA might decide that since the origin of the event is from a boiler, a *thermometer* representation might be appropriate, coloured red to indicate the urgency. If consistency was absolutely adhered to AMEBICA would select another dial and not a thermometer, thus making the whole reasoning process redundant.

There is this conflict between the two goals of selecting the most appropriate representation, and keeping the interface as consistent as possible The system therefore, as far as possible, tries to balance these goals.

To do this, the adaptivity agents need to have an idea of interface representation history. Therefore the Media Allocator Agent has a *consistency database.* This is also used by the Rendering Resolution Agent which takes into account the history for each representation event it considers. If it sees that in the past several representations have been presented at the inteface in the form of one particular represenation, it will raise the ranking of that Representation in the Representation List. This ensures that that representation has a better chance of being rendered by the Media Allocator Agent.. Additionally, the Media Allocator Agent uses the Consistency Database, when considering a set of Representation that can be rendered on screen it will select the consistent represenation as long as all things are equal.

If there are choice of several representations that can all be equally fitted onto screen.

AND

If all of these representations are of equal priority.

AND

If there are other representations of the same event and of the same priority already on screen or that have been recently displayed,

THEN

select the representation that has been used most at the Interface.

Affect Of Operator Induced Changes At The AMEBICA Interface

The above sections have mainly concentrated on the way the AMEBICA system operates on events from the process itself. How should operator induced change at the interface affect it?

As the operator moves renderings around the screen, changes their size, or kills them, AMEBICA should *not* attempt to dynamically adapt to these changes. In other words AMEBICA does not attempt, on-the-fly, to update the interface as soon as the operator has finished altering it.

Rather, AMEBICA will only react to operator induced changes when new events arrive which require AMEBICA to render a representation in response to them. Thus, as AMEBICA attempts to place the representation on screen, it will note that the operator has altered the interface usage resources and, through the Presentation Agent, will notify the framework of these changes. It therefore acts in the same way as it would in normal circumstances, with the Presentation Agent simply noting an alteration in resource usage and updating its tables.

An important point here is that the current window, that the operator is working on, *should not* be altered by AMEBICA. This is the case even if this current window is overlaying several other important windows. However, AMEBICA can eventually realise that the underlying windows have not been serviced (probably because they are not visible, they are covered by the operator's window) and try to adapt or emphasise the underlying windows to get the operators attention.

So how do changes that are operator induced ripple through to AMEBICA. The operator can perform two types of task:

1. The operator can affect the size parameters, or the location of a rendering.
2. The operator can induce changes on the rendering itself, such as alter the type of representation, or change some internal parameter (colours etc), or indeed kill it.

The mechanism for dealing with these two types of operation is completely different.

The Two Types of Operator Induced Change

Type 1: Operator Affecting The Size Parameters Or Location Of A Rendering

In this first case, the operator induced change is merely one of altering screen real estate usage. This is important for forthcoming events, which need to know of any changes of interface resource usage. Such operator induced changes are observed and recorded by the *Presentation Agent*. This alteration of the spatial parameters of a rendering has no direct affect on the Media Agent responsible for the rendering. All changes induced by the operator are handled by the renderings themselves in the normal way. Windows can be moved and overlay other windows without any recourse to adaptation.

Type 2: The Operator Inducing Changes On The Rendering Itself Including The Type Of Representation Or Internal Parameters

In the second case the operator is actually performing an action at the interface that involves AMEBICA changing state in some way. Any action, which involves AMEBICA changing state, should involve the *Operator Agent*. In this way the operator effectively requests a change of state via the *Operator Agent*. In other words if a rendering receives a request from the operator for a state change, it informs the *Operator Agent*, and awaits the requested change to occur from the Media Agent controlling it.

So if the operator clicks on a *Symbol Rendering Object* type rendering and requests a measurement value for that symbol (say the voltage values of a transformer symbol), then AMEBICA will require a new Media Agent to display it. Instantiating a new Media Agent affects AMEBICA's state and the request for this state change should come through the Operator Agent. Additionally, if the operator requests another representation of a Rendering Object they are viewing (want to change from DIAL measurement to THERMOMETER measurement), this demands a change at the Media Agent displaying that Rendering Object, thus inducing a state change of AMEBICA.

In the following section a more detailed examination of operator induced changes is undertaken.

The Role Of Renderings On AMEBICA State By Operator Induced Changes

This process naturally implies that the rendering itself, be more than just a simple window. Each representation at design time contains knowledge about the ways in which it can be manipulated. So if a operator clicks on the Rendering Object of a symbol for a resistor, for instance, the Rendering Object of the resistor will be able to lookup what other possible Rendering Object forms that resistor can have in AMEBICA. The operator might then decide to view a measurement type Rendering Object of the resistor symbol. This will involve the resistor symbol Rendering Object making a request for a measurement type Rendering Object to the *Operator Agent*.

Therefore each rendering must have two types of knowledge associated with it, first it must know how to communicate with the *Operator Agent*. Secondly, it will have to know not only all the alternative forms it can take, but also all the alternative forms its constituent sub objects (usually Rendering Objects) can take (for instance what other forms the Rendering Objects in the current Representation can take). Again, this is simply added to the sub-objects at design time, in this case the resistor symbol knows it can also be viewed as a measurement.

For communication with the *Operator Agent* the rendering (be it Rendering Object or Representation) must communicate at an agent level, and must therefore have pre-set communication strategies in which it fills in gaps with variables that have been operator selected. For instance, at instantiation time the *Media Agent* will pass a name argument to its rendering, so that the rendering knows which *Media Agent* is responsible for it in the AMEBICA framework. So a pre-set communications strategy requesting change of *representation* is a single string in KQML syntax of the form:

From: [media agent name]
To: Operator Agent
Ontology: *Rendering Object*

Content: (*Representation Change) From:* [current Rendering Object] *To:* [user requested Rendering Object] *Request Type: Operator Request*)

WHERE *italics* represent a design time communications strategy for state change of type representations. The words in (square brackets) are variables that the rendering fills it at run time with variables it obtains from the operators changes.

Each rendering will be able to link operator requests with the appropriate communications strategy and will be able to extract from a operator request the appropriate variables to complete the request.

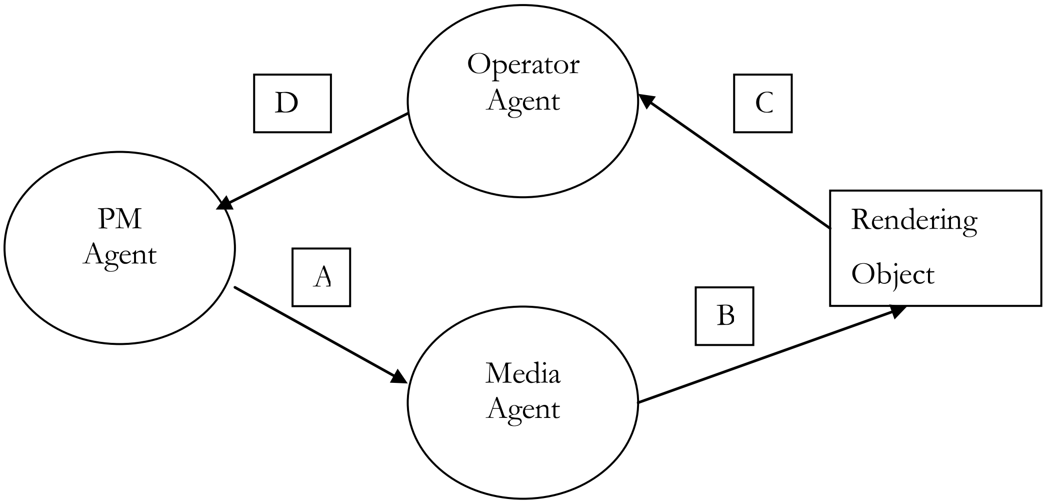


Figure 45 Sequence of Events Leading to Media Agent Instantiation and Operator Request For Adaptation

So we can see the pattern of events that follow a Rendering Object request in Figure 45. If we take these events from the very start we get the following order

1. The PM (Process Model) Agent receives a process event that requires viewing at the interface.
2. The PM Agent instantiates a Media Agent (A) and informs it of the type of request.

3. After talking to the framework the Media Agent selects an appropriate representation. It maps the selected representation and parameter information to an appropriate Rendering Object, which it instantiates and connects to the appropriate streams (B). It also passes in the form of an argument to that object its name, so that the Rendering Object knows which Media Agent is responsible for it.
4. The Rendering Object now receives a request from the operator for a change of representation. The Rendering Object selects the appropriate communications strategy, fills in the gaps and sends the representation change request to the Operator Agent (C).
5. The Operator Agent frames the request and adds any further information, as well as logging the request. The Operator Agent then passes the request onto the Process Model Agent. (D)
6. The PM Agent realises the request if of type *Operator Request* and is also a request for *Representation Change*. It therefore does not need to make any further streams available, and so formulates the request in an appropriate AMEBICA speak form for the Media Agent (A).
7. The Media Agent kills the current Rendering Object and connects up the streams to the new – operator selected Rendering Object. It places the new Rendering Object in the same space as the old Rendering Object.

It might seem redundant for a operator request to go through the *Operator Agent* and *Process Model Agent* to reach a *Media Agent*. However, each of these stages are significant at different times and for different reasons. The Operator Agent will log all requests going through it to get a clearer picture of operator activity. This log is used by the Rendering Resolution Agent later for representation selection. In addition to receiving direct information from renderings it monitors operators activity and inactivity at mouse and keyboard.

The Process Model Agent is notified because the operator request *may* require extra streams, or may trigger events at the process. For instance if the operator selects a different geographical area of a map representation, it will require the Process Model Agent to match that requests to appropriate streams. It may also require the Process Model Agent to launch a new Media Agent to represent the new streams.

Other Architectural Performance Constraints

Time And Operator Adaptation: How The System Adapts To Time And Operator Constraints

The Role Of the Operator Agent when Adapting to Time and to the Operator.

The *Operator Agent* plays a key role in the mechanisms for dealing with adaptations initiated by time delays and operator requests. For example, the *Operator Agent* can act on the operator's wishes (closing a window, making a request to enlarge a window or turn a sound off for instance), by interacting with the *Process Model Agent*. Thus it will adapt the interface as the operator requests it to. Additionally, the *Operator Agent* monitors the operator's activities and pro-actively acts upon them.

So if the operator has been inactive for long periods it might infer that the operator is inattentive. To counter this, it may request that the Rendering Resolution Agent increases the *Evidence* Level, thus ensuring current relevant renderings have accentuated presentation formats thus drawing them to the operators attention.

Role Of Media Agent when Adapting to Time.

The *Operator Agent* can only deal with certain time adaptation problems. One possibility is the changing of a rendering status over time. For example, if a rendering, which represents an alarm, has not been serviced over a period of time it may need to increase its *evidence level* in an attempt to draw the operators' attention to it, and thus get serviced.

The Process Model Agent has within it pre-defined (design time*)* knowledge of how much time an alarm can remain un-serviced, before its priority needs to be increased (thus adapting the representation over time). So when an alarm is received by the Process Model Agent it assigns an appropriate priority level to it, and uses this priority to deduce the allowable time that alarm can remain unserviced. After passing information on priority and stream allocation on instantiation to the appropriate *Media Agent* the Process Model Agent copies this information to its *event list*.

The event list contains within it all the alarms *the Process Model Agent* is monitoring and the times at which each must be checked. When the time arrives that an alarm must be checked, the *Process Model Agent* checks the stream response values for that alarm, if they are still at the same level (that is the alarm is unserviced), it proceeds to *increase* the priority for that alarm to a pre-determined level.

It then informs the *Media Agent* responsible for the unserviced alarm to increase its priority. This sets in motion the normal process of checking with the *Rendering Resolution Agent* and *Media Allocator Agent* for selection of an appropriate rendering and parameters. The only difference being that it increases its *evidence* level by an amount stipulated at design time as appropriate for its time threshold being exceeded.

It passes a Flag to the *Rendering Resolution Agent* indicating that it is making a request for increased priority of an **existing** *Media Agent* and NOT for instantiation of a new rendering. This flag is passed along with the ordered list to the *Media Allocator Agent.* This flag informs the *Media Allocator Agent* that the rendering currently exists at the interface. The *Presentation Agent* is then informed so that when it comes to re-ordering the list it can look for representations that equal the current rendering PLUS ANY extra available screen real estate. Without the flag, the *Media Allocator Agent* would assume the request is for a new event and may actually REDUCE the requesting *Media Agent* to accommodate what it sees as a new event.

This process explained above will vary depending on the level of increase of priority set by the Process Model Agent. If the increase is only slight the system will not change that type of representation only its parameters. The Rendering Resolution Agent will use the increased priority to highlight the representation (using colours or size), and pass this information on to the Media Allocator Agent, which then affects the changes without the need for utilising the

Presentation Agent. If the change in priority is much larger, a drastic change of Representation might be required and the Presentation Agent must be brought into use. The Media Allocator agent indicates to the Presentation Agent that the event is one of type update and must therefore be in the current EXISTING space of the representation. The Presentation Agent will then inform the Media Allocator Agent whether the requested size increase can be performed in that space, using either expansion or overlap. If neither can be done then the Media Allocator Agent applies the normal parameters changes (colour, highlighting and so on), and does not increase the size. If there is room for expansion or overlap then the Media Allocator Agent performs these changes.

Multiple Events Arriving At High Frequency

In a complex process control environment it is normal to expect a large number of signals arriving at the Process Model. In times of disurbance, often, these signals will cause large number of events to arrive at the adaptive system. It is crucial, therefore, that the system is designed in such a way as to handle large numbers of events.

One way to help the system deal with this problem, is to make the systems response time as quick as possible. To increase speed in the system to help it deal with multiple events, there are several mechanisms which adhere to system design but reduce lag. Examples of such mechanisms include

> Within the adaptation system sets of repesentations are passed around. However the system does not pass around the representation objects themselves, rather references in the form of meta-information. This use of meta-data ensures more efficient use of memory and a far quicker system interaction time.
>
> The *Media Allocator Agent* buffers and time stamps all incoming requests. In the case of high numbers of events arriving simulaneously, the requestes are queued in a Vector (mutable array) and re-ordered based on *priority*. The Media Allocator Agent services the

highest piority events first. If several events are of the equal priority it deals with the oldest one first (one that has been waiting longest, based on time stamp).

The Media Allocator Agent also implements time adaptation, so that any request that has been unserviced for a pre-set time will have its servicing priority increased, ensuring it moves up the queue and gets serviced quickly . This priority increase is *just* for the purposes of buffering, once the request is being dealt with by the *Media Allocator Agent* it utilises its original priority.

Any *Operator Request* is dealt with as of being *highest* priority and is put straight to the top of the queue, and is therefore deal with first thus ensuring minimum possible delay.

The *Presentation Agent* keeps a table of all the current *Media Agent* Representations being displayed along with their minimum size parameter information. These tables ensure quick lookup and minimum delay in selection of the interface allocation.

In an alarm flood situation the Media Allocator Agent examines the queue size and the time stamp difference between the earliest and last request. The *Media Allocator Agent* may then have a contingency plan to get through the list quicker. This may involve skipping the expansion schema and just letting windows overlap. This would mean higher chance of being displayed and quicker system throughput.

If the queue becomes larger than a second pre-defined limit then the Media Allocator Agent may require a second strategy. This second strategy is to utilise the representation of lowest priority on either the best or backup list and find the smallest size representation. The Media Allocator Agent will then render this representation along with others of a similar priority. This plan ensures that the information gets to the screen and the queue size is reduced. So, for instance, pure text might be selected for a measurement value instead of a large dial representation. This means the pure information is on-screen available for the operator to examine, but not taking up valuable interface real estate.

If the queue of lower level priority representations is too big then the Media Allocator can take a certain number of them out of the queue and place them into the *non-display* queue. The *non-display* queue is formed out of events whose minimum size representation cannot fit onto the screen and whose priority is sufficiently low to deem it possible to wait for screen space to clear up. These events can be listed in a separate window as being of type *waiting* so that the operator can see which events are waiting to be serviced by the *Media Allocator Agent*. This window can permanently be on screen and be of small size with scrolling window.

The Number Of Active Agents At One Time

The time response of the system is very important in a critical real-time process control situation. Therefore there is a trade-off between having many agents, negotiating intensely to provide the best possible solution, and a smaller yet optimised number of agents which cannot reason as intensively but are quicker. The solution given here is to utilise a smaller number of agents, with streamlined negotiation and optimised reasoning.

In our system a distinction is made between *active* agents, which are agents that work asynchronously and autonomously (Media Allocator, Rendering Resolution Agents), and *passive agents*, which are service provider objects that work synchronously with a client agent (Presentation Agent). To ensure better performance the number of active agents was limited as a trade-off for better system performance.

The Number Of Messages To Be Exchanged And Processed Between The Agents

The greater the number of messages that are processed by the agents (writing, sending, receiving, opening, reading, reacting), the greater the amount of resources that will be spent on communication, and the worse will be the time response. The number of messages is linked to the number of agents, the politics of communication, and the politics of negotiation.

Since there are a limited number of agents, there are a limited the number of messages.

Communication And Negotiation Policies

The nature of the communication/negotiation policies within the agent architecture has an important affect on the system performance. Choosing the incorrect strategy can lead to unnecessary overhead and critical delays in system response. The communication policy adopted has some influence on the negotiation strategy that is utilised, and in critical real-time situations it is important that the number of inter-agent messages should be kept to a minimum.

Negotiation (especially at the *Rendering Resolution Agent* level) is important and must be clearly and carefully structured. A comprehensive, global policy was discarded as a valid strategy as it was too inflexible, and did not optimise system performance. Instead the type of communication adopted depended on the position of the agent in the architecture. The agents have an internal acquaintance model of the other agents with which they interact. To the agent concerned this acquaintance model *is* the entire agent system, and the architecture can be decomposed into groupings of interactions. Thus, the *Process Model Agent* has a uni-cast, bi-directional communication policy with the *Media Agents*, with no negotiation involved (they assess and tag relevant data and pass it onto the relevant agent). As far as the *Process Model Agent* is concerned, the *Media Agent* is the AMEBICA system. This is the case for most inter-agent communication strategies. The Media Allocator Agent uses *broadcast* at certain times to distribute information to the *Media Agents*.

The Complexity Of Communication/Negotiation Between Agents.

Besides the time response, a second difficulty lies in the capacity to define, keep control of and assess the communication/negotiation policies. The more agents involved and the more complicated the communication policies, the more difficult becomes the management of complexity.

This is particularly true when there are several agents working and communicating in parallel. Non-determinism becomes a key issue.

Synchronisation is complex. Moreover, synchronization is bad for time response (it is heavier on CPU resources). Therefore, synchronization is only used when necessary (that is between reasoning agents). Otherwise all other agents communicate and negotiate asynchronously.

Conclusion

This chapter has given a detailed look at the mechanisms behind the key reasoning agents that orchestrate adaptation. An in-depth look has been taken at the internal architecture of the Media Agent, and how it utilises streams to render its representations. Additionally, the Presentation Agent has been explained with reference to how it maintains a current view of interface usage, how it deduces appropriate Representation Spaces and how it interacts with other key reasoning agents.

An examination was made of the Media Allocator Agent, and how it apportions space for representations. A detailed study was carried out on how the Media Allocator Agent determines whether it needs to expand or overlap Representation Spaces. If the Media Allocator Agent cannot allocate, or adjust the interface to contain an incoming representation, then it has several strategies for ensuring the representation gets processed, prime of which is a queuing mechanism.

The Media Allocator Agent has the ability to place associated representations together, using the *Satellite/Source* relationship. At design time one representation is defined as being a source, and associated representations are defined as being satellite. The Media Allocator Agent dynamically on the fly attempts to configure the interface so that satellites are placed in set positions with respect to their source. These pre-set position constraints are defined within each representation.

For large representations (electricity network grid for instance) that are composed of smaller rendering objects (sub-stations for instance), an important aspect of adaptation is the zoom factor and location of the visible window on the larger representation. To deal with this the Media Allocator uses a *parent/child* strategy (where the larger representation is the parent, and the sub-parts are children). Utilising the priority of the children, the Media Allocator Agent dynamically determines the best zoom factor and the location of the visible window.

This chapter has examined how the adaptive framework implements a rudimentary form of consistency checking, and how it deals with user driven changes at the interface. It has also presented a mechanism for implementing time-led adaptation.

Lastly, it has dealt with important conditions that the system must deal with to be successful. These conditions include the system receiving high frequencies of events, the number of active agents and communication/negotiation problems.

The next chapter looks at the experimental results of the implemented system, and how successful it was at fulfilling its goal.

Chapter 10

EXPERIMENTAL STUDY: THE USE OF THE ADAPTIVE PRESENTATION SYSTEM WITHIN THE DOMAIN OF ELECTRICITY NETWORK MANAGEMENT

The Final AMEBICA System

A complete prototype adaptive system was constructed according to the principles outlined in Chapters 8 and 9 using a multi-agent approach. AMEBICA is a complex system and the design and development work was spread over a number of contributors. Figure 46 below shows the

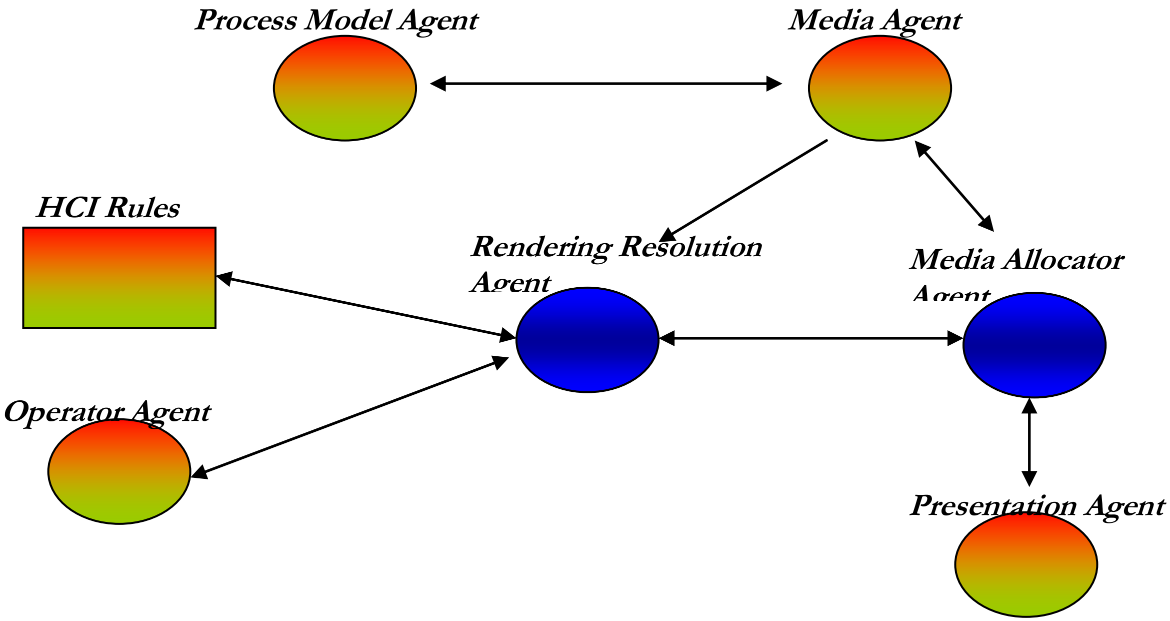


Figure 46 Adaptive System Prototype Contributions

final form of the system and indicates the contribution of the author.

In order to make clear the contribution of the author, Table 11 and Figure 46 show the parts of the system designed by the author in GREEN, the parts built and designed by the author are in BLUE and other contributions (build/test/detailed implementations) are in RED. Thus the overall contribution of the author is as follows

Component	***Overall Design***	***Detailed Design***	***Constructed and Tested***
Process Model Agent	Concept and overall design by author	Detailed design carried out by Softeco	Constructed and tested by Softeco
Media Agent	Concept, overall design and internal architecture by author	Detailed design by Alcatel	Constructed and tested by Alcatel
Rendering Resolution Agent	Concept, overall design and internal architecture by author	Detailed design by author	Constructed and tested by author
Operator Agent	Concept, overall design by author	Detailed design by IFE	Constructed and tested by IFE
Human Factors	Concept, overall design by	Detailed design by	Constructed and tested by

Rules Database	author	author	IFE
Media Allocator Agent	Concept, overall design and internal architecture by author	Detailed design by author	Constructed and tested by author
Presentation Agent	Concept, overall design and internal architecture by author	Detailed implementation by LGI2P	Constructed and tested by LGI2P
Overall Design of Architecture	Concepts, and overall design by author	Detailed design by author	Not applicable
Evaluation of Prototype	Contribution by Author to Evaluation Discussions	Detailed Evaluation by ELSAG personnel	All tests carried out by ELSAG

Table 11 AMEBICA Prototype Contributions

It should be noted that the author designed the agent architecture and the roles and interactions between these agents. The author was also responsible for the adaptation principles adopted, and system concepts such as *generality* and *streaming* as well as the detailed design and implementation of the core reasoning agents, the *Media Allocator Agent* and the *Rendering Resolution Agent*. Finally, the author designed the internal architectures for the Media and Presentation agents. The partners listed in Table 11 carried out the detailed design and coding of the Media and Presentation Agents.

Once built, the system was evaluated within two exemplars – a Thermal power Plant in Spain, and an Electrical Supply Network in Italy. The author contributed to the design and evaluation strategy for both these exemplars but was unable to take part in the evaluation process itself because this was restricted to domain personnel. Only the results of the evaluation of the

Electrical Supply Network (ESN) are given in the thesis. Most of the actual evaluation work was carried out by personnel at Elsag (Designers, Engineers, Operators). The evaluation results described below, show that the system did work, and received qualified approval.

Brief Description Of The Electrical Network Management Exemplar

In the Electricity Network Management (ENM) process, an electricity network diagram is displayed upon the screen with the following information:

> Electricity substations, which may assume different shapes and colours depending on network status and any substation problems.
>
> Electricity equipment, which may assume different shapes and colours dependent on network status and any equipment problems.
>
> Electric connections between equipment.
>
> Status and measurement information: connectivity status, alarms, voltage, current and power, etc.

Such a network diagram is normally very large and only a very limited area can be displayed on the screen at any one time. This narrow bandwidth view of the network diagram can lead to operator error and usually some additional interface tasks. Suppose, for example, that at a certain time the network is in a "steady normal state" and the operator is working on some routine task. Suddenly there is a network disturbance, and some equipment within the network changes its status (i.e. a tripped protection relay opens an electric breaker that was previously closed). In this situation the operator is expected to immediately **pan** and **zoom** the network diagram to the appropriate location in order to try and verify what has happened and why. Additionally, the operator may need to **open a detail window** that shows the composition of the specific equipment in question.

It is possible that whilst the operator is analysing this information, another piece of equipment on the network changes its status. The operator may then assimilate this new information so that the problem can be properly analysed. Since it is possible (even probable) that the two events are related, the operator will need to examine both sets of equipment simultaneously. To examine the status of both equipment faults the operator will need to **pan** and **zoom** the mimic board (the display is called a mimic board although technically it is a screen diagram) once again. A second detailed window will also need to be opened containing lower level detail information concerning the new alarm component.

Most modern graphical interfaces will open these new detail windows in a default screen location. The operator must then move the window and "place" it where it does not overlap important information on the background mimic. Additionally, depending on its location, the operator may be required to resize the window as well. Spatial adaptation was therefore considered to be an important part of the ENM application.

The ENM Scenario

The scenario developed to test the adaptive system describes the effects on the electrical network of an electric storm passing over a series of substations causing progressive outages. The operator needs to execute repetitive and tedious actions in order to understand the nature of the problem. The adaptive system was intended to relieve the operators of these actions and allows them to concentrate on more "effective" problem solving.

Basic requirements

Network In Disturbed State

When an electrical network is in a seriously disturbed state, the system often generates alarms at a rate too high for operators to acknowledge. This often results in the risk that important alarms may not be noticed in time. Therefore, dynamically filtering events in such critical conditions, highlighting the most important ones, and spatially reorganizing the display to bring related events together, can provide vital assistance to the operator.

Alarms Handling

When several alarms are active at the same time but have not been acknowledged, it is often the case that the complete set is not automatically shown on the screen, and therefore the operator has to manipulate the display. When this happens, typically, some important alarms may not be registered by the operator.. The adaptive system addresses this problem by exploiting presentation techniques that can accommodate larger numbers of alarms or by presenting less information on each alarm, e.g. small titles that can be expanded to provide the full alarm information if needed. As a result:

> The operator should be able to separate out major and minor alarm types. A major alarm type is displayed in a certain manner requiring immediate attention. A minor alarm type can be accentuated to enhance presentation but are normally deleted upon acknowledgement.
>
> The system can highlight every alarm condition using a combination of colour, intensity, inverse video, and blinking.
>
> The system provides the means for displaying an indication of the presence of unacknowledged alarms.

Comparisons with the Original System

Faulted Area Identification

On a standard interface during an alarm situation the operator has to locate the alarmed area as quickly and precisely as possible. In the present system this is done by double-clicking on the alarm itself. The system then automatically translates the network diagram so that it is centred round the alarmed element. The zoom level chosen is standard, and this can leave out important, related information.

The adaptive system, in contrast, automatically adapts the level of zooming for the diagram, so as to include, for instance, all other related alarms (geographically, topologically or chronologically) or connected elements. However, at some level the distribution of alarms reaches a complexity where an alternative has to be found - for instance an overview with rectangles/frames that can be taken up separately to view the alarms within the frame.

High Flow Of Alarms (Avalanche)

When the electrical network is in a disturbed state the alarms are generated by the system at a rate too high for the operators to keep up with: and the risk that important alarms are not noticed in time is increased. The adaptive system dynamically filters the alarm events in such critical conditions, or at least attempts to highlight the more important ones. For example if too many alarms are generated within a certain time-frame, instead of scrolling them at an unreadable speed in the alarm window, the system shows only the higher priority ones, making it clear it is doing so and that there are other alarms that remain to be seen.

"Unserviced" Alarms

When alarms arrive, the operator can acknowledge them either individually or collectively. It is often the case that during times of disturbance the operator acknowledges the alarms collectively in an effort to speed the process up. An inherent problem with this method is the frequency with which oversights occur. The adaptive system tries to alleviate this problem by accentuating the presentation of unserviced alarms. It does this by noting how long an alarm has remained unserviced, and, based upon its priority level, presents the alarm again for acknowledgement or makes it more conspicuous.

Mimic Handling Support

The adaptive system adjusts the presentation of information on the mimic diagram by:

Highlighting selected alarms on the basis of their status and the alarm time evolution.

Spatial reorganization of the display in order to keep in the same visual frame the maximum number of the more recent alarms In this way the operator can see important alarms at a glance

Track the alarms movement by panning, zooming, resizing so that the operator can easily follow the time – geographic evolution of the network state/alarms

Resizing of alarm windows

The ENM Scenario Implementation Details

The ENM scenario that was implemented for the prototype acted as a simulation, since the prototype could not be attached to a real process. Therefore, a process simulator was constructed which emulated the scenarios described (moving storm for instance) and the prototype reacted to the process simulator as it would to a real process.

Alarm Management:

The following functions are implemented:

Alarm acknowledgement

Alarm Pages

Display of information related to the network status (moving storm in the case under examination)

Level of priority of each alarm

Scheme Management

The following functions are implemented:

Management of

Substation alarms

Alarm acquisition

Zoom in/out and cantering of alarmed station with continuous panning

Smart opening of detailed schemes (windows embedded)

Smart spatial reorganization to avoid any hiding of critical information

Smart windows resizing

The ENM Scenario – Moving Storm

The figures that follow are actual screen displays of the tracking and spatial reorganization carried out by the prototype following the ENM scenario driven by a simulator.

Stage 1: Disturbance In Substation H

Figure 47 shows the effect of a disturbance in substation H. The system has automatically chosen to display the substation with the maximum attainable level of detail, because it is the only substation in alarm. The station is coloured in red to highlight the appropriate substation on the mimic board.

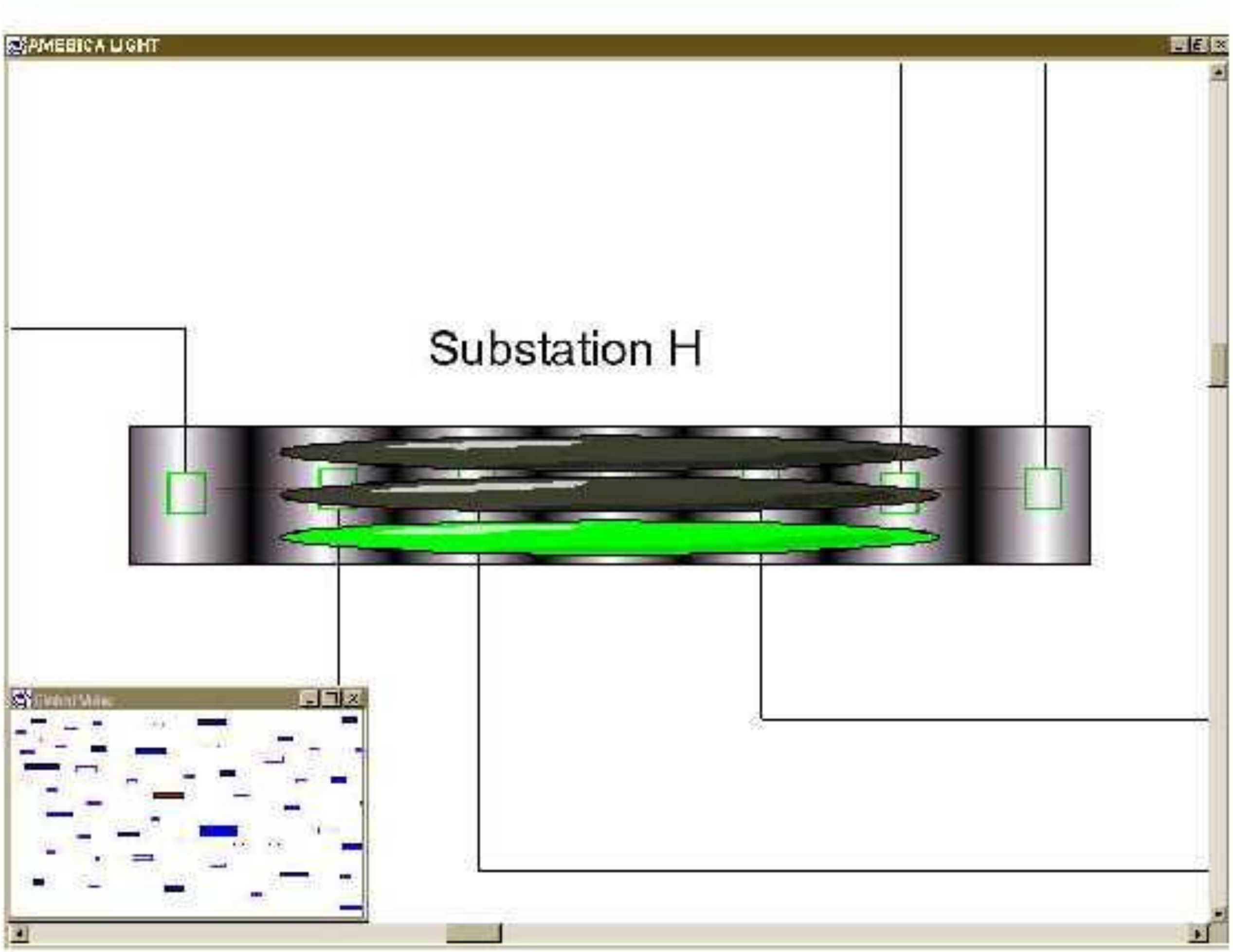


Figure 47 Alarmed substation

Stage 2: Substation H With Detail Window

Figure 48 shows that that the system has now opened a detailed scheme of the substation in a separate window (linked to the synthetic source representation using a blue bar).

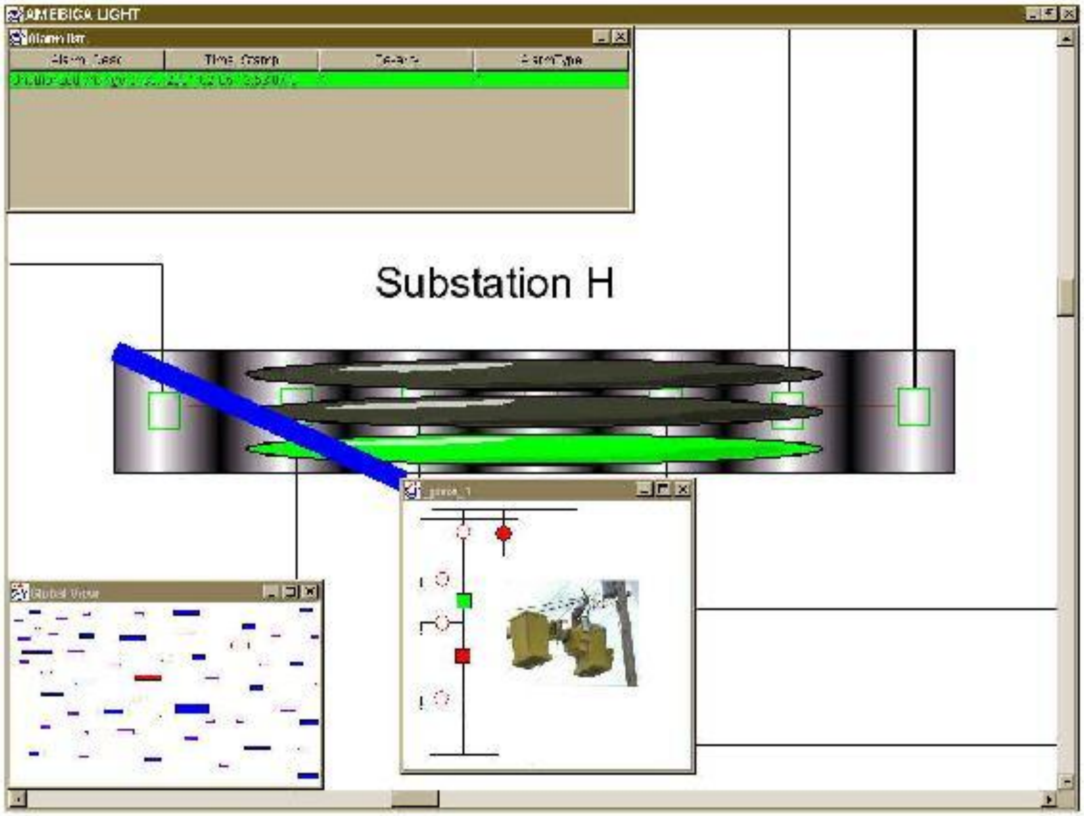
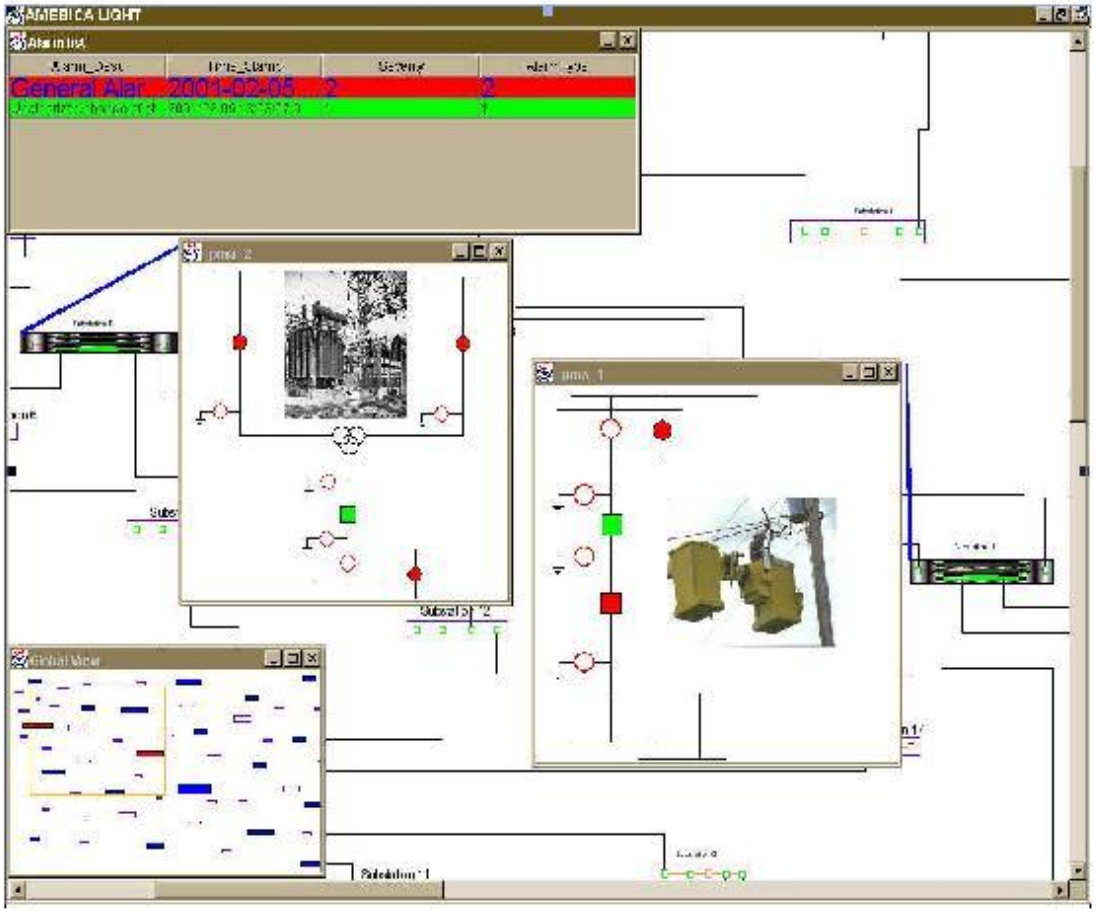


Figure 48 Substation H with detailed window

Stage 3: Disturbance In Substation B

Now a critical alarm appears at substation B (Figure 49). The background Mimic board, is panned and zoomed to the appropriate level of granularity to encompass both substation B and substation H. A detail window for substation B is created and the adaptive system places both detail windows in appropriate locations, and endeavours not to cover important information.

The level of the alarm is shown in the alarm list window, by the font and the colour of the relevant row. If the operator does not acknowledge the alarm within a certain period of time, it will change colour to alert the operator to this fact, and a vocal warning is given if the audio channel is free.

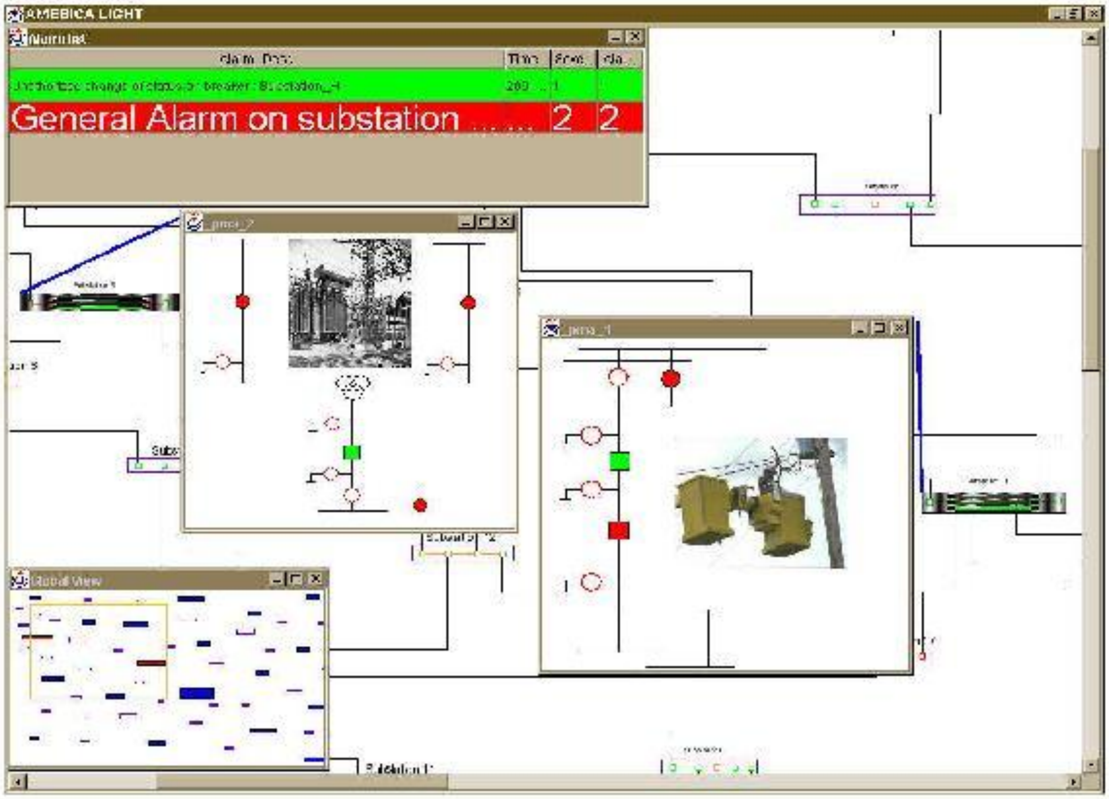


Alarm_Desc	Time_Alarm	Gateway	[illegible]
General Alar	2001-02-05 ...	2	2
[illegible]	[illegible]	[illegible]	1

Figure 49 Critical alarm in B, not acknowledged

Stage 4: Unacknowledged Alarm State

After some time (about a few seconds) the level of alarm importance increases since the operator has not acknowledged it. It is displayed within the alarm list window with a different (larger) font in a different colour (Figure 50). Within the small mimic window a yellow line square is created to show the alarmed zone.

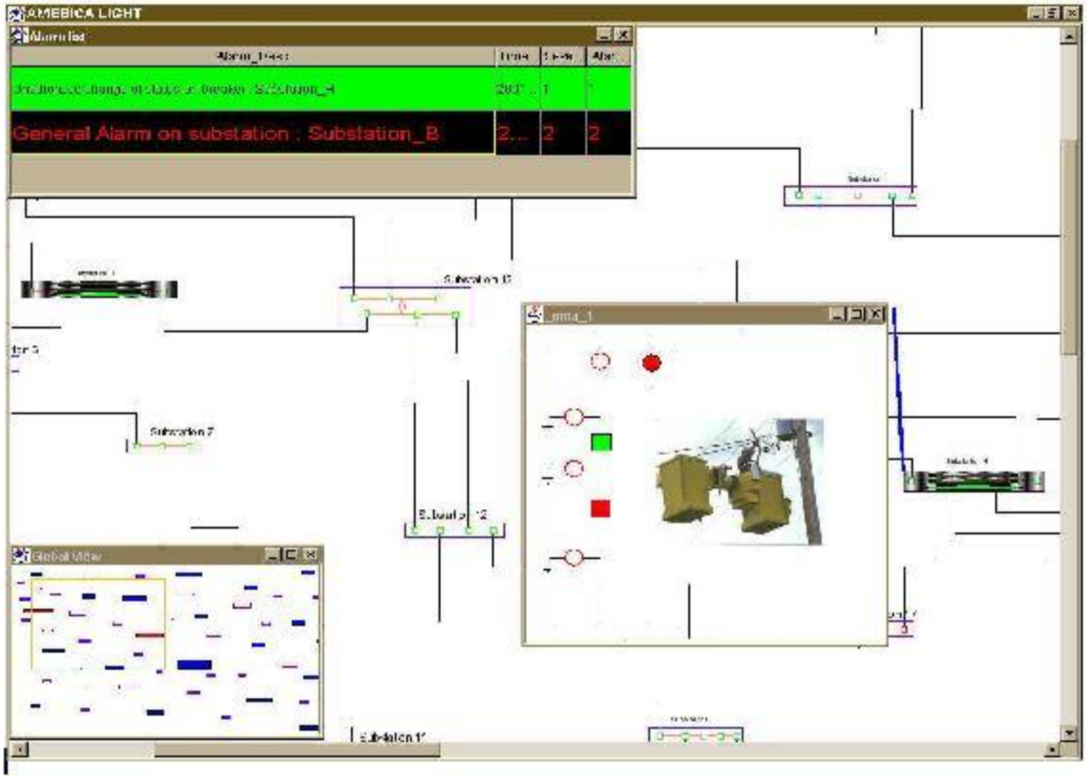


[illegible]	[illegible]	[illegible]	[illegible]
[illegible] breaker : Substation_4	203 ...	1	
General Alarm on substation		2	2

Figure 50 Critical Alarm not acknowledged

Stage 5: Alarm Acknowledgement

Figure 51 shows the critical alarm has been acknowledged and the detail windows disappears, the font is changed to a normal condition.



Alarm_Desc	Time	Seve..	Alar..
[illegible] Substation_H	200..	1	[illegible]
General Alarm on substation : Substation_B	2...	2	2

Figure 51 Critical alarm acknowledged

Stage 6: Use Of Alternative Representation

Figure 52 shows the use of the WEB cam. A critical alarm becomes active within substation H. The system opens a "live" window to increase the consciousness of the operator and to provide them with further "live" details of what visually is happening at Substation H.

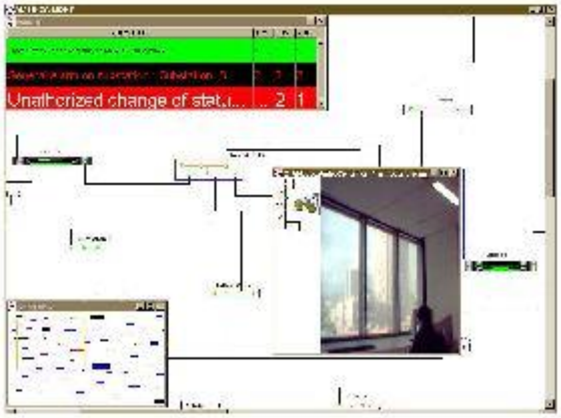
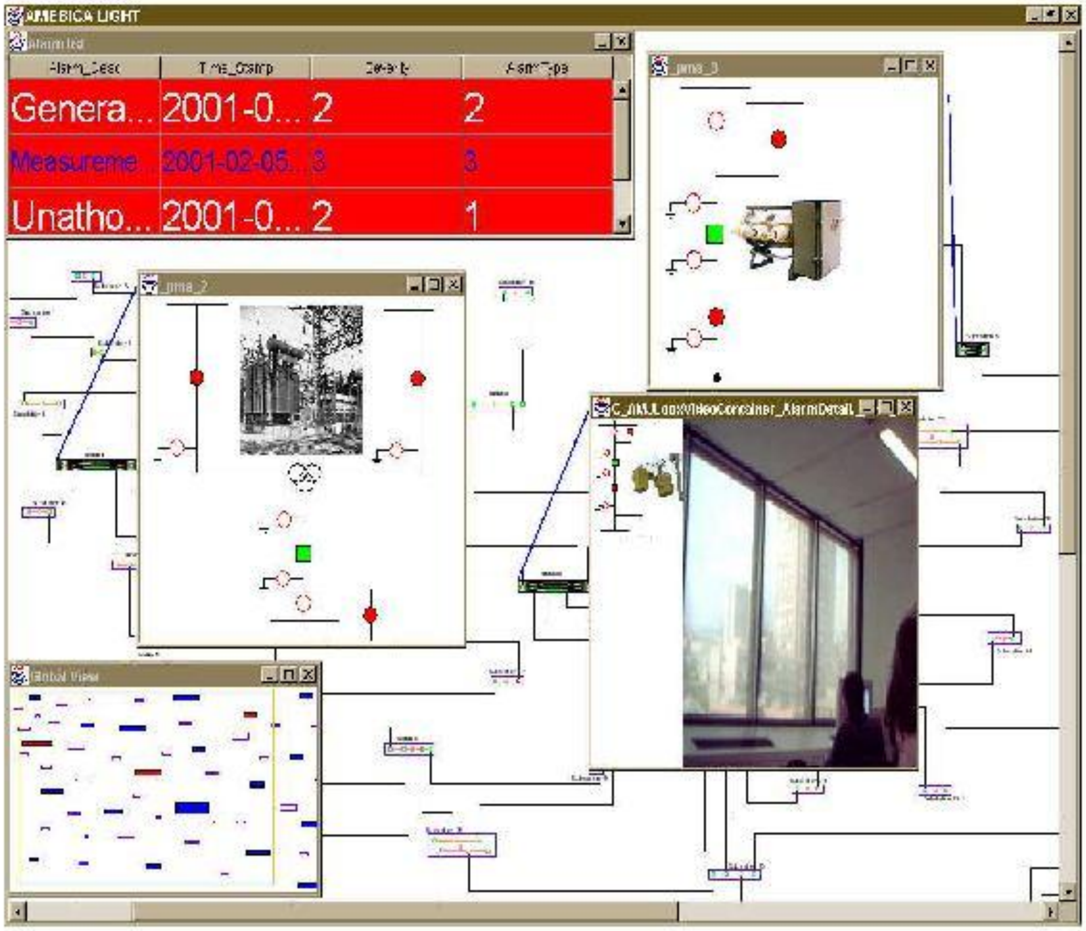


Figure 52 WEB cam utilisation

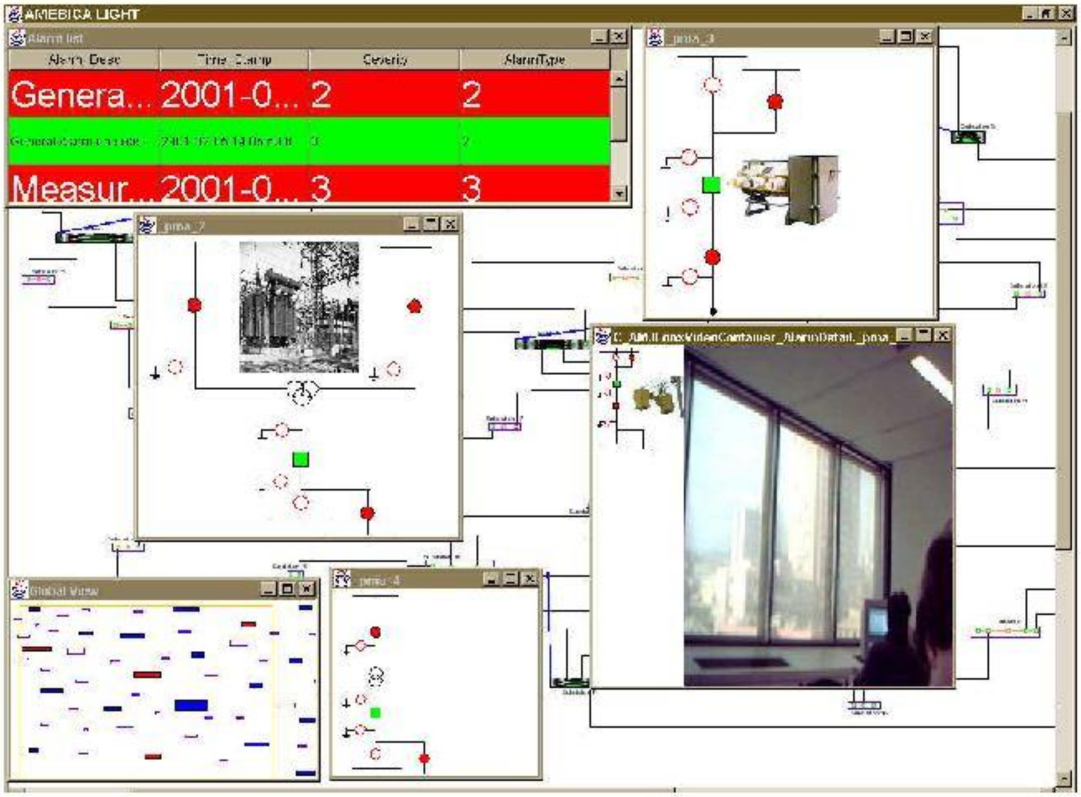
Stage 7: Spatial Management Of The Interface

Figure 53 and Figure 54 show the use of spatial management (1&2). These two displays show how AMEBICA works to best utilise the available screen real estate; it resizes the detailed windows to make it possible to display a complete view of the alarms, windows, and schemes without hiding any alarmed substation.



Alarm_Desc	Time_Stamp	Severity	AlarmType
Genera...	2001-0...	2	2
Measureme...	2001-02-05...	3	3
Unatho...	2001-0...	2	1

Figure 53 Space layout management (1)



Alarm Desc	Time Stamp	Severity	AlarmType
Genera...	2001-0...	2	2
[illegible]	[illegible]	3	2
Measur...	2001-0...	3	3

Figure 54 Space layout management (2)

Stage 8: Adaptation After User Induced View Change

Figure 55 is quite similar to the previous one, but the alarm window has been downsized to avoid any overlapping with the alarmed substation in the upper left corner.

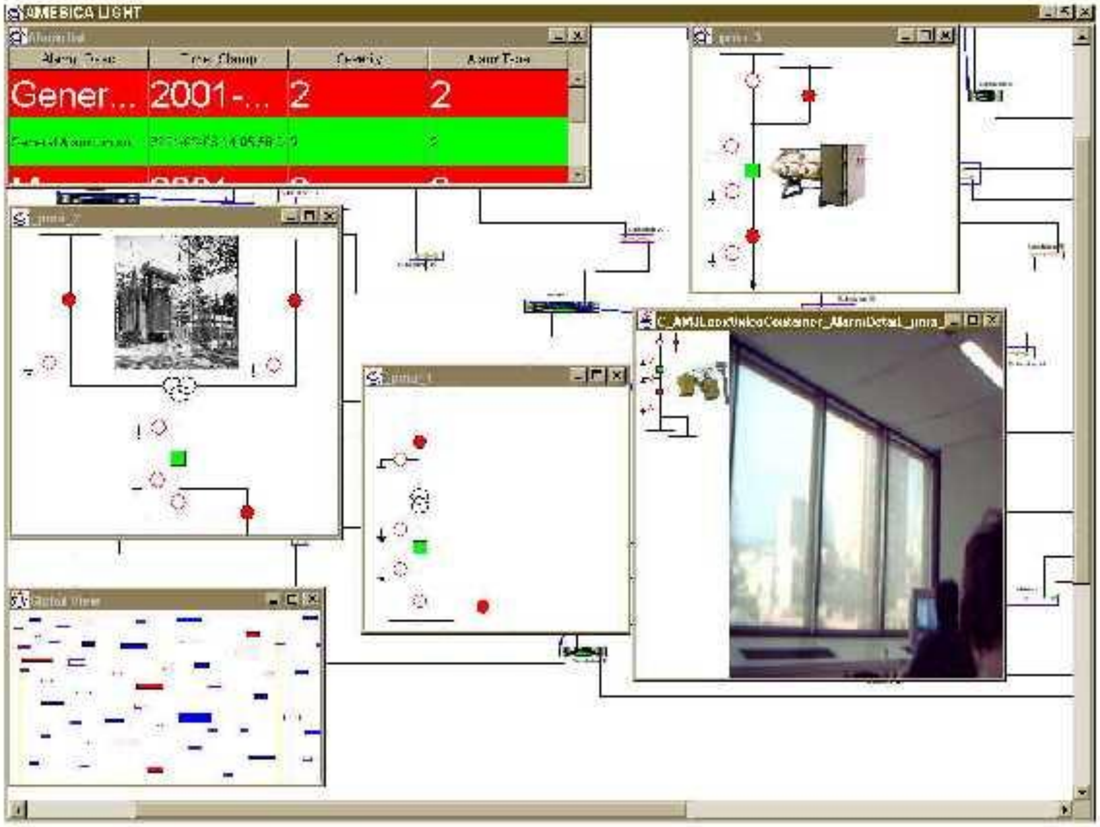


Figure 55 Resizable alarm windows

The Adaptation Matrix

The adaptation matrix shown in Table 12 has been specifically prepared for the evaluation for the “moving storm” scenario An adaptation matrix was developed for the evaluation in order to define in a clear and concise format the performance of the operator and of AMEBICA during the different scenarios. For each cell two descriptions are given; the first in “normal” font represents the status of the operator, the second one, in “italics” font, represents the goal of AMEBICA, i.e. the purpose and the direction of specific compensating functions that are activated.

OPERATOR RESPONSE	***PROCESS STATUS NORMAL STATE***	***DISTURBED STATE***	***HIGHLY DISTURBED STATE***
1 Correct Operation	Checks the status of the network and monitors the measurements	Operator acknowledge the alarms	Operator acknowledge the alarms *Accentuation of the faulted area, display reorganization to show other alarms*
	No Action	*Moves the diagram to centre around the alarm location*	
2 Delayed	Not applicable	Operator does not acknowledge	Operator does not acknowledge the alarms because he is not attentive
	Not applicable	*Display of information is accentuated in order to draw the operator attention*	*AMEBICA modulates the acoustic alarm in order to distract the operator*
3 No control Response	Not applicable	Operator acknowledges some information , but the disturbed status still persists	Operator does not succeed in restoring the “normal” situation
	Not applicable	*AMEBICA highlights alarm presentation on the small mimic and opens windows with alarms*	*AMEBICA highlights the zone affected by the last alarms looking for including the as may alarms as possible*
4 Erratic	The operator is looking at network diagrams and opens and closes different diagrams in random section	Operator does not acknowledge or acknowledges some information , but the disturbed status still persists	Operator does not acknowledge or acknowledges some information , but the disturbed status still persists
	No action	*AMEBICA accentuates alarm presentation*	*AMEBICA accentuates alarm presentation and continues to looking for including as may alarms as possible*
5 Disorganized	The operator is looking at network diagrams and opens and closes different diagrams in random section	Operator does not acknowledge or acknowledges some information , but the disturbed status still persists	Operator does not acknowledge or acknowledges some information , but the disturbed status still persists
	No action	*AMEBICA accentuates alarm presentation*	*AMEBICA accentuates alarm presentation and continues to looking for including as may alarms as possible*

Table 12 ENM Adaptation Matrix.

This adaptation matrix highlights the AMEBICA behaviour as a function of the process status and the operator actions.

Actual Alarm Acquisition Sequences

The following matrices show how alarms are handled within cells in the adjacency matrix. The 3x3 cell in each element of the matrix shows how a current alarm (either in a Waiting state(W), having high importance level (H), or a very important level (V)) is handled when the previous alarm was in one of a similar set of states. The numbers shown in the cells refer to the actions taken by AMEBICA and are listed below each adaptation matrix.

Matrix 1: **Alarm Acquisition**

	Process status														
Operator response	**Normal**					**Very disturbed** High information rate					**Disturbed** Low information rate				
Normal The Operator acquires the alarms	**No action**					**No action**					**No action**				
Delayed The Operator doesn't acquire the alarm, but they are working on previous alarms that have major or equal importance			Previous					Previous					Previous		
			W	H	V			W	H	V			W	H	V
	Current	W	**-**	**-**	**-**	Current	W	**-**	**-**	**-**	Current	W	**-**	**-**	**-**
		H		**1**	**1**		H		**-**	**-**		H		**1**	**-**
		V			**2**		V			**8**		V			**2**
Erratic The Operator doesn't acquire the alarm, but they are working on previous alarms that have minor importance			Previous					Previous					Previous		
			W	H	V			W	H	V			W	H	V
	Current					Current					Current				
		H	**3**				H	**3**				H	**3**		
		V	**4**	**4**			V	**2**	**2**			V	**4**	**4**	
Disorganized The Operator doesn't acquire the alarm and is not working or is working on non alarmed elements															
	Current	W	**5**			Current	W	**5**			Current	W	**5**		
		H	**6**				H	**9**				H	**6**		
		V	**7**				V	**7**				V	**7**		

Table 13 ENM Alarm Acquisition Matrix

ACTIONS BY AMEBICA (TABLE 13)

1. After a while, the Event to be acknowledged is shown in the alarm list window where only the important alarms are listed

2. After a while, the Event to be acknowledged is shown in the alarm list window, the audible console alarm is activated and only the very important alarms are listed

3. After a while, the Event to be acknowledged is shown in the alarm list window, the audible console alarm is activated and only the important alarms are listed

4. After a while, the Event to be acknowledged is shown in the alarm list window, the audible console alarm is activated and only the important and very important alarms are listed

5. After a while, the Event to be acknowledged is shown in the alarm list window and all alarms are listed

6. After a while, the Event to be acknowledged is shown in the alarm list window, the audible console alarm is activated and only the warning and important alarms are listed

7. After a while, the Event to be acknowledged is shown in the alarm list window, the audible console alarm is activated, the diagram is zoomed and centred in order to include all important and very important alarmed elements and the equipment diagrams relating to them

8. After a while, the Event to be acknowledge is shown in the alarm list window, and only the very important alarms are listed

9. After a while, the Event to be acknowledged is shown in the alarm list window, the console audible alarm is activated, the diagram is zoomed and centred in order to include all important alarmed elements and the equipment diagrams related to them.

Matrix 2: **Alarm Handling**

			Process status								
Operator response			Normal			Very disturbed High information rate			Disturbed Low information rate		
Normal The Operator makes controls on the alarmed element			No action			No action			No action		
Delayed After alarm acquisition, The Operator makes works on elements related to the most important alarms			Previous			Previous			Previous		
			W	H	V	W	H	V	W	H	V
	Current	W	-	-	-	-	-	-	-	-	-
		H		-	-		-	-		-	-
		V			**1**			**1**			**1**
Erratic After alarm acquisition, The Operator works on elements related to alarms of minor importance			Previous			Previous			Previous		
			W	H	V	W	H	V	W	H	V
	Current										
		H	**2**			**2**			**2**		
		V	**3**	**3**		**3**	**3**		**3**	**3**	
Disorganized After alarm acquisition, the Operator is working or not working on non-alarmed elements											
	Current	W	**4**			**4**			**4**		
		H	**5**			**7**			**7**		
		V	**6**			**6**			**6**		

Table 14 ENM Alarm Handling Matrix

ACTIONS BY AMEBICA (TABLE 14)

1. After a while, the Off normal list page is shown and only the very important alarms on which the Operator has not accessed are listed

2. After a while, the Off normal list page is shown, the audible console alarm is activated and only the important alarms which the Operator has not accessed are listed

3. After a while, the Off normal list page is shown, the audible console alarm is activated and only the very important alarms and the important alarms which the Operator has not accessed are listed

4. After a while, the Off normal list page is shown, the audible console alarm is activated and all the alarms which the Operator has not accessed are listed

5. After a while, the Off normal list page is shown, the audible console alarm is activated and only the important alarms and the warning alarms which the Operator has not accessed are listed

6. After a while, the off normal list page is shown with all unserviced alarms, the audible console alarm is activated. The diagram is zoomed and centred in order to include all important and very important alarmed elements and their related diagrams

7. After a while, the Off normal list page is shown with all the unserviced alarms, the audible console alarm is activated, the diagram is zoomed and centred in order to include all important alarmed elements and related diagrams.

Matrix 3**: Unsupplied Grid Areas**

	Process status		
Operator response	**Normal**	**Very disturbed** High information rate	**Disturbed** Low information rate
Normal There are no unsupplied areas	**No action**	**No action**	**No action**
Delayed There are some unsupplied areas, but the Operator is working on other network elements	**1**	**No action**	**No action**
Erratic The Operator is working on low priority areas, in the meantime important or very important areas are unsupplied.	**2**	**No action**	**2**
Disorganized The Operator is not working but there are some not unsupplied areas	**3**	**No action**	**No action**

Table 15 ENM Unsupplied Grid Areas Matrix

Actions by AMEBICA (Table 15)

1. After a while, the Off normal list page is shown and only alarms related to important unsupplied areas are shown

2. After a while, the Off normal list page is shown, the audible console alarm is activated, and only alarms related to important unsupplied areas are shown

3. After a while, the Off normal list page is shown with the alarms related to important unsupplied areas, the audible console alarm is activated and the diagram is zoomed and centred in order to include all important unsupplied areas.

The Usability Results

Who Evaluated The ENM Prototype

Network operators and Shift supervisors belonging to one of the "important" End User (ENEL) and by GRS system experts and consultants of Elsag and Softeco/Sismat carried out the evaluation.

The users were organized into an "expert team" (Table 16) whose purpose it was to give an expert evaluation on the prototype and talk through.

Company	Name	Yrs of experience	Main activity
Softeco	Gianni Viano	15	Design of MMI systems and general R&D
Softeco	Marco Dongu	1	Research and design of MMI systems
Softeco	Ugo Moretto	20	Development of automation plants
Company	**Name**	**Yrs of experience**	**Main activity**
Elsag	Alessandra Rognone	9	Design and implementation of MMI for SCADA
Elsag	Daniele Biglino	20	Design of control system for electrical networks
Elsag	Enrico Appiani	13	Design of distributed architecture for automation systems
Elsag	Lauro Mantoani	15	Design of control system for electrical networks
Elsag	Marco Corvi	6	Computer vision and Image processing
Elsag	Marina Rossi	15	Design and implementation of MMI for SCADA
Elsag	Paolo Frugone	10	Design and implementation of control system for power plants and electrical networks
Elsag	Riccardo Moretto	17	Design of MMI and interactive HELP interfaces
Company	**Name**	**Position**	
ENEL	Bargelli	Manager of network maintenance services	
ENEL	Del Caprio	Manager of SCADA system	
ENEL	Massimo Marangoni	Shift head of the control centre	
ENEL	Broccardo	Manager of MV network management	

Table 16 Expert Evaluators

The *evaluation group* were shown the prototype version and were demonstrated the deemed advantages that might be derived from its use. After a short training period, the evaluation group discussed the prototype and were interviewed to discover their impressions.

Overview Of Evaluation Techniques

The evaluation techniques are described below:

Expert Evaluation

This technique employs a diagnostic approach that uses process experts and Man-Machine experts to identify design errors or likely user problems. This method is cost-efficient and provides useful feedback.

Observational Evaluation

This technique involves the direct observation, or indirect monitoring by video cameras, software logging systems etc, of user behaviour at the interface. Much useful data can be collected in this way.

Survey Evaluation

This technique involves the formal collection of data about the operator's subjective impressions of the interface using interviews and/or questionnaires. Data is comparatively easy to collect, and efficient analysis can be carried out.

Talk Through

One of the most popular usability testing techniques is the "talk through". Here test subjects are encouraged to "talk through" as they are performing a task. The key point of this method lies in its ability to show what users do and why *whilst* they are doing it, as opposed to discussing later how they remembered they did it and why.

Usability Evaluation Methodology

The Usability evaluation of the AMEBICA prototype has been based on the scenarios described earlier in the chapter and employed to verify that the approach is consistent with the application.

Elsag's and Softeco's experts performed part of the evaluation. The quality of the results achieved in this way is reasonable since the Elsag/Softeco experts have many years of field experience in the field of Network Management.

They have been involved in the specification, design and testing of MMIs for more than 20 years and have spent many months with real MMI operations in both Italy and abroad. Every effort was made to avoid any potential bias of their judgments. The ENM evaluation plan covered:

Usability (the extent to which the system was used by specified users in achieving specific goals)

Effectiveness (the ability of users to complete tasks using the system, and the quality of the output of those tasks)

Efficiency (the level of resource consumed in performing tasks)

Satisfaction (users subjective reactions to using the system)

Ease of learning (the easiness with which subjects understood the concept and the operations of the AMEBICA console and how easy it was for them to remember its operating procedures)

Error rate (the number of errors that arose from to the subject during a working session, defined in terms of time and the number of test cases)

Quality of the system (the extent to which specified goals were achieved in terms of effectiveness, efficiency and satisfaction)

Actual procedures

The following evaluation techniques were employed informally:

Focus group: composed of experts from the GRS development team of GRS, the AMEBICA development team and End User operators.

Talk through: carried out by two/three experts on a group of people of around ten.

The end-users were questioned on:

Their opinions on AMEBICA capabilities to help the operator fulfil their task

Their understanding of what AMEBICA is doing

Their opinion about AMEBICA's capability in making it faster and easier to navigate through the displays

How efficiently they dealt with alarms

Which improvements they felt AMEBICA offered with respect to traditional Human Interface systems.

Table 17 lists the expert participants and their experience:

Company	Name	Yrs of experience	Main activity
Softeco	Gianni Viano	15	Design of MMI systems and general R&D
Softeco	Marco Dongu	1	Research and design of MMI systems
Softeco	Ugo Moretto	20	Development of automation plants
Company	**Name**	**Yrs of experience**	**Main activity**
Elsag	Alessandra Rognone	9	Design and implementation of MMI for SCADA
Elsag	Daniele Biglino	20	Design of control system for electrical networks
Elsag	Enrico Appiani	13	Design of distributed architecture for automation systems
Elsag	Lauro Mantoani	15	Design of control system for electrical networks
Elsag	Marco Corvi	6	Computer vision and Image processing
Elsag	Marina Rossi	15	Design and implementation of MMI for SCADA
Elsag	Paolo Frugone	10	Design and implementation of control system for power plants and electrical networks
Elsag	Riccardo Moretto	17	Design of MMI and interactive HELP interfaces

Company	Name	Position
ENEL	Bargelli	Manager of network maintenance services
ENEL	Del Caprio	Manager of SCADA system
ENEL	Massimo Marangoni	Shift head of the control centre
ENEL	Broccardo	Manager of MV network management

Table 17 ENM Expert Participants

Use Cases

Preparation of use cases was crucial for an effective evaluation of the adaptive system. They were derived from a comprehensive analysis of the system targets and experience gained during development. They were also derived in part from in depth discussions with the GRS developers and, in some cases from the feedback of field operators currently using GRS for EMS and DMS functions. Use cases are used to:

> Evaluate the overall usability of the system as well as its effectiveness, efficiency and the subjective opinion of the operator.
>
> Verify the performance of AMEBICA dependent on the perceived behaviour of the operator.

Usability Evaluation

Usability Criteria

Criteria	Method	Measure	Recording format
1. Information presentation	Guidelines/ check lists Observation Interview	Score on check lists Score on pre-specified observation notes Score on interview engine	Cross marks Notes
2. Performance	Talk through Expert judgment Interview	Percentage of goals successfully achieved/ successfully completed tasks Score on interview engine	Audio-recording Cross marks Notes
3. Error rate	Expert judgment Observation	Number of errors committed	Cross marks Notes
4. Situation awareness	Interview Expert judgment	Score on SA questions	Cross marks Notes
5. Mental workload	Interview Observation	Score on workload questions	Notes Cross marks Notes
6. Feeling of control	Interview	Score on control questions	Cross marks
7. Satisfaction	Interview	Score on satisfaction questions	Cross marks

Table 18 illustrates the criteria used for usability testing:

Table 18 Criteria for evaluation of the prototype

	Responsible	**Personnel to take part in test/evaluation**	**Time to conduct test (app.)**
Focus group	End users	Design Group	1-3 hours
Developing scenarios	End users	Process experts	
Talk through	End user or human factors	End users (operator)	2 – 3 hours
Observation	End user or human factors	End users, Process experts, human factors	2- 3 hours
Interviews	End user or human factors	End users, design team	3 hours
Check list	End user or human factors	System designer, human factors	4 hours
Analysis & documentation	End user or human factors	End users, design team, human factors	4 weeks

Table 19 Estimated resources for various parts of the evaluation

Test plan

The evaluation tests were carried out in two sessions:

Session A held in Genoa with Elsag and ENEL.

Session B held in Milan with CESI (ENEL)

The session were mainly based on the moving storm case and included vocal input /output capabilities

Each session included:

Organization of the personnel into focus groups

Training on AMEBICA

Explanation of the target to be reached

Explanation how interview will be conducted

Presentation of the evaluation section

Interviews and talk through

Comparison of information

Debriefing

Analysis of data

Interviews were based on scores; where YES/NO questions were used to test very specific and objective questions such as:

Is the colour right?

Is the size right?

Results Of The Workshops

Quantitative results

Genoa Workshop

This section reports on some of the results (Table 20) derived from the forms filled by the participants to the two workshops held in Genoa in Elsag and in ENEL respectively. The full set of results can be found in Appendix C.

Interviews

(1)Very Bad – very good (2) False – True (3) No – Yes (4) Very little– A lot (5) Very difficult – very easy (6) Little – Much (7) Bad – good

Question	1	2	3	4	5	6	M	Comments
Part 1							**4.9**	
Display of information							**5.2**	
Which is your overal impression of the system? (1)	5	3	4	6	7		**5**	Idea is good, but excessive automation should be avoided
Do you think the system provides information about the process state in a good way? (2)	6	5	5	6	7		**5.8**	An "operator guide" should be available
Does too much information exist in the display? (2)	5	6	3	3	4		**4.2**	Information amount should depend on the process and adapt to it
Do you find the way of presenting the information logical? (3)	5	6	4	5	7		**5.4**	Generally yes
Is it easy to prevent errors? (5)	4	2	-	4	-		**3.3**	
Efficiency							**5.5**	
Does the system provide you with a good overview of the process? (6)	4	3	6	5	7		**5**	Not enough information to give an assessment
Is the Interface Pleasing to Use							**4.9**	
Do you like the way information is presented? (6)	5	3	6	7	5		**5.2**	Not enough information to give an assessment
Do you feel in control of the system? (6)	4	4	-	6	-		**4.7**	Not enough information to give an assessment
Part 2							**5.2**	
Alarms							**5.2**	
How easy is it to diagnose faults by using alarms? (3)	5	5	6	6	6		**5.6**	
Is the alarm system making you attentive when deviations occurs in the system? (3)	5	5	6	7	6		**5.8**	
Are the alarms presented consistent with other information on the screens? (7)	5	3	6	6	6		**5.2**	
How do you evaluate presentation of alarms? (3)	3	2	6	6	6		**4.6**	

Are the alarms organised so that natural relationships between alarms are shown? (3)	-	6	7	7	6		**6.5**	
Do you believe that multimedia provides good support fro the network control? (3)	-	6	7	7	6		**6.5**	Some people believe its too expensive

Table 20 Workshop Results

The Evaluation Checklist

1 = never 2 = sometimes 3= almost always 4 = always

Table 21 Evaluation Checklist

Question	**1**	**2**	**3**	**4**	**5**	**6**	**M**	**Comments**
Section 1: Clarity of the representation					**3.1**			
1. Is important information highlighted on the screen?	-	3	3	3	-		**3**	
2. Does information appear to be organized logically on the screen?	3	3	3	3	-		**3**	
7. Does the screen appear uncluttered?	-	-	3	4	3		**3.3**	
9. Is it easy to find the required information on a screen?	3	2	3	4	-		**3**	
Section 2 Functionality					**3.2**			
2. Does each screen contain all the information that the evaluator feels is relevant to the task?	3	3	3	4	4		**3.4**	
6. Does the system help the evaluator to understand the state of the process?	4	3	-	4	4		**3.8**	
8. Does the nature of adaptation ensure that the evaluator is warned of a process disturbance in time?	4	3	3	3	2		**3**	
9. Are the adaptation strategies consequent?	4	3	3	3	2		**3**	
10. Is the media chosen for adaptation appropriate?	4	3	3	4	3		**3.4**	

Question	1	2	3	4	5	6	M	Comments
14. Does the system ensure that important information is presented for the evaluator at appropriate times?	3	2	2	4	-		**2.8**	
Section 3: Alarms and error messages				**3.1**				
1. Does the system clearly warn the evaluator about a disturbance or a deviation from a normal situation?	3	2	3	4	4		**3.2**	
2. Is the alarm of such a nature that it interrupts the evaluator from what he is doing?	2	3	1	3	4		**2.6**	
Section 4: Information Feedback				**3**				
1. Are instructions and messages displayed by the system concise and positive?	4	4	4	4	-		**4**	
4. Is it clear what actions the user can take at any stage?	3	2	3	3	-		**2.8**	
10. Do alarm messages inform the evaluator about the priority and nature of the deviation?	3	1	4	4	-		**3**	
11. Do alarm messages guide the evaluators' initial actions?	2	1	3	3	-		**2.3**	
Section 5: Consistency				**3**				
3. Are icons, symbols, graphical representations and other pictorial information used consistently throughout the system?	3	3	3	4	-		**3.3**	
8. Is the method of selecting options consistent throughout the system?	-	2	2	4	-		**2.7**	
Section 6: Compatibility				**3**				
7. Does the organization and structure of the system fit the user's perception of the task?	3	2	2	4	4		**3**	
8. Does the sequence of activities required to complete a task follow what the user would expect?	3	2	-	4	-		**3**	
10. Does the system support the evaluator so that the probability of conducting errors is minimized?	3	3	2	3	3		**2.8**	
Question	**1**	**2**	**3**	**4**	**5**	**6**	**M**	**Comments**
Section 7: Usability problems				**2.8**				
1. Working out how to use the system	3	2	3	4	3		**3**	

Question	**1**	**2**	**3**	**4**	**5**	**6**	**M**	**Comments**
2. Understanding how to carry out the tasks	3	1	3	4	3		**2.8**	
4. Finding the information you want	2	1	2	4	3		**2.4**	
5. Too many colours on the screen	3	2	3	4	3		**3**	
6. An inflexible, rigid system structure	3	2	3	4	3		**3**	
10. Unexpected actions by the system	3	2	3	3	3		**2.8**	
11. An input device which is difficult or awkward to use	3	3	3	4	3		**3.2**	
8. Having to remember too much information while carrying out a task	2	1	3	4	1		**2.2**	

1 = Very unsatisfactory 2 = fairly unsatisfactory 3 = neutral
4 = fairly satisfactory 5 = very satisfactory

Questions	**1**	**2**	**3**	**4**	**5**	**6**	**M**	**Comments**
Section 1: Clarity of the representation								
How do you assess the system??	4	4	-	5	4		**4.3**	
Section 2: Functionality								
How do you appraise the system in terms of functionality	5	4	-	5	4		**4.5**	
Section 3: Alarms and error messages								
How do you estimate the system in terms of alarm and messages presentation?	4	3	-	5	4		**4**	
Section 4: Feedback								
How do you assess the system in terms of feedback	4	2	-	5	-		**3.7**	
Section 5: Consistency								
How do you assess the system in terms of consistency?	4	3	-	5	-		**4**	
Section 6: Compatibility								
Which is the level of compatibility of the system?	5	4	-	5	4		**4.5**	
Section 7: Usability problems								
How do you evaluate the system usability?	-	-	-	-	-		**-**	

Average = 4.2

Table 22 Evaluation Conclusions

General questions

Questions	**Replies**
Which are the best aspects of the system?	The capability to exploit multimedia to display information
	Use of vocal commands can be of some help.
	The possibility to focus on the solution of a problem without worrying to look for the specific information. The capability to interact simultaneously via manual and vocal input The optimal spatial reorganization of the display
	Graphic interface and autoadaptivity Alarm priority pointed out by colours

Which are the worst aspects of the system?	Difficult to say without an actual AMEBICA console
	Iconic presentation of the alarms might be useful (instead of alarm lists)
	Maybe the excessive level of intrusion of the system
	Colour assignment might be changed
Which improvement do you suggest?	Multimedia features might be confusing for some operators
	Some operators don't want an excessive reduction of information. Experienced operator may wish to have al the information at a glance (especially in some instances: relay protection for instance).
	Auto adaptivity should be customisable for each operator
What do you like to add?	All functionalities are enough covered
	Include a Context sensitive HELP function.
	Capability to export information to field maintenance crews

Table 23 ENM Workshop General Questions

ENEL General questions

Question	Replies
Which are the best aspects of the system?	Immediacy of alarms
	Multimedia as complement to auto adaptivity to support operator
	Operator guide
Which are the worst aspects of the system?	No tool to include different procedure (on line)???
	Too much information in relation to the workload
	Sometime difficult to understand
Which improvement do you suggest?	Selective filtering of information
	Try to simplify the operating procedures Reduce effects Operator customisable
	Improved readability
What do you like to add?	On line configuration of the AT network
	Possibility to trace the execution of standard ENEL command procedures (for example load transfer)
	Check of default operations

Table 24 ENEL General Questions

Qualitative results

Workshop in Elsag

Presentation Of Workshop Requirements

The following comments were made:

Vocal input helps to have different activities performed in parallel

Volume increase is not welcomed

Presentation Of The Moving Storm Scenario

The following comments were derived:

The system must not be intrusive

No consensus on the automatic cantering and zooming (some people like to give more freedom to the operator. At least one person did not like the automatic repositioning of the windows and the related automatic panning.)

Operators will need sufficient experience to fully exploit the multimedia capabilities

It is stressed that operator must not be confused by "non deterministic" display methods. It is clarified that the adaptive system is something akin to playing chess, the rules are deterministic, but the combinations available are enormous.

One improvement would be to reduce the speed with which the windows are re-positioned or resized (operator might miss something)

The implementation of a "undo" mechanism, with which the operator can automatically uncheck the steps of a procedure, would be useful.

It is not an easy task to correctly ascertain the behaviour of an operator

There should be support for the system to modify the amount of information displayed on the screen dependent on the experience of the operator.

Overlapping alarms should be reduced in size.

Automatic zoom might be confusing some time

Auto-adaptivity should intervene when operators appear confused

Moving Storm

The following comments were given:

In case of serious fault on the HV network, only "root" alarms should be displayed.

It is important that only network critical alarms top the alarm list, there is no need to track alarms chronologically without reference to their importance.

Video images are useful only for a few substations ("strange" substations in ENEL's terminology)

Workshop in ENEL (Milan)

A general initial feeling of conservationism on the side of the final users was quickly replaced by a much more collaborative attitude, finally resolving into enthusiastic and sincere offers of collaboration for possible future developments of the technology. The following comments were given:

A **correct and gradual approach to the operator** should be undertaken in the beginning, in order to avoid outright rejection of the new functionality. This point has been strongly stressed by the people in charge of electrical operation.

The idea of an Auto adaptive Multimedia Interface was deemed very promising.

The potential to **interact vocally** with the system was deemed very promising. This capability could be used, for instance, to display portions of the network or specific diagrams, instead of performing a time-consuming "search" operation.

They were also very interested in the use of particular icons/symbols and graphical representations to **highlight important events**.

The introduction of a **more probabilistic approach** in judging network events and solutions has been suggested by the ENEL research people.

One aspect was deemed to be very encouraging and potentially useful and effective. That is the ability of the system to accentuate unserviced alarms. If **an alarm has been acknowledged, but not properly dealt with** for a set period of time, the same alarm could be automatically brought *back again and again, with increased significance to* the attention of the operator. The fact that an operator acknowledges an alarm without really "seeing" it has been reported as one of the possible causes of prolonged power outages, and the control system should take proper action to ensure that the operator has not been distracted. The fact that the operator has "properly dealt" with the alarm is not easy to establish, but a few basic rules could be applied: the fact that the operator has opened windows or required information somehow related to the elements involved outage could be considered a reasonable clue that he/she has actually *seen* the problem.

Performance Based Regulation Criteria, which in recent Power Authorities' legislation have sought to implement rules that guarantee a good quality of service, could provide a very strong motivation towards delivering proactive control systems, which "monitor" the actions (on non-action) of the operators, in order to insure prompt responses to alarm situations.

Eventually this might generally be considered as the main reason why control systems should eventually be improved. Every Utility should have as its prime goal guaranteed continuity of service to its customers and Performance Based Regulation Criteria actually multiply this need, as they strongly penalize power outages that exceed certain given maximum length and frequencies.

Conclusions

The AMEBICA project was a collaborative effort between a mixed consortium of academics and industrialists. However, the author produced the adaptive architecture and the detailed design of the agent system as well as the coding and development of the key reasoning agents.

This chapter has examined the adaptive system used in the domain of Electricity Network Management (ENM). The traditional system's flaws were listed and used to demonstrate in which areas the adaptive system could be useful. A scenario was produced to test the adaptive system on, that of a moving storm.

The system produced was tested on a series of experts and end users, the results of which are given in this chapter. The overall results were good; most participants saw the value of the system, although they indicated certain refinements might be needed.

Chapter 11

CONCLUSIONS AND RECOMMENDATIONS

Introduction

Adaptive Multimedia Interface Research is a recent, and currently very active research area. Approaches differ through the ways and means used to obtain adaptation (and even in the definition of what adaptation actually is). Examples have been given of an Automated Multimedia Authoring and Multimedia Presentation Tools approach, the development of intelligent interfaces for process control for the nuclear power industry and the PROMISE project, which provided dynamic choice of media to operators at runtime. Many approaches have adopted a Knowledge-Base Systems approach.

All the above works are based on classical Artificial Intelligence techniques. Such techniques are powerful when action must be decided upon and taken in the context of a consistent and monotonic world. However in many process control situations (like our proposed industrial applications) operator awareness, system state and context are continuously changing and are sometimes in contradiction with each other. Any multimedia interface in this environment needs to achieve a balance between these three components and be flexible enough to adapt to changes in the relative importance of each.

The approach adopted, for the system presented in this thesis, achieves this balance by driving adaptation from a combination of the process and operator state. This method is novel and has provided an important model of what drives adaptation.

Additionally, the system presented in this thesis fulfils the main requirements of an adaptive system. It tries to reduce the need of the operator to request help, to anticipate the operators' needs and to reduce the frequency and magnitude of operator errors. Since there are not usually enough interface resources to achieve all these goals, the system deals with this using a priority approach.

The AMEBICA adaptation principle provides adaptation in two main forms

> On-line selection of a rendering from a set of possible representations together with the parameters of that representation (we call this a flexible mapping)
>
> Spatial adaptation of presentation, where the system layout manager attempts to make the display as organised and clear as possible (we call this spatial adaptation).

Our on-line selection approach differs from traditional (rigid) interfaces in that interface mappings are chosen at run-time from a set of defined mappings. This is in contrast to many current approaches where mapping decisions are made at design time and then fixed (usually with a one-to-one correspondence). These rigid mappings are always a compromise. Although our approach still requires design work to be completed on the nature of the representations, many alternative mappings are retained from the design process. The most appropriate selection can then be made at run-time taking into account the current context. This flexible mapping approach involves run-time reasoning between a set of predefined alternatives.

The second form of adaptation uses run-time reasoning to create new instances of adaptation. The domain here is spatial control of presentation. It is generally accepted that as the complexity of an application grows, the operator spends a significant part of their time arranging and re-sizing windows on the screen to suit the current task/context, in addition to acting upon the information present in the windows. Experiments have shown that task completion times in windowed systems are often longer than in non-windowed systems due to the time spent in window arrangement (Bury 1985). So, although multiple windows reduce the user's short-term memory load, they often impose an additional management workload on the user. Our system attempts to alleviate this problem by continuously reasoning about the spatial layout and making on-line adjustments in order to maintain clarity and important relationships.

The system has achieved the following:

> Paying attention to the current context and using it to control the behaviour of the interface.
>
> Reacting to unanticipated events. This is very important when trying to reduce operator overload in emergency situations.
>
> Interrupting actions. When the system state changes, it may be important to be able to interrupt on-going actions in order to perform other actions that have increased in importance. Interrupted actions should be recorded and perhaps can be resumed later.

It is also important that during such spatial adaptation processes, information should be moved as little as possible. Thus, in the AMEBICA system, adaptation has been bounded so that usually adaptation is only allowed before information is placed on the screen. Generally speaking, once upon the screen, information should not be subject to large displacements unless absolutely necessary. AMEBICA will also place relevant information as near as it can to other information of the same type, and will attempt to select the best representation for that information in the current context (using flexible mapping).

The system tries to solve many of the problems inherent in mission critical systems. Such problems arise from the increased complexity of the controlled system, the need to react in a timely fashion to critical problems, to ensure maximum reliability, and finally the need to manage the system so that its functioning at its optimum economic point at all times.

Operators, in modern process systems, are increasingly experiencing difficulties handling the huge quantities of real time information that are critical elements of their job. Their task is not aided by the added responsibilities placed on them by the economic consequence of their actions.

Tools that relieve part of this responsibility and help the operator discovering and discriminating the most essential information from the many thousands of variables available are of primary importance.

General System Overview

General System Successes

This section details the major successes the adaptive system has achieved. The final system achieved most of its early goals, and strongly adhered to the vision dictated by the initial conceptual architecture.

Streaming

The streaming concept was a vital constraining element of the system design. Without the ability of the system to handle signals in real time, the system would have been inoperable. In critical Process Control situations, such delays are unacceptable. Thus, the basic structure of the architecture was dictated by the constraint that it should incur no delays, and yet reason in a precise manner about the nature of the adaptation it would produce. This resulted in a difficult trade-off between reasoning time, and the ability of the system to operate as swiftly as possible. It was clear that this problem could not be solved purely through the use of raw processing power, or through an increase in available system resources. The situation required a conceptual mechanism that rendered information in a highly optimised manner. This mechanism was the *streaming concept*.

The *streaming concept* was founded on a basic tenet decided upon early in the inception of the system. This tenet stipulated that process signals should pass straight through the system and be rendered instantly at the interface, in the form of a default representation. The system would act on the meta-data provided by the Process Agent as it registered the signal. Only after this would the system then decide upon the form and magnitude of the adaptation required, and performs its adaptation on the default representation, and other representations if necessary. The difficult decision faced at conception was deciding whether the operator might be confused by a default representation appearing on screen, and then potentially changing its form and size (and perhaps location). However, this problem was deemed less important than the operator not receiving critical data immediately, even if the data was not in an optimised form. In fact upon completion it was discovered that the actual adaptation system was sufficiently rapid that the operator usually did not notice, or was unaffected, by the change.

Generality

Another important cornerstone of the architecture was its ability to be self-contained yet customisable so that it can operate in a variety of different domains. The ability of the system to be generic and applicable to different types of domains will mean significant savings in the development costs of producing a fully adaptive interface system.

This allows designers to load the system with a set of customised representations, and inform the system of the constraints under which those representations should be considered for adaptation. They must also translate AMEBICA system calls to interface calls, and these instructions must be converted to platform/process dependent calls that will be rendered on the applied platform.

Although all the elements required to make the system generic were developed and implemented in the prototype, the prototype itself was not fully generic, because it could not be attached to a real Process Control interface. Instead, a custom Java based interface (using JLOOX) was constructed which AMEBICA manipulated. It proved the principles and worked well.

Use Of Multimedia

The adaptive system utilised a wide variety of different media. Both the audio and visual modalities were used. Within the graphical modality, static and dynamic representations were used by the system for presentation purposes. Extensive use of text, animation, symbols, dynamic graphics and video were implemented in the prototype successfully. Audio was used in the form of tones and voice.

The system also incorporated speech recognition capabilities to allow the operator to interact with the system on a variety of levels. Redundancy was extensively used to re-enforce the operator's view of the interface, in terms of accentuating certain representations.

The system also allowed flexible configuration of representation parameters. So representations could be altered on the fly to set such aspects as colour, size, animation and form of any visual representation.

Successful Production Of Two Exemplar Prototypes

The adaptive architecture was customised and tested on two exemplars the ENM (Electricity Network Management) process and the TPP (Thermal Power Plant) process (not reported in this thesis). Both exemplars were customised with relatively few problems by expert industrial developers in those areas. The usability studies for both domains indicated very positive results, both by end users and by system designers.

One of the most positive aspects of the study on these two exemplars was that despite being totally different in terms of interface format and requirements, the adaptive system operated effectively on either.

The Adaptation Matrix

The Adaptability Matrix has already proved to be enormously useful in defining the adaptability conditions and responses in the two AMEBICA industrial domains. It provides future designers of adaptive systems a framework with which to match appropriate actions to particular well-defined contexts. It also enables designers to understand the complicated nature of adaptation, and apply their expertise in a well-defined manner

However a number of problems remain. The most serious of these is the recognition of the operator situation. How does one distinguish between a delayed response and an erratic one? There are many cases where the distinction is obvious but there are others where such a distinction will be difficult. Fortunately, in most Process Control environments there exist a set of procedures the operator must follow in a variety of different situations. These guidelines provide a very useful basis for determining the operator state. If the system can deduce the current context, and cross-reference the relevant procedures for that context it can then compare the operators current actions, with those that should be done according to the procedures. Thus, the system has a reasonable chance of deducing whether the operator is performing actions in the wrong order (erratic), or if they are performing the right actions according to the procedure, but taking longer than is recommended (delayed). This situation is specific to a closed loop system such as the Process Control domain. Determining operator actions in an open loop system is far harder.

Another way of attacking the operator state recognition problem would be to record interactions and compare with operator perceptions of the situation. In experimental situations, the operators could be asked to input their state as the interactions progress.

The AMEBICA Industrial Exemplar

The usability studies have indicated that the AMEBICA demonstrators were largely successful. They flexibly managed the information and the procedures required during emergencies. It also successfully demonstrated maintenance interventions including alarm views, mimics, historical views, operator guides, emergency tracing and recovery, optimisation and on-line help. More specifically, the introduction of AMEBICA capabilities improved the overall interface and operators acceptance by the integration of new features such as:

Auto-adaptive management of different graphical elements and formats

Spatial reorganization of information and animation of graphical elements

Automatic tracking of alarms and process information

Integration and presentation of static and dynamic images, graphical and textual documents (e.g., description of network components, operational and procedural information, schemes and construction details, etc.) depending on a specific operational contexts

Integration of audio support (vocal input, alarm sounds)

Management of different levels of granularity dependent on the zoom and scale factors, with the capability of selecting areas of interest for enlargement and provision of detailed information

The advantages specifically gained in the ENM specific application were:

Enhancement of the graphical interface by more flexible information management and improved realism of interaction

More accurate and detailed descriptions of network information by integration and joint management of multimedia data (e.g., on-line documentation, live images, animation, clips.)

Improved configurability due to the systems generality. Therefore development times and cost should be significantly reduced.

Improved safety margins through the efficient use of alarms and warnings and utilisation of optimum media

The reduction of delays in the operator actions. As a consequence, decisions and actions are performed in a timely fashion.

The avoidance of lost revenue due to the adaptive system reacting promptly in order to avoid the possible failure of the system due, for example, to undetected degradation. Additionally, avoiding system failure means the supplier will potentially avoid penalties for having failed to satisfy mandatory standards of supply.

Increased security and safety of the process itself since many electricity blackouts have resulted from the avalanche effect derived from a relatively minor initial disturbance. These minor initiating problems are often not correctly identified and corrected at the beginning. The adaptive system is expected to help operators detect these types of problems.

General System Limitations

The previous section detailed some of the systems successes. Of course, as with any complex system, some limitations inevitably became apparent. This section outlines some of the general problems experienced.

More Sophisticated Reasoning

Although the system generally worked well, there was room for improvement. One area in which the system could be improved is in its reasoning ability. This is true particularly in the case of consistency checking. A more sophisticated reasoning mechanism would be able to strike a better balance between being wholly consistent and being wholly adaptive.

Greater Exploration Of Matrix Constraints

Although the matrix constraints work very well within the current system, a more comprehensive study of operator actions within a specific process system would give a more accurate and detailed appraisal of the adaptive system behaviour. The matrix as it stands, provides a good general adaptation framework. To operate well in other processes a better-tailored matrix would provide the system with adaptation operations better suited for that environment.

Fuller Understanding Of Operators Role

Deducing operator's behaviour and condition is a very difficult, uncertain problem. However, since the Process Control environment is a closed loop system, it makes the deduction of adaptation conditions a somewhat simpler problem. Since the operator must follow certain procedures under a certain set of process conditions, the system can watch for these conditions and follow the operator's actions. By comparing the actual actions to those that procedure requires, a deduction can be made of how far the operator is deviating from expected performance. Even utilising this method it is still difficult to correctly ascertain accurately the operators condition. Much more work is required in this area.

As the system stands there is only a superficial examination of the operator's condition, which suffices for the prototype. In the future, a more sophisticated, powerful mechanism is necessary to deduce operator state.

Implementation Issues

Implementation Successes

In the previous section an examination was made of general, system wide success and limitations. In this section specific issues relating to the actual prototype implementation, its successes and limitations are detailed

Real Time Operation

Real time operation of the system was imperative if the system was to be an acceptable and efficient adaptation system. It was critical that the system operate in a timely manner. This was achieved through several optimisation procedures imposed on the system.

The streaming concept ensured that data streams arrived at the interface in a timely manner, without adversely affecting the reasoning process. Part of the general performance problem was the nature of the Java language, which required optimisation to ensure timely operation. This was achieved by the use of the HotSpot Just-in-time compiler, and heavy code optimisation. In other programming languages support for multi-threading is limited (C++ for example) and makes programming of stable, reliable multi-agent system very difficult. Utilising Java, however, within AMEBICA, provided built-in support for threads and provided a way to obtain fast, lightweight concurrency within a single process space.

The initial use of the JACK agent toolkit was abandoned midway through the implementation as it slowed down inter-agent communication. Although it provided many conceptual benefits at design time and handled multi-threads within a single address space very well, it slowed operations down by an unacceptable factor. To counter this, the final implementation was built using pure Java, and was designed to act in a similar way to the JACK agent code.

Additionally, JACK utilised a BDI (Belief, Desires and Intentions), strong AI framework for implementation. In an optimised, closed loop system such as the one described here, the BDI model is not really appropriate. Therefore this system was abandoned for a simpler, more dedicated Java framework where the reasoning process could match constraints with a single plan, rather than a variety (under JACK).

System Performance

The system prototype operated within all the performance boundaries required of it. There was only a negligible delay in the time between the default representation being rendered at the interface, and the adaptive system operating to optimise that representation. This performance gain was achieved partly through the form and nature of the architectural framework. The architecture was designed for optimum performance operation, and this was achieved through highly efficient message exchanging between agents so that communication was directed specifically according to the position of agent within the system. For instance, bi-directional unicast communication was used between agents that only communicated with each other, and no other agent.

Additionally performance gains were achieved by carefully designing the multi-agent system for optimum performance, by striking a fine balance between having several smaller agents in separate threads dealing with conceptually separated processes, and by combining several concurrent tasks into monolithic agents. It is important achievement of AMEBICA that a balance between these competing constraints was reached.

Another key method of increasing system performance was achieved by optimising the form of reasoning the multi-agent system used. For example, very basic, but fast reasoning increases system performance, but severely limits the effectiveness of the resulting adaptations. Elaborate, complex reasoning achieves the best possible form of adaptation, but renders the results obsolete, as the time taken to perform this reasoning is prohibitively long. The Multi-agent architecture succeeded in doing this, by carefully stripping down interactions to a minimum, and rejecting reasoning by agent negotiation. Instead the key reasoning processes were implemented in two highly optimised agents. By reducing the overhead of a many-threaded system, impressive performance gains were achieved

Successful Implementation Of Generality

One of the main achievements of the architecture was to implement the system so that it offered a generic system, which allows deployment on a variety of different platforms and processes. By coding the system in Java the system could be implemented on most available platforms. The implementation also used CORBA extensions to Java to interface the Process Model with the Process (in the prototype implementation, the Process was represented by a process simulator). The use of CORBA also allowed the system to be integrated with most legacy software, through the use of wrappers.

Since the prototype could not be integrated with a real process graphical user interface, a custom interface was developed. However, the prototype still maintained the ability to allow interfacing with a real process interface. This ability was implemented through the use of an adaptor called the Abstract Rendering Interface (ARI).

A generic approach was implemented by the complete separation of internal general-purpose knowledge and external application specific knowledge. The internal knowledge is represented and used by the internal framework agents to manage the graphical user interface. The external knowledge is concentrated into some special elements (Process Model Agent, Media Selection Tables, Graphical Objects) and is entered during customisation. The framework knowledge does not contain reference to any particular application domain and is expressed in terms of a graphical object management ontology. The main elements are streams, evidence levels, representations, graphical objects and graphical object properties. Such elements are managed with the help of rules and constraints that describes their relationships. Human factor rules express general knowledge on best user interface organization modalities to optimise user understanding and to reduce stress and confusion.

The architecture allows the system to be integrated into existing applications without a complete re-design. This ability is exacted through the use of a "flexible mapping" approach. AMEBICA works "in parallel" with a conventional applications and manipulates user interface graphical objects in an asynchronous way. The user interface can continue to work as it would without the AMEBICA auto-adaptive functionality. The integration of AMEBICA and the application requires some modifications at the graphical interface to allow AMEBICA to operate on it through the Abstract Rendering Interface. It would also require the system designers to represent specific application knowledge with the AMEBICA customisation modalities.

Spatial Adaptation And Interface Configuration

Spatial adaptation of the interface worked very well. . Initially, some problems were encountered with the system placing renderings in the wrong position and of an incorrect size. After some investigation it was discovered that the problem was not with the Media Allocator Agent, but rather with the Presentation Agent. It was deriving spatial co-ordinate incorrectly and this problem rippled through to the Media Allocator. The problem was swiftly arrested, although it did highlight the tight inter-dependencies between these two agents. It was suggested that in future versions of the system, the Presentation Agent might be developed as part of the Media Allocator Agent. Although, this might improve performance marginally, it would violate the conceptual differences between the agents. By integrating the two together, problems might arise when attempting to update the Presentation Agent, and although small performance gains might be achieved, it is certainly possible that these would be lost by overburdening the Media Allocator agents computational thread system, and thus increasing reasoning time.

Intra-Agent Message Passing

The system successfully used a mixed initiative approach to agent communication. A single, global communication policy was rejected as being too cumbersome and inefficient. Instead several communication strategies were adopted depending on the position and role of the agents within the architecture. Most of the key communication routes were point-to-point (most efficient communication method, low overhead). At strategic points a broadcast approach was used (Media Allocator Agent to Media Agents).

A very successful use of data abstraction was used to provide maximum meta-information about system context using tried and tested Object Oriented approaches. Thus, Meta data was encapsulated and captured within Representation Data objects. This allowed successive agents to glean appropriate information about the state of other objects without resorting to extra communication steps (further improving performance).

Implementation Limitations

Since the implementation was a prototype, not all of the full functionality could be implemented in the time allowed. The following section describes some of the systems limitations.

Limited Full Use Of Complete Range Of Media

Although a wide range of different media were implemented, it was not possible to fully integrate all the media types required in a full process implementation. The area most affected was the auditory domain. The prototype concentrated mainly on graphical media, at the expense of the auditory medium. The system did implement speech recognition, and basic tones for alarm signals. However, earcons and other auditory forms were not included. This did not detract greatly from the successful operation of the system, it did however limit the systems flexibility.

Customisation Difficulty

The system requires customisation to allow it to be deployed upon different platforms and processes. To make this task easier it is preferable for the system to incorporate within it, a full set of customisation tools that allow designers to set up the system for their specific process with a minimum amount of effort.

At the present time these tools do exist but only in limited form. Although this did not affect the prototype, it might have a detrimental affect on designers using it on other systems.

Limited Implementation Of Operator Agent And Human Factors Database

The final prototype did not fully implement all the functions required of the Operator Agent and Human Factors database. The Operator Agent was not very sophisticated and succeeded only in deducing, in a basic fashion, the operator state. For the limited scenarios tested on the prototype this was not a problem, for deployment on a much more comprehensive, industrial system it would require a great deal more development.

The same is true for the Human Factors database. A limited set of rules was implemented that were customised to meet the scenarios the prototype on which they were tested. For full operation on an industrial scale, a much bigger set of rules would be required. Both of these issues arose due to time constraints, but with additional time and further work, both systems could be fully implemented with little difficulty.

Debugging Problems

As in all multi-agent systems debugging became an important issue. Many problems were encountered when attempting to successfully debug the system. To alleviate these problems several very useful debug tools were constructed which allowed developers to trace intra-agent messages, and allowed designers to observe the internal states of key variables. For system construction, these tools were found to be very useful.

Limited Consistency Checking

The Consistency Database was only implemented in a limited fashion. As in the case of the Human Factors Database and the Operator Agent, the consistency checking was limited to the two exemplars the prototype was to be tested on. This in no way invalidated its operation; it merely demonstrated the system working for a customised environment only.

Overall Conclusion

The work in this thesis attempts to deal with the problems of bandwidth-limited interfaces and information overload by introducing an element of adaptation into the interface to render the most salient information at the most appropriate times. By doing this, an element of flexibility was introduced at the interface, which improves operator efficiency, and the way in which operators deal with process disturbances. A multi-agent approach was adopted to allow the actors responsible for making decisions about suitable adaptations, to each be represented by an autonomous agent.

This thesis has presented and discussed the rationale behind, and the conceptual design of, an intelligent, flexible, agent architecture. It has demonstrated how adaptation can be bounded, and how system rigidity can be avoided via the use of a flexible representation mapping system. Additionally, the architecture has been designed to ensure that the core agent framework is reasonably process-independent, and therefore applicable to many application areas. This process independence was achieved by the use of external interfaces to both the process and the system-rendering engine, which translate process dependent terms to proprietary terms and vice versa.

The system was constructed as a prototype for use in two different industrial scenarios, the Thermal Power Plant and Electricity Network Management systems. In both cases the prototype was tested on actual operators, and was largely deemed to be successful. To operate within these domains it was critical that the system operate in a timely manner. This was achieved through successful code-optimisation techniques, and by the use of tools to increase the speed of the Java language. Additionally, concepts such as the Streaming Concept allowed important data to be displayed as quickly as possible. Above all, the AMEBICA system has shown that multi-agent approach, when carefully constructed, *can* achieve the performance levels required.

Finally the thesis has investigated the issues surrounding adaptation and has suggested (through the Adaptation matrix) a possible design methodology for producing adaptive systems.

Future Work

There are still many issues to be resolved, and this thesis has addresses merely some of the more salient points. Many problems were encountered in the construction and prototyping of the architecture that suggested avenues for further investigation. Such points include:

Increasing run-time speed

Improving the systems recognition of operator states.

Improving thread resource usage to ensure efficient inter-agent communication and prevent deadlock.

Incorporating greater media input.

Testing on different process domains.

Improved consistency checking

Full implementation of the Operator Agent and Human Factors Database

Full implementation of customisation tools, to improve the systems ability to be deployed on different processes

Test deployment on a real process interface attached to a real process.

Full implementation of a variety of media including a full selection of auditory modes.

Each of these topics would help streamline the prototype and provide invaluable date for future adaptive interface designers of real-time process control systems The system as it stands, in prototype form, is merely a platform for the testing of new concepts. To be deployed on a full system, a great deal more work would be required on stabilising the system, and ensuring system performance is totally optimised

BIBLIOGRAPHY

Agre PE & D. Chapman(1987) *'Pengi: An Implementation of a Theory of Activity',* In: Proceedings of the 6th National Conference of Artificial Intelligence, Morgan Kaufmann, 268-272

AHCI (1998), *Aviation Human-Computer Interface (AHCI) Style Guide*, Technical Report; Report Number 64201-97U/61223, May 1998

Alexandros M *(*1995*) 'Amalthaea: Information Discovery and Filtering using a MultiAgent Evolving Ecosystem',*., MIT Media Lab Report,01/04/95

Alty J. L.&, Bergan M. (1992), '*Guidelines For Multimedia Interface Design in a Process Control Application*', Int. Conf on Design and Safety of Nuclear Reactors Anp'92, Vol 4, (Oka, Y, and Koshizuka Eds.), Atomic Energy Society of Japan, pp 43 Iv-1 to 43 Iv-7

Alty J. L., Bergan M., Craufurd P., Dolphin C. (1992), '*Experiments using Multimedia Interfaces in Process Control: Some Initial Results',* Hyperstructure Concepts and Cooperative Work, Proc. 2nd Eurographics Workshop on Multimedia, Darmstadt, (Ch. Hornung Ed.), EG Tech. Rep. EG92MU, pp 27-50

Alty, J.L. & McKell, P. (1986) '*Application Modelling in a User Interface Management Syste*m,' in: M.D. Harrison and A.F. Monk (Eds.), People and Computers: Designing for Usability (Cambridge: Cambridge University Press).

Alty, J.L. & Mullin, J. (1987) '*The role of the dialogue system in a User Interface Management System*', in: B. Shackel and H-J. Bullinger (Eds.), Proc. INTERACT '87, Second IFIP Conference on Human-Computer Interaction (Amsterdam: Elsevier Science Publishers B.V.).

Andre, A. & Wickens, C. (1992). '*Compatibility and Consistency in Display-Control Systems: Implications for Aircraft Decision Aid Design'*. Human Factors*, 34* (6).

Arens Y., and Hovy (1995), E. '*The Design of a Model-Based Multimedia Interaction Manager*'". Artificial Intelligence Review, Vol. 9,Num 2-3,
Austin J.L. (ed.) (1969), *'How to do Things With Words'*, Oxford University Press, 1962.

Bainbridge, L. (1987), '*Ironies of Automation*', New Technology and Human Error, pp 271 - 283, John Wiley

Baker (1996) Baker, A.D, *'Chapter 8 - Metaphor or Reality: A case study where agents bid with actual costs to schedule a factory,'* Market-based Control, Scott H. Clearwater (ed.), World Scientific, pp. 184-223, 1996.

Barnard, P. (1987) '*Cognitive Resources and the Learning of Human-Computer Dialogues'*. In Carroll, J. (ed.) Interfacing thought: Cognitive aspects of human-Computer Interaction. MIT Press, Cambridge, Mass

Beltracchi L. (1988), '*Alarm Coding of a Model-Based Display*', Human Factors and Power Plants: Proceedings of the IEEE Fourth Conference Monterey, California, 5-9 June, pp 157 - 164.

Benyon D.R., Innocent P.R., Murray, D.M. (1987) '*System adaptivity and the modelling of stereotypes'*. In: B. Shackel and H-J. Bullinger (Eds.), Proc. INTERACT '87, Second IFIP Conference on Human-Computer Interaction (Amsterdam: Elsevier Science Publishers B.V.).

Benyon D.R., Milan, S. and Murray, D.M. (1987) '*Modelling users' cognitive abilities in an adaptive system'*, In: J. Rasmussen and P. Zunde (Eds.), Proc. 5th Symposium EFISS, Risø National Laboratory, Denmark, November 1987 (New York: Plenum Publishing).

Benyon, D.R., Jennings, F. and Murray, D.M. (1990) '*An adaptive system developer's toolkit*'. In: Diaper, D. Cockton, G., Gilmore, D. and Shackel, B. (Eds.), Proc. INTERACT '90, Third IFIP Conference on Human-Computer Interaction (Amsterdam: Elsevier Science Publishers B.V.).

Benyon, D.R. and Murray, D.M. (1988) '*Experience with adaptive interfaces*',The Computer Journal*,* Vol. 31(5).

Benyon, D. R. & Murray (1993a), D. M. '*Adaptive Systems; from Intelligent Tutoring to Autonomous Agents.'*. Knowledge-based Systems, 6 (3)

Benyon, D. R. & Murray, D. M. (1993b) '*Applying user modelling to human-computer interaction design',* AI Review (6)

Benyon, D.R. (1984) '*Monitor: a self-adaptive user interface'*. In: B. Shackel (Ed.), Proc. INTERACT '84, First IFIP Conference on Human Computer Interaction (Amsterdam: Elsevier Science Publishers B.V.).

Benyon, D.R. (1988) '*User models: what's the purpos*e', In: M. Cooper and D.Dodson (Eds.) Alvey Knowledge-Based Systems Club, Intelligent Interfaces Special Interest Group, Proc. of the Second Intelligent Interfaces Meeting, May 28-29th 1987, London (London: Department of Trade and Industry Information Engineering Directorate).

Benyon, D.R. (1990) '*Information and Data Modelling*'. Blackwell Scientific Publishers, Oxford

Benyon, D.R. (1992a) '*The Role of Task Analysis in Systems Design*', in: Interacting with Computers*,* vol. 4 no. 1, 102 – 121

Benyon, D.R. (1992b) '*Task Analysis and Systems Design: The Discipline of Data.*' In Interacting with Computers, 4(2) 246 - 259

Benyon, D.R. (1993a) '*Adaptive Systems; A Solution to Usability Problems*', in: User Modelling and User Adapted Interaction Kluwer. (1993) 1-22

Benyon, D. R. (1993b) '*Accommodating Individual Differences through an Adaptive User Interface*'. In Schneider-Hufschmidt, M., Kuhme, T. and Malinowski, U. (eds.) *Adaptive User Interfaces - Results and Prospects,* Elsvier Science Publications, North-Holland, Amsterdam. 1993

Benyon, D. R (1997): '*Intelligent Interface Technology to Improve Human-Computer Interaction (Tutorial)',* In: HCI International '97. San Francisco, USA. (1997).

Bieger & Glock (1984): Bieger, G R & Glock M D *'The Information Content of Picture-Text Instructions'* Journal of Experimental education 53 68-76

Bock H. W. (1988), '*Future Main Control Room Design For Siemens Nuclear Power Plant*s', Man-Machine Interface in the Nuclear Industry: Control and Instrumentation, Robotics and Artificial Intelligence: Proceedings of the International Conference, Tokyo, 15-19 February, pp 613 - 623, International Atomic Energy Authority.

Boohrer (1975): Boohrer, H R "'*Relative Compehensibility of Pictorial Information and Printed word in Proceduralised Instructions',* Human Factors 17, 3, 266-277
Boone, G. (1998). '*Concept features in Re: Agent, an intelligent email agent*.' Proceedings of the Second International Conference on Autonomous Agents (pp. 141{148). Minneapolis: ACM Press.

Bosser, T. (1987) '*Learning in Man-Computer Interaction*', Springer-Verlag

Branjik, G., Guida, G. and Tasso, C. (1987) *User modelling in intelligent information retrieval*, Information Processing and Management, Vol. 23(4).

Branjik, G., Guida, G. and Tasso, C. (1990) '*User Modelling in Expert Man-Machine Interfaces: A case study in Information Retrieval*', IEEE Trans. Systems, Man and Cybernetics*,* Vol. 20(1)

Bransby M. and J. Jenkinson (1998), IEE Computing and Control Engineering Journal, Vol. 9, Number 2, p61-67

Bronisz, D., Grossi, T. and Jean-Marie, F. (1989) '*Advice-giving dialogue: an integrated system*'. In: Proc. 6th Annual ESPRIT Conference, Brussels, November Kluwer Academic Publishers.

Brooks RA. (1991) *'Elephants Don't Play Chess',* P. Maes(ed.), *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*. The MIT press, 3-15

Browne, D.P, Trevellyan, R., Totterdell, P.A and Norman, M (1987) '*Metircs for the building, evaluation and comprehension of self-regulating adaptive systems*'. In Bullinger, H.J. and Shackel, B (eds), INTERACT '87. Elsevier – North Holland, Amsterdam

Browne, D.P., Totterdell, P.A. and Norman, M.A. (1990) '*Adaptive User Interfaces'* (London: Academic Press).

Browne, P. D (1993). '*Experiences form the AID project'*. In: Schneider Hufschmidt, M., Kuhme, T., Malinowski, U. (eds.): Adaptive User Interfaces:Principles and Practice. Adaptive User Interfaces. Elsevier Science Publishers B.V Amsterdam, North-Holland. (1993) 69-78

Brustoloni Jose C (1991) *'Autonomous Agents: Characterisation and Requirements',* , Carnegie Mellon Technical Report CMU-CS-91-204, Pittsburgh: Carnegie Mellon University

Bundy, A. (1983) (ed.) *Alvey 1983 Intelligent Front End Workshop* 26-27 Sept. 1983 Cosener's House, Abingdon, England. DTI, London

Burch (1973) Buch, N. '*Theory of Film Practice'*. Saeker & Warburg: London, 1973

Burton, R. R. (1982) '*Diagnosing Bugs in a Simple Procedural Skill*' in D. H. Sleeman and J. S. Brown (eds.) Intelligent Tutoring Systems. (Academic Press, New York)

Bury (1985): Bury K.F, S.E.Davus and M.J.Darnell; *'Window Management; A review of issue and some results for user testing*', Technical Report HFC-53, IBM Human Factors Centre, San Jose, CA, June 1985

Carberry, S. (1989) '*Plan recognition and its use in understanding dialog*'. In: W. Wahlster and A. Kobsa (Eds.) op. cit. Card, S., Moran, A.P., and Newell, A. (1983) The Psychology of Human-Computer Interaction (Hillsdale, NJ: Lawrence Erlbaum Associates).

Carroll, J.M. & McKendree, J. (1987) '*Interface design issues for advice-giving systems'*, Communications of the ACM, Vol. 30(1).

Carroll, J.M. (1991) '*The Nuremberg Funnel.*' MIT Press, Cambridge, Mass

Chaib-draa, B., & Moulin, (1987) '*Architecture for Distributed Artificial Intelligent systems*', IEEE Proceedings, Montreal, pp. 64-69, 1987.

Chaib-draa, B., Moulin, B., Mandiau, R. & Millot, P. (1996), '*Chapter 1 - Trends in Distributed Artificial Intelligence*', G. M. P. O'Hare and N. R. Jennings (eds.), John Wiley & SonsInc.,pp. 3-55, 1996.

Chappel H., Wilson M. (*1993)*, *'Knowledge-Based Design of Graphical Responses'*, International Workshop on Intelligent User Interfaces, pp 29-36.

Chignell, M.H. and Hancock, P.A. (1988) '*Intelligent Interfaces*' in M. Helander (Ed.) Handbook of Human-Computer Interaction (Amsterdam: Elsevier Science Publishers B.V.).

Chin, D.N. (1986) '*User modelling in UC, the Unix consultan*t.' In: M. Mantei and P. Orbiton (Eds.), Proc. CHI '86, Human Factors in Computing Systems (New York: ACM).

Chin, D.N. (1989) *KNOME: 'Modelling what the user knows in UC*'. In: W. Wahlster and A. Kobsa (Eds.) op.cit. Computational Linguistics, (1988) Vol. 14(3).

Cohen, P. R. & Levesque, H. J (1988), '*Rational Interaction as the Basis for Communication*', SRI Technical note, 1988.

Corkill, D.D., & Lesser, V.R (1983), '*The use of meta-level control for coordination in a distribute problem solving network*', Proceedings of the 8th IJCAI, Karlsruhe, Germany, pp. 748-756, 1983.

Corsberg D. (1988), '*Effectively Processing and Displaying Alarm Information*', Human Factors and Power Plants: Proceedings of the IEEE Fourth Conference, Monterey, California 5-9 June, 1988, pp 95 - 100.

Cote-Munoz, A. H. (1993) '*AIDA: An Adaptive System for Interactive Drafting and CAD Applications*'. In: Schneider-Hufschmidt, M.,Kuhme, T., Malinowski, U.(eds.): Adaptive User Interfaces: Principles and Practice. Adaptive User Interfaces. Elsevier Science Publishers B.V., Amsterdam, North-Holland. (1993) 225-240

Coutaz, J. (1987) '*PAC: An object orientated model for implementing user Interface*'. In H.J. Bullinger and B. Shackel (Eds.) Human- Computer Interaction, Proceedings of INTERACT '87 (Amsterdam: Elsevier Science Publishers B.V.).

Crossman M., & Cooke, J.E. (1974), *'Manual Control of Slow-Response Systems',* The Human Operator In Process Control (Edwards E. & Lees F. Eds.), London, Taylor and Francis 1974

Davis, R. & Smith, R. G. (1983), '*Negotiation as a Metaphor for Distributed Problem Solving*,' Artificial Intelligence 20, pp. 63-109, 1983.

Dede, C. (1986) '*A review and synthesis of recent research in intelligent computer-assisted instruction*', International Journal of Man-Machine Studies , Vol. 24(?).

Degani, A., Shafto, M. & Kirlik, A. (1996). '*Mode Usage in Automated Cockpits: Some Initial Observations*', NASA report

Dennett, D. (1989) *'The Intentional Stance',* MIT press, Cambridge, Ma.

Diaper, D. (1989) *'Task Analysis for Human-Computer Interaction',*.Chichester: Ellis Horwood

Dicken C.R. (1999); *'Soft' Control Desks and Alarm Displays'*, IEE Computing and Control Engineering Journal, Vol 10, Number 1 , p11-16

Dieterich, H., Malinowski, U., Kuhme, T., Schneider Hufschmidt, M (1993): '*State of the Art in Adaptive User Interfaces'*. In: Schneider-Hufschmidt, M.,Kuhme, T., Malinowski, U. (eds.): Adaptive User Interfaces: Principles and Practice. Adaptive User Interfaces. Elsevier Science Publishers B.V, Amsterdam, North-Holland. (1993) 13-48

Edmonds, E.A. (1981) '*Adaptive Man-Computer Dialogues'*. In Coombs, M. J.and Alty, J. L. Computing Skills and the user Interface. Academic Press, London

Edmonds, E.A. (1987) '*Adaptation, Response and Knowledg*e', Knowledge-Based Systems , Vol. 1(1), Editorial.

Egan, D.E. (1988) '*Individual differences in Human-Computer Interaction'*. In Helander, M. (Ed.) Handbook of Human-Computer Interaction (Amsterdam: Elsevier Science Publishers B.V.).

Elkerton, J. (1987) '*A framework for designing intelligent human-computer Interfaces'*, in: G. Salvendy (Ed.) Cognitive Engineering in the Design of Human-Computer Interaction and Expert-Systems (Amsterdam: Elsevier Science Publishers B.V.)

Endsley, M. R. (1995). *'Toward a Theory of Situation Awareness in Dynamic Systems'*, Human Factors, 37(1), pp. 32-64.

Faraday P (1998), *'Theory based Design & Evaluation of Multimedia Presentation Interfaces*'", PhD. Thesis, 1998

Finin, T. & Wiederhold, G. (1993), *'An Overview of KQML: A Knowledge Query and Manipulation Language'*, Department of Computer Science, Stanford University,1993.

Finin, T.W. (1989) '*GUMS - A general user modelling shell*'. In: W. Wahlster and A. Kobsa (Eds.) op.cit.

FIPA1: *FIPA Rationale*, http://www.cselt.stet.it/fipa/fipa_rationale.htm, 1996

FIPA97spec http://www.cselt.stet.it/fipa/spec/fipa97reston.htm, FIPA '97 specification - Reston draft (ver 1.0), 19 April 1997.

Fischer, G. (1987) '*Making computers more useful*'. In: G. Salvendy (Ed.) Cognitive Engineering in the Design of Human-Computer Interaction and Expert-Systems (Amsterdam: Elsevier Science Publishers B.V.).

Fischer, G. (1989) '*HCI Software: Lessons learned, challenges ahead*', IEEE Software , January 1989.

Fischer, G., Lemke, A.C. and Schwab, T. (1986) '*Knowledge-based help system*s'. In: M. Mantei and P. Orbiton (Eds.), Proc. CHI '86, Human Factors in Computing Systems (New York: ACM).

Fischer, G., Morch, A. and McCall, R. (1989) '*Design environments for constructive and argumentative design*'. Proc. CHI '89, Human Factors in Computing Systems (New York: ACM).

Flach (1995) Flach, J.M. & Dominguez, C.O. '*Use-centered design*'. Ergonomics in Design, July, 19 - 24.

Foner L (1993) *'What's an Agent Anyway? A Sociological Case Study',* Agent Memo 93-01, Agent Group, MIT Media Lab 1993.

Fowler, C. & Murray, D.M. (1987) '*Gender and cognitive style differences at the human-computer interfac*e.' In: B. Shackel and H-J. Bullinger (Eds.), Proc. INTERACT '87, Second IFIP Conference on Human-Computer Interaction (Amsterdam: Elsevier Science Publishers B.V.).

Fox ,M. S. (1993) '*A Common-Sense Model of the Enterprise'*, Proceedings of Industrial Engineering Research Conference, pp. 178-194, 1993.

Franklin & Graesser (1996); '*Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents'*, Stan, Proceedings for The Third International Workshop on Agent Theories, Architectures and Languages, Springer-Verlag, 1996

Frivold, R., Lang. R., & Fong, M. (1994), '*Extending WWW for Synchronous Collaboration,'"* Electronic Proceedings of Second WWW Conference, Phoenix, pp. 559-583, 1994.

Furnas, G. (1985) '*New Jersey experience with an adaptive indexing schem*e'. In: A. Janda (Ed.), Proc. CHI '85, Human Factors in Computing Systems (New York: ACM).

Gasser, L. & Bond (1988), A. H. (eds.), '*Readings in Distributed Artificial Intelligence*', San Mateo, CA: Morgan Kaufmann,

Gasser, L. (1991), '*Social Conceptions of Knowledge and Action: DAI Foundations and Open Systems*', Artificial Intelligence 47, pp. 107-138, 1991.

Genesereth, M. R. & Ketchpel, S. P. (1994), '*Software Agents*', Communications of the ACM 37 (7), pp. 48-53, 1994.

Gervasio, M., Iba, W., & Langley, P. (1998). '*Learning to predict user operations for adaptive scheduling*'. Proceedings of the Fifteenth National Conference on Artificial Intelligence (pp. 721-726). Madison, WI: AAAI Press.

Gilbert& Janca *'IBM Intelligent Agents'*-

Goodwin R. (1993) '*Formalizing Properties of Agents';* CMU Internal Report CMU-CS-93-159

Gray, W., Hefley, W. and Murray, D. M. (eds.) (1993) '*Proceedings of 1st international Workshop on Intelligent User Interfaces',* ACM Publications, New York.

Green, J.M. Hoc and D.M. Murray (Eds.) '*Working with Computers: Theory versus Outcome'*, (London: Academic Press).

Green, M. (1985) '*Report on Dialogue Specification Tools in UIMS'*, in Pfaff, G.E. (Ed.),User interface Management Systems, Springer Verlag, Heidelberg.

Green, T.R.G., Schiele, F. and Payne S. J. (1988) '*Formalisable models of user knowledge in human-computer interaction'*, In: C.C. van der Veer, T.R.G.

Greenberg, S. & Witten, I.H. (1985) '*Adaptive personalised interfaces –a question of viability'*, Behaviour and Information Technology*,* Vol. 4(1).

Hancock, P.A. & Chignell, M.H. (1988) '*Mental workload dynamics in adaptive interface design'*, *IEEE Trans. SMC,* Vol. 18(4).

Hancock, P.A. and Chignell, M.H. (1989) (eds.) '*Intelligent Interfaces; Theory*, *Research and Design*'. North-Holland, New York

Hansen, S.S., Holgaard, L. and Smith, M. (1988) '*EUROHELP: intelligent help systems for information processing system*s.' In: Proc. 5th Annual ESPRIT Conference, Brussels, November 1988 (Kluwer Academic Publishers)

Hartson, H.R. and Hix, D. (1989) '*Toward empirically derived methodologies and tools for HCI development*' in International Journal of Man Machine Studies 31, 477-494

Hayes-Roth *(1995) 'An Architecture for Adaptive Intelligent Systems'*, Artificial Intelligence: Special Issue on Agents and Interactivity(1995), 72, 329-365

Hermens, L. & Schlimmer, J.,(1994) *'A Machine Learning Apprentice for the Completion of Repetitive Forms',* Proceedings of the 9th IEEE Conference on Artificial Intelligence Applications, Orlando, Florida: IEEE Press, pp. 164-170, 1994.

Hewitt, C. (1986), '*Viewing Control Structures as Patterns of Passing Messages',* Artificial Intelligence 8(3), pp.323-364,1986.

Hollnagel, Erik (1993). '*The Phenotype of Erroneous Actions.'* International Journal of Man-Machine Studies, 39:1-32,1993.

Iba, W., Gervasio, M., Langley, P., & Sage, S. (1998). '*Experimental studies of intelligent assistance for crisis response'*. Proceedings of the Twentieth Annual Conference of the Cognitive Science Society. Madison, WI: Lawrence Erlbaum. *IEEE Software* , (1989)

Innocent, P.R. (1982) '*A self-adaptive user interface'*, International Journal of Man Machine Studies , Vol. 16(3) 287 – 300

Jennings F., Benyon D.R. and Murray D.M. (1991) '*Adapting Systems to individual differences in cognitive style'*, Acta Psychologica,

Jennings, F. & Benyon, D.R. (1992) '*Database Systems: Different Interfaces for different users'*

Jennings, N. R (1995), '*Chapter 6 - Coordination Techniques for Distributed AI'*, Foundations of Distributed Artificial Intelligence, G. M. P. O'Hare and N. R. Jennings (eds.), John Wiley & Sons, pp. 187-210, 1995.

Jennings, Alty et al (1996) *'ADEPT: Managing Business Processes using Intelligent Agents*'**,** Proc. BCS Expert Systems 96 Conference (ISIP Track),Cambridge UK 1996

Jennings, N. R. & Wooldridge, M. J. (1998). '*Applications of Intelligent Agents. In Agent Technologies: Foundations, Applications, and Markets*'. N. R. Jennings and M. J. Wooldridge (eds.)

Jerrams-Smith, J. (1985) '*SUSI -A smart user-system interface*', In: P. Johnson and S. Cook (Eds.), People and Computers: Designing the Interface (Cambridge: Cambridge University Press).

Johnson, P. (1989) '*Supporting System Design by analysing current task knowledge',* in Diaper, D. (ed.) Task Analysis for Human-Computer Interaction. Ellis-Horwood

Jorg & Hormann (1978): Jorg, S. & Hormann, H.: '*The influence of general and specific labels on the recognition of labeled and unlabelled parts of pictures'.* Journal of Verbal Leaning & Verbal Behaviour, 17, 445-454.

Kass, R. and Finin, T. (1988) '*The need for user models in generating expert system explanations*', International Journal of Expert Systems , Vol. 1(4).

Kass R. (1989) '*Student modelling in intelligent tutoring systems*'. In: W. Wahlster and A. Kobsa (Eds.) op. cit.

Kay (1990) *'User Interface: A Personal View'*, In B. Laurel, ed., The Art of Human-Computer Interface Design, Addison-Wesley, Reading, Mass., 1990

Kay, J. (1991) '*UM: a toolkit for user modelling',* In Schneider-Hufschmidt, M., Kuhme, T. and Malinowski, U. (eds.) *Adaptive User Interfaces - Results and Prospects,* Elsvier Science Publications, North-Holland, Amsterdam.

Khalil C.I.J. (1999a) *The Architecture of the AMEBICA Agent Based Adaptive Process Control Interface* ; 18th European Annual Conference on Human Decision Making and Manual Control Loughborough University, UK October 25th - 27th 1999

Khalil C.I.J (1999b); '*AMEBICA - An Auto Adaptive Multimedia Environment Based on Intelligent Collaborating Agents'* The Human Error and System Design and Management International Workshop, March 24-26 1999, Clausthal Technical University, Germany

Kieras, D. and Polson, P.G. (1985) '*An approach to the formal analysis of user Complexity*', International Journal of Man Machine Studies , Vol. 22 (?).

Kobsa, A. (1987) '*A taxonomy of beliefs and goals for user modelling in dialog System*s', Memo Nr. 28, Universitat des Saarlandes, Saarbrucken.

Kobsa, A. (1988) '*A bibliography of the field of user modelling in artificial intelligence dialog systems*', Memo Nr. 23, Universitat des Saarlandes, Saarbrucken.

Kobsa, A. & Wahlster, W. (1989) '*User models in dialog systems',* Berlin: Springer-Verlag

Lai K & T. Malone & K. Yu, (1988) *'Object Lens; A Spreadsheet for Co-operative Work',* , ACM Trans. Office. Inf. Syst. 6,4 332-353

Lang, K. (1995). '*NewsWeeder: Learning to Filter news*.' Proceedings of the Twelfth International Conference on Machine Learning (pp. 331-339). Lake Tahoe, CA: Morgan Kaufmann.

Langley, P., & Simon, H. A. (1995). '*Applications of machine learning and rule induction'*,. Communications of the ACM, 38, November, 55-64.

Langley (1997), '*Machine learning for adaptive user interfaces',* in German Artificial Intelligence, p53-62, Germany: Springer

Laurel, B. (1990) '*Interface Agent*s.', In: B. Laurel (Ed.) *The Art of Human-Computer Interface Desig*n, Addison Wesley, Wokingham

Lehner, P.E. (1987) *Cognitive factors in user/expert-system interactio*n, Human Factors, Vol. 29(1).

Lesser V & L. D. Erman (1980), *'Distributed interpretation: A model and an experiment', '*Special Issue on Distributed Processing, IEEE Trans. Comput. vol. C-29, pp. 1144)1163, December 1980.

Lesser, V. & Corkill, D (1987), '*Functionally Accurate, Cooperative Distributed Systems'*, IEEE Transactions on Systems, Man, and Cybernetics C-11(1), pp. 81- 96. 1987.

Lesser V.R (1998), '*Reflections on the Nature of Multi-Agent Coordination and Its Implications for an Agent Architecture'*. Autonomous Agents and Multi-Agent Systems, 1, 89)111 1998 Kluwer Academic Publishers.

Levitt et al (1994): Levitt, R., Cohen, P., Kunz, J., Nass, C., Christiansen, T. & Jin, Y., *'The Virtual Design Team: Simulating how Organisational Structure and Communication Tools affect Team Performance',* Computational Organisation Theory, Carley, K.& Prietula, M. (eds.,) San Francisco: Lawrence Erlbaum, pp. 67-91, 1994

Lind M., Osman A., Agger S., Jensen H. (1989), *Human-Machine Interface For Diagnosis Based on Multilevel Flow Modelling*, Cognitive Science Approaches to Process Control: Proceedings of the Second European Meeting, Siena, Italy, October 24

Lind, M. (1999). *'Making Sense of the Abstraction Hierarchy'*, Human-Machine Reliability and Co-operation, 20-24 September 1999, Villeneuve d'Ascq, Lille, France.

Maes, (1991) *'Intelligence Without Representation'*, Artificial Intelligence 47, 139-159

Maes, P. (1991) (ed), *'Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back'*, London, The MIT press, 1991.

Maes, P. & Kozierok, R. (1993) '*Learning Interface Agents',* AAAI '93, Conference on Artificial intelligence, Washington

Maes P, (1994) *'Agents That Reduce Work and Information Overload'*, Communications of the ACM 37(7), 31-40

Maes & Schneideman (1997); Maes, P. and Schneiderman, B. (1997). '*Direction manipulation vs. interface agents: a debate*'. Interactions, Vol. IV Number 6, ACM Press.

Mason, M.V. & Thomas, R.C. (1984) '*Experimental adaptive interfaces*', Information Technology: Research & Development , Vol. 3.

Mason, M.V. (1986) '*Adaptive command prompting in an on-line documentation system*', International Journal of Man-Machine Studies , Vol. 25

Mason, J. & Edwards, J.L. (1988) '*Surveying projects on intelligent dialogu*e', International Journal of Man-Machine Studies , Vol. 28

Mayer & Andersen (1991): Mayer, R. Anderson: '*Animation Needs Narration: An experimental test of a dual coding hypothesis':* Journal of Educational Psychology, 83(4), 484-490.

McCoy, K.F. (1989) '*Highlighting a user model to respond to misconception*s.' In: W. Wahlster and A. Kobsa (Eds.) op. cit.

Minsky, M. (ed.) (1985), *The Society of Mind*, New York: Simon & Schuster, 1985.

Mitchell et al (1994) Mitchell, T., Caruana, R., Freitag, D., McDermott, J. & Zabowski, D. (1994),*'Experience with a Learning Personal Assistat"* Communications of the ACM 37 (7), pp. 81-91, 1994.

Moore, J. & Swartout, W.R. (1988) '*Planning and reacting*', Proc, AAAI Workshop on text planning and generation, August 25, 1988, St. Paul, Minnesota.

Moran, T.P. (1981) '*Command language grammar: a representation for the user interface of interactive computer systems*', International Journal of Man- Machine Studies , Vol. 15(3).

Moran, T.P. (1983) '*Getting into a system: external-internal task mapping analysis',* In: R.N. Smith and R.W. Pew (Eds.) Proceedings CHI'83: Human Factors in Computing Systems (ACM Press).

Morik, K. (1989) '*User models and conversational settings: modelling the user's wants*'. In: W. Wahlster and A. Kobsa (Eds.) op. cit.

Murray, D.M. (1987a) '*A survey of user modelling definitions and techniques*', NPL DITC Report 92/87.

Murray, D.M. (1987b) '*Embedded user models*'. In: B. Shackel and H-J. Bullinger (Eds.), Proc. INTERACT '87, Second IFIP Conference on Human-Computer Interaction, Amsterdam: Elsevier Science Publishers B.V.

Murray, D.M. (1988) '*Building a user modelling shell*'. in: P. Zunde (Ed.), Proc.6th Symposium EFISS, Georgia Tech., Atlanta, Georgia, USA, October 1988, New York: Plenum Publishing

Murray, D.M. and Benyon, D. R. (1988) '*Models and designers tools for adaptive systems*', presented at 4th European Conference on Cognitive Ergonomics (ECCE-4), Cambridge, U.K., September 1988.

Murray, D.M. (1989) '*Modelling for adaptivity*', Proceedings of 8th Interdisciplinary Workshop, Informatics and Psychology, Scharding, Austria, May 1989 Amsterdam: North Holland.

Negroponte, N. (1989) '*Beyond the desktop metaphor*', International Journal of HCI , Vol. 1(1).

Newell, A. (1982) '*The Knowledge Level Artificial Intelligence*' 18(1) 87 127

Nielsen, J. (1986) '*A virtual protocol model for computer-human interaction',* in International Journal of Man-Machine Studies , Vol. 24.

Nielsen, J. (1992), *Usability engineering*, Academic Press, 1992.

Noah, W, & Halpin, S.M. (1986) '*Adaptive user interfaces for planning and decision aids in CHI systems',* IEEE Trans. SMC , Vol. 16(6).

Norcio, A.F. and Stanley, J. (1989) '*Adaptive human-computer interfaces: a literature study and perspective*', IEEE Trans. Systems, Man and Cybernetics ,Vol. 19(2).

Norman, D. (1986) in Norman, D. and Draper, S. (eds.) '*User Centred System Design.'*

Norman, D. A. (1990). '*The Design of Everyday Things*', New York: Doubleday.

Nwana, H (1996) *'Software Agents an Overview',* Knowledge Engineering Review, Vol 11:3, 1996 p209

Oppermann (1992) R. Oppermann, B. Murchner, H. Reiterer, M. Koch: *'Ergonomic Evaluation.The Guide EVADIS II'* (in German), 2nd edition, de Gruyter, Berlin, 1992.

Paris, C.L. (1989) '*The use of explicit user models in a generation system*'. In: W. Wahlster and A. Kobsa (Eds.) op. cit.

Parunak & H.Van Dyke (1996), *'Case Grammar: A Linguistic Tool for Engineering Agent-based Systems',* 1996.

Parunak, H.Van Dyke (1996), '*Go to the Ant: Engineering principles from Natural MultiAgent Systems'*, Annals of Operations Research (submitted), 1996.

Paraunak (1999) *Practical and Industrial Applications of Agent-Based Systems*, PAAM 1999 London

Payne, S.J. (1988) '*Complex problem spaces: modelling the knowledge needed to use interactive devices*,. In: B. Shackel and H-J. Bullinger (Eds.), Proc.INTERACT '87, Second IFIP Conference on Human-Computer Interaction (Amsterdam: Elsevier Science Publishers B.V.).

Payne, S.K. & Green, T.R.G. (1989) '*Task Action grammar: The Model and Its developments*'. In Diaper, D. (ed.) Task Analysis for Human-Computer Interaction. Ellis-Horwood

Pazzani, M., & Billsus, D. (1997). 'Learning and revising user profiles: The identification of interesting Web sites'. Machine Learning, 27, 313-331.

Pedersen R. (1999): '*A Systematic Approach to Design of Process Displays',* PhD. Thesis., Automation Department, Technical University of Denmark.

Penner, R.(1998) '*Automating User Interface Design.*' In Proceedings of the International Systems, Man, and Cybernetics Conference. San Diego, California.

Pollack, M.(1985) '*Information Sought And Information Provided. An Empirical Study of User/Expert Dialogues'*. In Proceeding of CH1'85. Human Factors in Computing Systems, ACM: New York

Pylyshyn, Z. W. (1984) *Computation and Cognition* MIT press, Cambridge,Ma.

Rasmussen, J. (1986) '*Information processing and human-machine interaction*', Amsterdam: Elsevier North-Holland

Rasmussen, J. & Vicente, K. J. (1987), '*Cognitive Control of Human Activities and Errors: Implications For Ecological Interface Design*', Research Report (Riso-M-2660), Riso National Laboratory.

Rasmussen, J. (1987) ‘*Mental models and their implications for design*’, In:Austrian Computer Society 6th Workshop on Informatics and Psychology, June 1987.

Rasmussen, J., Petjersen, A.M and Goodstein, L.P (1994). *Cognitive Systems Engineering*. John Wiley.

Reason J. (1998), ‘*Errors and Violations: the Lessons of Chernobyl*’, Human Factors and Power Plants: Proceedings of the IEEE Fourth Conference Monterey,California 5-9 June, pp 537 – 540.

Reason J. (1992), ‘*Human-Computer Interaction Is Not Enough: Considering Multi-Disciplinary Factors Underlying Human Erro*r’, CSERIAC Gateway, III(3), pp 10 – 11.

Rekimoto (1996); ‘*Transvision: A Hand-held Augmented Reality System for Collaborative Design’*, Jun, Virtual Systems and Multi-Media (VSMM)’96, 1996

Rich, E. (1979) ‘*User modelling via stereotype*s’, Cognitive Science , Vol. 3.

Rich, E. (1983) ‘*Users are individuals: individualising user models*’, International Journal of Man Machine Studies , Vol. 18

Rich, E. (1989) ‘*Stereotypes and user modelling*’. In: W. Wahlster and A. Kobsa (Eds.) op. cit.

Rivers, R. (1989) ‘*Embedded User Models. Where next?’,* Interacting with Computers, Vol. 1(1)

Rogers, S., & Langley, P. (1998). ‘*Interactive refinement of route preferences for driving’,* Proceedings of the AAAI Spring Symposium on Interactive and Mixed-Initiative Decision-Theoretic Systems (pp. 109-113). Stanford, CA: AAAI Press.

Roth (1993) S.F. Roth, W.E. Hefley, ‘*Intelligent Multimedia Presentation Systems: Research and Principles’*, in M. Maybury M. (ed), *Intelligent Multimedia Interface*s, AAAI Press, 13-58, 1993

Rouse, W.B. et al (1989) ‘*Intelligent Interfaces*’, Human-Computer Interaction , Vol. 3

Sanderson P. M. (1989), ‘*Supporting Emerging Human Roles in Advanced Manufacturing Systems: Cognitive and Organizational Aspect*s’, Research Report (EPRL-89-08),University of Illinois at Urbana-Champaign, Engineering Psychology Research Laboratory (EPRL), 1989

Sanderson P. M., Verhage A. G., Fuld R. B. (1989), ‘*State-Space and Verbal Protocol Methods For Studying the Human Operator in Process Contro*l’, Ergonomics,32 (11), pp 1343 - 1372, 1989

Schlimmer, J. C., & Hermens, L. A. (1993). ‘*Software agents: Completing patterns and constructing user interfaces’*. Journal of Artifcial Intelligence Research, 1, 61-89.

Schneiderman & M. McGill *(*1988) ; *'Direct Manipulation: A Step Beyond Programming Languages'*, B., IEE Comput. 16, 8 (Aug 1988), 57-69

Searle, J.R. (1969) (ed.), *'Speech Acts - An Essay in Philosophy of Language*', Cambridge University Press, 1969.

Seel, N. (1990) '*From here to Agent Theory*'. AISB Quarterly, no. 72 Spring
Self J. (1986) '*Applications of machine-learning to student modelling*', Instructional Science, Vol. 14.

Self, J. (1987) '*User modelling in open learning systems*'. In: J. Whiting and D. Bell (Eds.), Tutoring and Monitoring Facilities for European Open Learning (Amsterdam: Elsevier Science Publishers B.V.).

Shackel, B. (1990) '*Human Factors and Usability'* in Preece, J. and Keller, L. (Eds.) Human-Computer Interaction, Hemel Hempstead, UK: Prentice Hall

Shahidi A. K., Mcechan P., Terriza J. (1990), '*M4I Requirements For the SimulatorExemplar*', PROMISE ESPRIT Project 2397, PRO/HCI/34/1, Deliverable D40,1990

Sheth, B. & Maes, P. (1993), *'Evolving Agents for Personalized Information Filtering,':* Proceedings of the 9th IEEE Conference on Artificial Intelligence for Applications, pp. 113- 121, 1993.

Shoham Y (1993) *'Agent Oriented Programming'*, Artificial Intelligence 60 (1) , 51-92

Simon, H. A. (1969). '*The sciences of the artifcial*' . Cambridge, MA: MIT Press.

Skinner B.F (1974); '*About Behaviorism',*. RandomHouse Inc, NewYork, 1974.

Sleeman, D. (1985) '*UMFE: A user modelling front-end system'*, International Journal of Man Machine Studies , Vol. 23

Smith, S.L., & Mosier, J.N. (1984), *'Design Guidelines for the User Interface Software'*, Technical Report ESD-TR-84-190 (NTIS No. AD A154 907), U.S. Air Force Electronic Systems Division, Hanscom Air Force Base, Massachusetts, September 1984.

Sohier C., Bertels A. (1992), '*Task Analysis Model*', PROMISE ESPRIT Project 2397,Document No PRO/KUL/71.

Stanfill C. & D. Waltz (1986) *'Towards Memory Based Reasoning',* , Comm. ACM 29,12,(Dec. 1986), 1213-1228

Steels, L. (1987) '*The deepening of Expert Systems'*, AICOM, No 1, 9-16

Stehouwer, M. & Van Bruggen, J. (1989) '*Performance interpretation in an intelligent help system*'. In: Proc. 6th Annual ESPRIT Conference, Brussels, November 1989 (Kluwer Academic Publishers).Schneider-Hufschmidt, M., Kuhme, T. and Malinowski, U. (eds.) (1993)*Adaptive User Interfaces - Results and Prospects,* Elsvier Science Publications,North-Holland, Amsterdam

Sukaviriya, P., Foley, J (1993). '*Supporting Adaptive Interfaces a Knowledge-Based User Inter-face Environment'* . In: Gray, W. D., Hefley, W. E., Murray, D. (eds.): Conf. Proc. International Workshop on Intelligent User Interfaces. Orlando, FL. New York. ACM Press. (1993) 107-114

Sullivan & S.W.Tyler (1991) *'Intelligent User Interfaces"*,ACM Press N.Y.

Swezney (1991) Swezeny, R W *'Effects of Instructional Strategy and Motion Presentation Conditions on the Acquisition and Transfer of Electrochemical Troubleshooting Skill';* Human Factors 33,3, 309-323

Sycara K., (1995) '*Intelligent Agents and the Information Revolution',* In: UNICOM Seminar on Intelligent Agents and their Business Applications 8-9 November, London 1995, 143-159

Tang, H., Major, N., and Rivers, R. (1989) '*From users to dialogue*s'. In: L. Macauley and A. Sutcliffe (Eds.) People and Computers V (Cambridge: Cambridge University Press).

Thimbleby, H. (1990a) '*You're right about the cure - don't do that*', Interacting with Computers , Vol. 2(1).

Thimbleby, H. (1990b) '*User Interface Design',* (Wokingham: Addison Wesley).

Thomas, R.C., Benyon, D.R., Kay, J. and Crawford, K. (1991) '*Monitoring Editor Usage: The Basser Data Projec*t', in proceedings of NCIT '91 Penang.

Tilley,K.J (1996), '*Chapter 9 - Machining Task Allocation in Discrete Manufacturing Systems,'* Market-based Control, Scott H. Clearwater (ed.), World Scientific, pp.224-251, 1996.

Totterdell, P.A, Norman, M.A. and Browne, D.P. (1987) '*Levels of adaptivity in interface design*'. In: B. Shackel and H-J. Bullinger (Eds.), Proc. INTERACT '87, Second IFIP Conference on Human-Computer Interaction (Amsterdam: Elsevier Science Publishers B.V.).

Totterdell, P.A. & Cooper, M. (1986) 'Design and evaluation of the AID adaptive front-end to Telecom Gold.'

Trigg, R., Moran, T., and Halasz, F. (1987). '*Adaptability and Tailorability in NoteCards*', in Bullinger and Shackel (Eds.) Proc. INTERACT `87, North-Holland, 1987

Van der Veer, G.C (1990) *'Human-Computer Interaction. Learning, individual differences and design recommendations',* Offsetdrukkerij Haveka B.V., Alblasserdam;

Vicente, K. & Williges, R.C., (1988) '*Visual Momentum as a means of accommodating individual differences among users of a hierarchical file system*'. In: J. Rasmussen and P. Zunde (Eds.), Proc. 5th Symposium EFISS, Risø National Laboratory, Denmark, November 1987 (New York: Plenum Publishing).

Vicente, K. J. & Rasmussen, J. (1990), '*The Ecology of Human-Machine Systems II: Mediating 'Direct Perception' in Complex Work Domains*', Ecological Psychology, 2 (3), pp 207 - 249, Lawrence Erlbaum.

Vicente, K. J. (1991), '*Supporting Knowledge-Based Behavior Through Ecological Interface Design*', Research Report (EPRL-91-01), University of Illinois at Urbana-Champaign, Engineering Psychology Research Laboratory (EPRL).

Vicente, K.J. & Williges, R.C. (1987). '*Assaying and isolating individual differences in searching a hierarchical file system*', Human Factors , Vol. 29, 349-359.

Vicente, K.J. & Williges, R.C. (1988). '*Accommodating individual differences in searching a hierarchical file syste*m', International Journal of Man-Machine Studies ,Vol. 29.

Wahlster W. and Kobsa A. (1987) '*Dialogue-based user models*', Proc. IEEE Vol. 74(4).

Wickens C.D. (1984) '*Engineering Psychology and Human Performance',* Charles E Merrihill Publ.

Wickens C. D. (1992), '*Engineering Psychology and Human Performance (2nd Ed)*', Harper Collins Publishers Inc.

Wilensky, R., Arens, Y. and Chin D. (1984) '*Talking to Unix in English: an overview of* UC', Communications of the ACM , Vol. 27(6).

Williges, R.C. (1987) '*The use of models in human-computer interface design*', Ergonomics , Vol. 30(3).

Wilson, M.D., Barnard, P. J, Green, T.R.G. and Maclean, A. (1988) '*Task Analyses in Human-Computer Interactio*n'. In: C.C. van der Veer, T.R.G. Green, J.M. Hoc and D.M. Murray (Eds.) Working with Computers: Theory versus Outcome, (London: Academic Press).

Wooldridge & N.R.Jennings (1995) '*Intelligent agents: Theory and Practice*'. The Knowledge Engineering Review, 10(2):115-15

Wooldridge, M. (1995), '*Conceptualising and Developing Agents*', Proceedings of the UNICOM Seminar on Agent Software, 25-26 April, London, pp. 40-54, 1995.

Young, R. M. and Hull, A. (1982) '*Categorisation structures in hierarchical menus',* in *Proceedings of 10th International Symposium on Human Factors in Telecommunication*s, Helsinki. pp 111 – 118

Zissos, A. & Witten, I., I. (198*5) 'User modelling for a computer coach: a case Stud*y', International Journal of Man-Machine Studies, Vol. 23.

(URL1) Network Computer World, January 1998.
(URL2) http://www.symantec.co.uk/region/uk/product/inettools/vcafepde.html
(URL3) http://java.sun.com:81/products/hotspot/index.html
(URL4) http://www.alphaWorks.com
(URL5) http://www.zdnet.com/pcweek/reviews/0623/23hpj.html
(URL 6) http://www.twr.com/
(URL7) http://www.instantiations.com/jove.htm
(URL8) http://www.jivetech.com/redshift/product-brief.html
(URL9) http://www.idg.net/idg_frames/english/content.cgi?vc=docid_9-69514.html
(URL10) PC Magazine Java Speed Trials Graphic Threads 10/22/96
(URL11) http://www.volano.net/guide/mark.html
(URL12) http://www.borland.com/jbuilder
(URL13) http://www.asymetrix.com/products/supercede
(URL14) http://www.asymetrix.com/products/supercede/news/benchmarks.html
(URL15) http://www.zdnet.com/products/stories/reviews/0,4161,293221,00.html
(URL16) http://www.agentbuilder.com/
(URL17) http://www.ececs.uc.edu/~abaker/JAFMAS/
(URL18) http://java.stanford.edu/java_agent/html/
(URL19) http://www.networking.ibm.com/iag/iagsoft.htm
(URL20) http://www.crim.ca/sbc/english/lalo/
(URL21) http://www.iks.com/agentx.htm
(URL22) http://www.infosys.tuwien.ac.at/Staff/lux/Gypsy/
(URL23) http://www.agent-software.com.au/jack.html
(URL24) http://www.fujitsu.co.jp/hypertext/free/kafka/
(URL25) http://www.kinetoscope.com/via/default.htm

APPENDIX A: INTEGRATED DEVELOPMENT ENVIRONMENTS

JBuilder 2

JBuilder 2 (URLURL12) delivers an excellent interface, strong distributed computing options and powerful two-way programming tools.

Features

Debugger: JBuilder 2 includes an integrated set of tools for which aid debugging. The IDE gives easy access to panes of information on Threads, the Stack, Data, an Inspector, and an expression evaluation dialog

RMI and CORBA Support: JBuilder 2's support for distributed components is strong. It has support for distribution of components through CORBA and RMI. To aid the developer in using these systems JBuilder 2 has a powerful wizard that helps the process. The CORBA wizard works with some input from the developer to create an Interface Definition Language (IDL) file from the Java class, and can quickly compile the IDL file to complete the process. RMI is deployed through a similar wizard: They are almost identical, except for the generated output.

Database Support: JBuilder 2 includes InterBase database server and DataGateway middleware. This provides a set of tools for building a sophisticated Java solution.

JBuilder 2 provides a fast compiler that uses its proprietary Smart Dependencies Checking technology (SDC). The SDC results in fewer unnecessary compiles of interdependent source files, which in turn shortens subsequent edit/recompile cycles. When compiling, JBuilder 2 analyses the nature of the changes made to source files. Instead of deciding whether to recompile a source file based only on the time stamp of the file. A source file is recompiled only if it uses (or depends on) a particular element that has changed within another source file. The dependency checker only flags a file for recompilation if a method signature or data member changes. This allows the addition of comments to code or the ability to change a method body without recompiling the entire application.

Asymetrix Supercede

SuperCede (SC) (URL13) is the only tool that allows the use of pure Java to create Java byte code applets or applications, and still allow the creation of native binary executables, thus saving extra money on obtaining a native compiler.

Features

Built in Native Compilation option: The unique advantage of SuperCede is that it supports pure Java solutions that allow the generation and compilation of Java code to SuperCede's highly optimised Java virtual machine (VM). The SuperCede Java VM can be used to compile code on the fly or as a Web browser plug-in. The SuperCede VM provides a high-performance Java runtime program that boosts the performance of Java to C++ speeds. Asymetrix claims performance that is 50 times better than interpreted Java and up to 5 times better performance than the best *just-in-time* (JIT) Compiler and Builders. See (URL14) for more information on their claims.

Debugger: SuperCede has a two additional and powerful features called *trace points* and *action points*. Trace points are set like breakpoints, except messages can be output to the debug window instead of stopping execution. Action points allow the programmer to execute a piece of code each time a specified point is reached, without necessarily halting the execution of the program.

Libraries: Asymetrix has bundled a number of other libraries with SC. These include Netscape's Internet Foundation Classes (IFC), ObjectSpace Inc.'s Java Generic Library (JGL), and Object Design Inc.'s ObjectStore Persistent Storage Engine (PSE).

C++ Integration: Supercede implemented its Java and C++ classes using the same object model, and therefore provides seamless integration with C++. This facility is useful in creating Java wrappers for C++ programs or C++ wrappers for Java programs, so that either can call the other. Using SuperCede C/C++ code can be called directly, or by using the more complicated Java Native Method scheme. With the included support for C++ legacy code, it is possible to leverage existing code base and expertise.

Symantec Visual Cafe

Visual Café (URL2) is heavily used in both academia and industry, and is renowned for it's ease of use and intuitive user interface.

Features

Compiler: Symantec's JIT (Just-In-Time Compiler and Builder) claims to be the fastest in both compile and runtime in the industry and is now included in JavaSoft's JDK 1.1 as the Performance Runtime for Windows. Visual Cafe project management is also among the best. Projects can contain sub-projects, and developers can call batch files to perform custom functions, such as call other tools. Visual Cafe integrates easily with several popular configuration management tools, including SCCS and RCS.

Debugger Visual Café's debugger is full-featured, and has remote debug capability—that is, the developer can actually debug Java that is running in another machine. The debugger is tightly integrated with the Java virtual machine, allowing Java code on the fly to be entered and the results examined.

Libraries: VC includes about 100 visual components beyond the basic AWT widgets. One valuable feature is that users can add their own objects to the tool palette—a feature exclusive to VC at this time.

Comparison Of Java IDE's.

The following results were obtained from (URL15)

Output File Sizes of a Sample Java Application	
	Bytes - Lower score is better
CodeWarrior Professional 2.0	35,152
JBuilder Client/Server Suite	41,6 89
Sun Java WorkShop 2.0,native	40,403
Sun Java WorkShop 2.0,fast compiler	50,145
Sybase PowerJ Enterprise 2.0	34,140
Sybase PowerJ Enterprise 2.1	34,477
VisualAge for Java 1.0	34,501
Visual Café for Java 2.1,optimized	35,476
Visual Café for Java 2.1,unoptimized	41,689

Table 25: Output File Sizes of Sample Java Application

Compile Times for Native Compiler

Full debug build Partial debug build	**Chess Application** Seconds - lower score is better	**JMark 1.02**Seconds - lower score is better
CodeWarrior Professional 2.0	6.5 4.4	9.4 5.2
JBuilder Client/Server Suite	1.9 0.8	1.8 0.8
Sun Java WorkShop 2.0,native	8.7 6.1	10.1 6.4
Sun Java WorkShop 2.0, optimised	2.2 1.7	2.8 2.2
Sybase PowerJ Enterprise 2.0	6.2 6.4	8.2 8.4
Sybase PowerJ Enterprise 2.1	8.6 2.7	8.6 3.8
VisualAge for Java 1.0	N/A	N/A
Visual Cafe for Java 2.1,Database Development Edition	1 1	1.3 1.2

N/A—Not applicable: The product does not have a discrete compile step.
Table 26: Compile Times for Native Compiler

Conclusion

For ease of use and power the choice comes down to JBuilder 2 and Visual Café. Of these two, JBuilder 2 best fits the agent developers needs on several fronts. First it is 100% pure Java compliant and can therefore integrate with JDK1.2. Secondly, it has interface support tools for Swing and lastly it has excellent support for the distributed application utilising its client/server paradigm, fully supporting RMI and CORBA. Additionally, the supplier Inprise has bundled its excellent VisiBroker libraries with JBuilder 2, which allow easy integration of applications into a CORBA environment.

The tables above illustrate that on performance alone, there is no significant difference between the two packages. Unlike Visual Café, JBuilder 2 offers high-end features, like explicit transaction processing and full record navigation in data aware Pure JDBC™-compliant Java components.

One of the primary elements of any agent environment is a fast compiler. The advanced Java compiler in JBuilder 2 has two features that are not available in Visual Café: *SmartChecker* for smart dependencies checking, and a source code obfuscator for code protection.

The Jbuilder 2 also includes a source code obfuscater that Visual Café does not, that makes it more difficult to reverse-engineer byte-code Java class files.

Visual Café and JBuilder 2 both have excellent professional debuggers with similar features. However, JBuilder 2 provides an especially powerful tool, which will be very useful within any agent project, a *multi-thread debugger*. JBuilder 2 can do this by switching from one thread to another using the Thread/Stack pane in the AppBrowser while debugging. The breakpoints can be set to be active only for a certain thread.

Thus the recommendation here is to use JBuilder 2, which has most of the main features an agent developer requires of an IDE. These features are summarised below:

Integrated RMI and VisiBroker CORBA/IIOP development tools: With its CORBA and RMI integration, JBuilder 2 makes it easier to build and deploy distributed applications in heterogeneous environments.

Integrated Version Control is provided by Intersolv's market-leading *PVCS Version Manager*, to help manage team development. Visual Café includes Starbase for version control.

SQL Explorer enables developers to browse, modify and manage data on SQL databases like Oracle, Sybase, Microsoft SQL Server, DB2, Informix, and InterBase.

SQL Builder is a graphical SQL query tool that automatically generates ANSI SQL-92 queries.

SQL Monitor enables developers to view the execution of SQL queries for performance tuning and testing.

DataGateway for Java provides developers with a multi-tier, fast and reliable database connectivity solution adhering to the industry standard, JDBC. This provides native connectivity to Oracle, Sybase, DB2, Microsoft SQL Server, Informix, Borland InterBase, Paradox, dBase, FoxPro, and MS-Access; plus, additional connectivity through standard ODBC drivers; providing more native connectivity options than any other middleware product in the market, including Symantec dbANYWHERE.

APPENDIX B: AGENT TOOLKITS

Agent Builder

Summary Of Features

AgentBuilder (URL16) is an integrated tool suite for constructing intelligent software agents. AgentBuilder consists of two major components - the Toolkit and the run-time System. The AgentBuilder Toolkit includes tools for managing the agent-based software development process, analysing the domain of agent operations, designing and developing networks of communicating agents, defining behaviours of individual agents, and debugging and testing agent software. The run-time system includes an agent engine that provides an environment for execution of agent software.

Agents constructed using AgentBuilder communicate using KQML. In addition, AgentBuilder allows the developer to extend the standard KQML performatives to include any additional performatives deemed necessary.

All components of both the AgentBuilder Toolkit and the run-time System are implemented in Java. Likewise, the agents created with the AgentBuilder Toolkit are Java programs so they can be executed on any Java virtual machine.

The AgentBuilder toolkit is designed to provide the agent software developer with an integrated environment for quickly and easily constructing intelligent agents and agent-based software.

Conclusion

This commercial agent toolkit is largely aimed at the enterprise market in non-time critical situations. It has many good features including support for the difficult task of designing and defining individual agent behaviour, as well as system behaviour. To this end it includes a highly useful *Ontology Manager* and tools to help define the agent rule base (for planning and learning).

In addition, and perhaps one of its most useful features, AgentBuilder provides, as part of its Agent Manager an agent *visualiser* so that agent interactions can be tracked to aid system debugging. Also, it has an Agent Debugger to monitor and communicate with executing agents. AgentBuilder is based upon, but not restricted to, the *strong* notion of agency and thus agent behaviour is defined in terms of beliefs, desires and intentions.

AgentBuilder appears to be a fine toolkit, it satisfies many assessment criteria, it is KQML compliant, pure Java and is FIPA compliant (At least it appears to be, its Agent Manager looks to be equivalent to FIPA's Directory Facilitator Agent). Also its agent management tools are a tremendous asset as they save the developer building their own debugging agents. However, it does not appear to be widely used in the agent community. Perhaps this is because Reticular systems do not provide evaluation copies, and the system costs around $900. As a result the toolkit was unavailable for evaluation.

The only doubts that can be raised against this agent toolkit are the runtime performance of its agent engine. Each agent makes decisions based on its rule base, AgentBuilder allows simple construction of these rules, however at runtime they are marshalled by a central Inferencing Engine, which can conceivably leads to additional lags in communication times. There are no official figures on the speed of interactions, and without trying the actual program, it is difficult to assess its throughput speed and KQML parsing. However, it is believed that the agent engine runs on one iteration per millisecond. Much of this commentary is supposition since the toolkit was not tested.

Overall, on paper this is the one of the best toolkits around, but there could performance and scalability problems.

FOR: FIPA Compliant, KQML Compliant, Agent Debugger, Agent Visualiser, Agent Project Management, Ontology Manager, Ability to implement Planning and Learning facilities, Provides Communications Platform, Pure Java.

AGAINST: Performance, Scalability, and Price.

JAFMAS

Summary of Features

JAFMAS (URL17) provides a framework to guide the development of multi-agent systems, together with a set of classes for agent deployment in Java. The framework is intended to help developers structure their ideas into concrete agent applications. It directs development from a speech-act perspective and supports multicast and directed communication, KQML or other speech-act performatives and analysis of multi-agent system coherency and consistency.

Conclusion

JAFMAS along with JATLite is one the most popular agent tools in the academic community, although it appears to be rapidly losing ground to its competitor. This can be attributed to a number of factors, not least the multitude of bugs that seem to crop up on a regular basis. In addition although it supports the use of speech-act performatives it does not inherently use the KQML standard, which could be a problem.

That point aside, the important feature of JAFMAS that distinguishes it from traditional systems that require some sort of Agent Name Server (ANS), is that JAFMAS offers a unique multicast facility. The infrastructure allows agents to establish connections with each other, and establish each other's identity, by using multicasting (distinct from broadcasting) without the use of any central registry such as an ANS. This is an important point because this kind of architecture is better suited for a closed system where all the agents know who they need to communicate with.

Advantages include RMI based communication API's and a well-developed agent architecture. Disadvantages with JAFMAS included being *very* hard to get set-up and working correctly. The current version seems to be bug ridden, and requires deep knowledge of RMI to get working.

FOR: Scalable, Fast, Supports peer-to-peer communication, Comes in the form of standard Java packages, support speech-act performatives, Pure Java.

AGAINST: Difficult to set up, unreliable, no direct support for KQML, Basic in form, relies solely on Java's built in security mechanisms.

JATLite

Summary of Features

JATLite (URL18) takes the form of a set of Java packages and a Java runtime router. JATLite provides a basic infrastructure in which agents register with an Agent Message Router facilitator using a name and password to connect/disconnect. Once connected to the router, agents are able to send and receive messages, transfer files, and invoke other programs or. JATLite fully supports the KQML standard, and provides a robust lightweight communications platform, with in-built FTP and SMTP extensions. JATLite in based on the client/server mechanism and was built, primarily, to be used with Applets rather than Applications. However, it works equally well with either.

Conclusion

JATLite, like JAFMAS is notoriously difficult to install and requires significant network experience to set-up. However, once running it affords a flexible and useful agent infrastructure. JATLite is, probably, the most widely used agent toolkit within the Agent research community. It is quite well supported and new extensions to its functionality appear quite regularly.

Unlike JAFMAS, JATLite uses a centralised router to direct communication between the agents. The JATLite router does have a unique robustness feature. It buffers all messages. An agent can therefore retrieve all outstanding messages - in fact, all messages that it has not explicitly deleted. The author has used JATLite for some time now and would recommend it as an agent platform in the general sense. However, serious doubts can be raised about its performance in a real time system, where the KQML parser in the router would hinder the timely delivery on messages. Having talked to some of its developers about this issue, it was discovered that a new faster KQML parser is due for release soon.

FOR: KQML Compliant, FIPA Compliant, Robust communications platform, well supported, Pure Java.
AGAINST: Difficult to set-up, Slow KQML parser, Uses centralised router

IBM-ABE

Summary of Features

IBM's Agent Building Environment (ABE) (URL19) provides a communication infrastructure and an inference engine to dictate agent behaviour. In the current version, the intelligent agent watches for a certain condition, and decide what to do based on a set of rules, and triggers an action as a result.

The architecture for the agent is based on reasoning engine, and adapters that allow the agent to interact with the rest of the world. The developer can construct custom adapters to interface with custom applications or legacy systems.

Conclusion

The ABE places a great emphasis on providing a set of tools to develop the functionality of an autonomous agent. In this way it is similar to AgentBuilder, at its core lies an inference engine, which allows the agent to react to changes in the environment according to its rule set. Where it falls down is its lack of support for KQML and ontology checker.

The ABE is similar in some ways to AgentBuilder except that it is far less useful. One of its redeeming features is the number of adapters (API extensions) that let the agent interact with other systems. Other than that it was found that the system was *very* difficult to get running, and the architecture was muddled and convoluted. In addition, the inference engine was slow and its rule builder software was crude and unreliable.

FOR: In-built rule and planning facility, Useful adapters.

AGAINST: VERY difficult to get running, messy architecture, slow inference engine, crude rule builder.

LALO

Summary of Features

LALO (URL20) is a programming environment, which permits the development of multi-agent systems. The architecture is extensible and allows the creation of multi-agent systems including reactive agents and deliberative agents. The inter-agent communication language is KQML. A program written in LALO is translated into C++ source code, and then compiled with a C++ compiler.

Conclusion

LALO is a good agent infrastructure but is let down by not being Java compliant. In addition, an inherent part of AOP is the strong notion of agency and therefore includes the BDI model (Beliefs, Desire and Intentions). Within the scope of many multi-agent projects, the weak notion of agency is used, whereby developers do not imbue their agents with BDI concepts, rather the system gains value through the combined interactions of a number of simple agents.

FORS: Flexible architecture, established, FIPA compliant.

AGAINST: Written in C++, uses strong notion of agents.

Agent X

Summary of Features

Agent X (URL21) provides distributed computing libraries that support object request broker, RMI and mobile agent services written in Java. The libraries were designed to provide object request broker facilities that were easier to use and more functional than the RMI libraries bundled with the Sun JDK

AgentX also provides programming support for the creation and release of autonomous mobile agents

Unlike RMI and CORBA AgentX does not require that stubs or skeletons be created, or that any IDL be written.

The only requirement is that an AgentX server application be run on a machine that will respond to RMI and mobile agent requests. The server component uses only a single TCP/IP port to efficiently handle both types of requests.

Conclusion

AgentX is a commercial package that provides a very good communications infrastructure. Its use of normal Java API packages as tools for communication provides one with an instantly understandable and efficient transport mechanism. This tool is relatively new and not very well supported. Its communication platform improves on RMI.

FOR: Easy to use communications platform, very lightweight, scalable, FIPA compliant.

AGAINST: Targeted for use in mobile agent market, little support. Offers little else in agent functionality. No KQML support.

Gypsy

Summary of Features

Gypsy (URL22) is a project investigating mobile agents, but it does include within its framework the ability to support a lightweight communications infrastructure, due to its support of RMI and a CORBA like registry, it includes the following features:

JavaBean mobile agents (supervisor/embedded/normal).

JavaBean places as special mobile agents.

RMI and CORBA IIOP communicators.

CORBA Mobile Agents MAF like registry using RMI.

Secure JAR classloader & security manager.

GUI remote administration and configuration tool.

Conclusion

The prime motivation for this toolkit is not to support a multi-agent framework as traditionally envisaged, rather to implement a toolkit that provides roving agents over an open network. Further, it does not include any support for KQML or have any active guidelines on development. It's communications platform is relatively difficult to get to grips with, and is completely set-up for use with mobile agents.

FOR: GUI tools, support for Java Beans.

AGAINST: For use with Mobile agent systems, communications platform is an extension of RMI and provides little in the way of ease of use, no KQML support.

Jack Agent Toolkit

Summary of Features

Jack (URL23) proves an in-built architecture based on a communications infrastructure. Thus, it is a lightweight architecture that allows the developer to build whatever types of agents they require on top of the base kernel.

Features:

Allows easy integration using standard infrastructure, such as CORBA, RMI, HLA or DCOMM.

Has a language specification and object oriented design targeted to allow easy extension for new agent models

Uses JACK Agent Language, JAL, which follows the standard JAVA/Object Oriented paradigm.

Includes user interfaces for the development and debugging of agent applications.

The Jack Agent Language reduces the learning curve and JACK's type-safe and object-oriented approach assists in developing more reliable applications.

Is extremely lightweight - is designed to handle hundreds of agents running on low-end hardware.

Conclusion

JACK seems to provide a very good solution to many difficult agent problems. Its communications mechanism is lightweight. It offers an agent language which is an extension of Java and therefore intuitive and easy to use. Further, unlike many agent toolkits it includes an agent debugger, which is a very useful addition.

Unfortunately, further comment cannot be made on this toolkit, as information on JACK is available only via email with the company itself. Several emails were sent to Agent-Oriented

Software and, as of yet, no reply has been received. Thus without the software to hand it is hard to judge its use. However the author did get in contact with someone who has been involved in some consultancy on JACK and was told that it contains some excellent libraries for assigning a single agent to handle multiple threads.

FOR: Formal agent language, debugger, good communications platform, pure Java, very scalable.

AGAINST: Not widely used, hard to obtain and test software. No KQML implementation.

Kafka Agent Library

Summary of Features

Kafka (URL24) is based on Java's RMI and has the following features:

Runtime Reflection:

Agents can modify their behaviour (program codes) at runtime. The behaviour of the agent is represented by an abstract class Action. It is useful for remote maintenance or installation services.

Remote Evaluation:

Agents can receive and evaluate program codes (classes) with or without the serialized object. Remote evaluation is a fundamental function of a mobile agent and is thought to be a push model of service delivery.

Distributed Name Service:

Agents have any number of logical names that do not contain the host name. The distributed directories can manage these names.

Customisable security policy

A very flexible, customisable, 3-layered security model is implemented in Kafka.

100% Java and RMI compatible:

Kafka is written completely in Java. Agent is a Java RMI server object itself, so agents can directly communicate with other RMI objects.

Conclusion

Kafka provides *very* lightweight support for multi-agent architectures in the form of Java API libraries. The libraries are imported in the form of packages and used as normal Java classes. It does provide fast, scalable support for an agent system. However, it does not have any support for KQML or for higher-level agent functions.

Kafka is not really an agent toolkit; rather a collection of packages, which aid agent designers by providing custom, classes for dealing with a distributed collection of objects. The libraries are not well chaptered and operate at a low level. Kafka requires a good deal of familiarity with RMI and distributed computing. The very fact that these API's operate at a low level means Kafka offers a good deal more flexibility than most toolkits, as many layers can operate above the Kafka level. In terms of use within agent projects, this may not be the toolkit of choice. However, the beauty of Kafka is that agent developer can selectively integrate some of its more useful API's within any other agent toolkit used (as long as they are based on RMI). Thus the reflection library could provide a very valuable addition for an agent developer.

FOR: Flexibility, reflection API, *very* lightweight communications platform, pure Java, based on Java packages, collection of libraries.

AGAINST: Badly chaptered requires good knowledge of RMI. Not user friendly.

Via Agents.

Summary of Features

Via (URL25) is a commercial agent package for development of multi-agent systems. It provides the following features:

> The Via agent server: Via's Agent Manager server manages all agent activity while agents are in the field. The primary advantages of Via's Agent Manager are that it is cross-platform, scalable, and persistent.
>
> Pre-Built Client GUIs: Via comes with customisable end-user interfaces for dynamically creating, controlling and describing the behaviour of agents in the network.
>
> Multiple sensory and action modules: A complete set of agent "tasks" and agent "actions" allow agents to interact with on-line resources such as databases, LDAP directories and Web sites, and communicate with users.
>
> Notification capabilities: Via includes a suite of ready-to-use telecommunications capabilities which allow users to receive notifications eight different ways - through text-to-speech based telephony, alphanumeric pager messages, e-mail messages, changes to HTML pages, faxes, Java client GUIs and using "push" channel technology in Microsoft's and Netscape's browser products.
>
> The Via System API (VSAPI): VSAPI allows developers to create custom agents and to extend and modify the agents and agent capabilities that come with the Via System.

Conclusion

Via appears to be a very useful client/server suite, and has been designed to offer services, directly, to various users over an Intranet. Thus, the whole configuration is set-up on a user-agent basis rather than an agent-agent basis. In essence, Via agents perform tasks on behalf of the user directly (Such as search web pages, filter email etc) and interact with their direct environment rather than other agents (although they do have this capability).

Users can apply changes to agents directly via the GUI that is provided by Kinetiscope. This severely limits the developers ability to set-up a custom GUI. Users can alter the behaviour of their agent by http, cgi etc. This is a rather good agent toolkit, but sadly it is too rigid in its format for common use. Because the whole toolkit is set-up for direct interaction with the user, it would be of little use when trying to define agent-agent interactions. Predominantly this is because Via agents can only interact with their environment via *SMTP, HTTP* etc and not faster more reliable protocols such as TCP/IP or UDP.

FORS: Simple to use, simple architecture, good array of added agent functionality (FTP,EMAIL,CDF,NNTP), well designed GUI's, robust.

AGAINST: VERY inflexible. No support for TCP/IP. Can only use pre-defined GUI's. No inter-agent communication supported.

Object Space Voyager

Summary of Features

Voyager (URL26) is a 100% Java agent-enhanced Object Request Broker (ORB). It combines the power of mobile autonomous agents and remote method invocation with full CORBA support and comes complete with distributed services such as directory, persistence, and publish subscribe multicast. Voyager caters for both traditional and agent-enhanced distributed programming techniques.

Voyager uses regular Java message syntax to construct remote objects, send them messages, and move them between applications. Voyager allows agents to move themselves and continue executing as they move. In this way, agents can act independently on the behalf of a client, even if the client is disconnected or unavailable.

Conclusion

The Voyager platform would be recommended for any system that requires an agent toolkit, which provides a reliable, easy to use communication substrate. Voyager is a set of Java class libraries and server programs that provide - in a simple and straightforward way - a tremendously rich and flexible foundation for all sorts of distributed computing applications, including agent systems. What Voyager will do for development work is "take the strain" out of object distribution, remote messaging, and routing and distributed events. This allows the developer to concentrate on agent functionality rather than *how* it will be distributed. In fact, the developer can make existing objects distributed without changing the original source code,

Voyager also comes bundles with ObjectSpaces Java Generic Libraries(JGL), one of the most widely used and thoroughly tested Java APIs in the world, and has been licensed for inclusion in almost every major Java IDE. It provides a set of Java classes for data processing which fully optimise compile, network load, and execution times which along with a Native compiler should help any distributed agent system be even better equipped to handle real-time or multiple signals.

Of the agent packages reviewed here, Voyager is by far the best platform for pure distributed computing. It does not, however, have any support for KQML. Apart from this Voyager was found to be easy to learn, beautifully constructed, intuitive and joy to use

FOR: Easy to use, powerful, pure Java. FIPA compliant. Highly flexible, well supported, fast and free.

AGAINST: No KQML support, more of a communication platform. No inherent agent support principles.

BOND Distributed Object System

Summary of Features

The Bond (URL27) distributed object system provides a message oriented middleware environment for developing distributed applications. Bond uses the KQML language for object communication. The message space of Bond is divided into sub-protocols. Closed set of messages that can be viewed as task oriented micro languages. Examples of sub-protocols are, the property access, persistent storage access or security sub-protocols.

Bond objects can be extended with new sub-protocols by the means of probes. Probes implement the functionality of a specific sub-protocol and are attached as dynamic properties to the given Bond object. A special class, called pre-emptive probes process the message before it is delivered to the object, so they can act as filters for security, accounting or logging purposes. For example, servers can be implemented without concern for accounting or monitoring. These aspects can be added as probes during runtime.

Bond executables usually run as threads in the runtime environment provided by a resident. The resident provides the messaging thread and the local directory service for the running executables. Although the Bond programs can run in standalone mode, they can be more efficiently used by grouping them in a domain. A domain contains a number of core servers such as the directory server, persistent storage server, authentication server and the monitoring agent.

The agent framework of the Bond system simplifies the task of developing agents by allowing the programmer to concentrate on the specific strategies of a new agent. Bond agents have the intrinsic capability to be controlled remotely and to co-operate with each other. The task of an application programmer is limited to specify the agenda, the finite state machine of the agent, and the strategies associated with each state.

Bond agents can be specified by their "blueprint". The blueprint is not a full-featured programming language: the various aspects of agent strategies have still to be programmed in Java. However, the database of ready-made strategies allows the most common aspects of the agents to be assembled from the strategy database without the need of programming. The blueprint provides the assembly instructions of the agent, which are used by the bondAgentFactory object to assemble the agent during runtime. The blueprint of an agent implicitly defines its control sub-protocol, which can be used by an external object to control the agent.

The Bond system is currently under implementation at the Bond Lab at Computer Science Department of Purdue University. A beta version is planned for late January 1999.

Conclusion

The BOND agent system looks very promising indeed and could offer the agent developer a very attractive option. This looks like the nearest rival to AgentBuilder. However the BOND tool is *very* new and the author was unable due to time constraints to look in depth, or indeed even try the very early Alpha version). For further details please read (URL28).

Other Agent Toolkits.

Introduction

This section details some of the other available toolkits for general information purposes. Few of those described below are flexible enough (most adhere strictly to their own architecture) to be adapted to a generic agent architecture.

AARIA Agent

AARIA(URL29) agents aim to decentralise manufacturing systems by connecting, via agents, various distributed aspects of the manufacturing system. The MES functionality includes basic "what-if" simulation, finite capacity scheduling, and intelligent shop floor interfaces. The ERP functionality includes basic planning, order entry, purchasing, bill-of-materials management, inventory management, resource management, personnel management, integrated financials, and reporting.

AARIA is designed to dialog with customers and suppliers and allocate resources to new jobs as they enter the system, to optimise schedules across resources, to recover from faults in the factory, to dispatch work against the schedule, and to report results. It is designed to exhibit this functionality either when running an actual factory or when in a simulation or "what-if" mode.

The Agent Building Shell

The Agent Building Shell (URL30) provides several reusable layers of languages and services for building agent systems: co-ordination and communication languages, description logic based knowledge management, co-operative information distribution, and organisation modelling and conflict management. This approach is being applied in the area of manufacturing enterprise supply chain integration.

Echelon

Echelon (URL31) has made an architecture for developers who want to build communications and intelligence into their products that sense, monitor, or control. Nodes, or intelligent control devices, communicate with one another and can perform necessary control functions.

InteRRaP

InterRRaP (URL32) is a layered architecture that has been designed to model autonomously interacting agents. It is meant for designing complex dynamic agent societies for scheduling applications and for robotics. It defines agents by how actions are defined from the agents perception and mental model, belief revision and abstraction, situation recognition, goal activation, and scheduling perspectives. It is available on UNIX only.

APPENDIX C: USABILITY RESULTS

Results Of The Workshops

Quantitative results

Genoa Workshop

This section reports the cumulated results derived from the forms filled by the participants to the two workshops held in Genoa in Elsag and in ENEL respectively.

Interview

Question	1	2	3	4	5	6	M	Comments
Part 1							**4.9**	
Display of information							**5.2**	
Which is your overal impression of the system? (1)	5	3	4	6	7		**5**	Idea is good, but excessive automation should be avoided
Do you think the system provides information about the process state in a good way? (2)	6	5	5	6	7		**5.8**	An "operator guide" should be available
Are important parameters emphasised in the interface? (2)	5	4	5	5	7		**5.2**	
Does too much information exist in the display? (2)	5	6	3	3	4		**4.2**	Information amount should depend on the process and adapt to it
Is the most important information easy to find? (3)	6	2	5	7	7		**5.4**	An option to stop the sequencing of information should be available
Is the amount of information sufficient? (3)	6	5	6	6	4		**5.4**	Depends on the complexity of the network
Do you find the way of presenting the information logical? (3)	5	6	4	5	7		**5.4**	Generally yes
Effectiveness							**4.2**	
Is it possible to carry out tasks easy and logically? (3)	4	2	-	6	6		**4.5**	
Are you able to carry out your tasks efficiently when using the system? (4)	4	3	-	7	-		**4.7**	
Is it easy to prevent errors? (5)	4	2	-	4	-		**3.3**	
Efficiency							**5.5**	
Does the system provide you with a good overview of the process? (6)	4	3	6	5	7		**5**	Not enough information to give an assessment
Do you find the information you need to work in the system? (3)	-	5	6	6	7		**6**	Not enough information to give an assessment
Is the amount of information sufficient? (3)	5	4	6	6	7		**5.6**	Not enough information to give an

								assessment
Is the Interface Pleasing to Use							**4.9**	
Do you like working with the system? (6)	4	4	5	5	-		**4.5**	Non applicable
Do you like the way information is presented? (6)	5	3	6	7	5		**5.2**	Not enough information to give an assessment
Do you find the system pleasing to interact with? (6)	6	3	-	6	-		**5**	Not enough information to give an assessment
Do you feel in control of the system? (6)	4	4	-	6	-		**4.7**	Not enough information to give an assessment
How easy is it to learn to work with the system? (3)	5	5	6	4	-		**5**	
Is the information presented supporting your main tasks? (3)	5	5	6	4	-		**5**	
Part 2							**5.2**	
Alarms							**5.2**	
How easy is it to diagnose faults by using alarms? (3)	5	5	6	6	6		**5.6**	
Is the alarm system making you attentive when deviations occurs in the system? (3)	5	5	6	7	6		**5.8**	
Does the alarm system provide sufficient information on the priority and cause of the deviation? (3)	4	2	6	6	7		**5**	
Are the feedback you get from the alarms helping you to decide if you carried out the right action? (3)	4	2	-	4	-		**3.3**	It depends on the experience of the operator
Are the alarms presented consistent with other information on the screens? (7)	5	3	6	6	6		**5.2**	
How do you evaluate presentation of alarms? (3)	3	2	6	6	6		**4.6**	
Is the alarm text understandable? (3)	-	2	6	7	1		**4**	
Are the alarms organised so that natural relationships between alarms are shown? (3)	-	6	7	7	6		**6.5**	
Is vocal input/output helpful? (3)	-	6	7	7	6		**6.5**	
Do you believe that multimedia are a good support fro the network control? (3)	-	6	7	7	6		**6.5**	Some people believes too expensive

(1) Very Bad – very good (2) False – True (3) No – Yes (4) Very little– A lot (5) Very difficult – very easy (6) Little – Much (7) Bad – good

The evaluation checklist

1 = never 2 = sometimes 3 = almost always 4 = always

Question	1	2	3	4	5	6	M	Comments
Section 1: Clarity of the representation				**3.1**				
1. Is important information highlighted on the screen?	-	3	3	3	-		**3**	
2. Does information appear to be organized logically on the screen?	3	3	3	3	-		**3**	
3. Are bright or light colors displayed on a dark background and vice versa?	4	1	3	4	-		**3**	
4. Does the use of colour help to make the displays clear?	4	3	2	4	4		**3.4**	
5. Where colour is used, will all aspects of the display be easy to see if used on a monochrome or low-resolution screen or if the user is colour blind?	3	2	4	4	-		**3.3**	
6. Is the information on the screen easy to see and read?	3	2	2	3	-		**2.5**	
7. Does the screen appear uncluttered?	-	-	3	4	3		**3.3**	
8. Are schematic and pictorial displays clearly drawn and annotated?	3	2	3	4	-		**3**	
9. Is it easy to find the required information on a screen?	3	2	3	4	-		**3**	
10. Are voice messages useful and clear?	3	2	3	4	-		**3**	
11. Do vocal commands help in fulfilling the task?								
12. It is easy to interact with the system using vocal command?	3	3	4	4	4		**3.6**	
Section 2 Functionality				**3.2**				
1. Is the way in which information is presented appropriate for the tasks?	4	3	4	3	3		**3.4**	
2. Does each screen contain all the information that the evaluator feels is relevant to the task?	3	3	3	4	4		**3.4**	
5. Is system feedback appropriate for the task?	3	3	-	4	-		**3.3**	
6. Does the system help the evaluator to understand the state of the process?	4	3	-	4	4		**3.8**	
7. Does the system support the evaluator in feeling in control of the process?	4	2	2	4	4		**3.2**	
8. Does the nature of adaptation ensure that the evaluator is warned of a process disturbance in time?	4	3	3	3	2		**3**	
9. Are the adaptation strategies consequent?	4	3	3	3	2		**3**	
10. Is the media chosen for adaptation appropriate?	4	3	3	4	3		**3.4**	
11.Are the voice media effective?	3	3	2	4	-		**3**	
11. Does the system decrease the users' workload?								
13. Is vocal information appropriate?								
14. Does the system ensure that important information is presented for the evaluator at appropriate times?	3	2	2	4	-		**2.8**	
Section 3: Alarms and error messages				**3.1**				
1. Does the system clearly warn the evaluator about a disturbance or a deviation from a normal situation?	3	2	3	4	4		**3.2**	

Question	1	2	3	4	5	6	M	Comments
2. Is the alarm of such a nature that it interrupts the evaluator from what he is doing?	2	3	1	3	4		**2.6**	
3. Is a normal situation indicated by the absence of colours and sounds that demand attention?	4	4	4	4	4		**4**	
4. Is the system designed so that alarms that demand immediate attention are presented in a way that they are perceived and understood in all situations, including situations with a high degree of disturbance?	3	1	4	4	4		**3.2**	
3. Is a normal situation indicated by the absence of colours and sounds that demand attention?	3	3	2	4	2		**2.8**	
9. Are voice alarms helpful to call for operator attention?	4	3	2	4	1		**2.8**	
6. Can audible alarms be heard despite background-sound?								
8. Are voice commands helpful in carrying out parallel activities								
7. Does the system provide the possibility of turning an alarm sound off from several places in the control room?								
5. In the alarm display, is there enough space to show all high priority alarms at the same time?								
Section 4: Information Feedback				**3**				
1. Are instructions and messages displayed by the system concise and positive?	4	4	4	4	-		**4**	
2. Are messages displayed by the system relevant?								
3. Do instructions and prompts clearly indicate what to do?	3	2	3	4	-		**3**	
4. Is it clear what actions the user can take at any stage?	3	2	3	3	-		**2.8**	
5. Are status messages informative?								
6. Does the system clearly inform the user when it completes a requested action?	3	2	4	4	-		**3.3**	
7. Does the system promptly inform the user of any delay, making it clear that the user's input or request is being processed?								
9. Is it clear to the user what should be done to correct an error?	4	2	3	4	-		**3.3**	
10. Do alarm messages inform the evaluator about the priority and nature of the deviation?	3	1	4	4	-		**3**	
11. Do alarm messages guide the evaluators' initial actions?	2	1	3	3	-		**2.3**	
12. Do alarm messages verify whether the evaluator's response corrected the deviation?	2	1	1	4	-		**2**	
Section 5: Consistency				**3**				
1. Are the different colours used consistently throughout the system?	3	2	3	3	-		**2.8**	
3. Are icons, symbols, graphical representations and other pictorial information used consistently	3	3	3	4	-		**3.3**	

Question	1	2	3	4	5	6	M	Comments
throughout the system?								
4. Is the same type of information displayed In the same location on the screen? b) In the same layout?	3	3	3	4	-		**3.3**	
5. Is the same item of information displayed in the same format, wherever it appears?								
6. Is the format in which the user should enter particular types of information on the screen consistent throughout the system?								
8. Is the method of selecting options consistent throughout the system?	-	2	2	4	-		**2.7**	
Section 6: Compatibility				**3**				
1. Are colours assigned according to conventional associations where these are important? (e.g. red = alarm, stop)	4	3	3	4	-		**3.5**	
2. When abbreviations, acronyms, codes and other alphanumeric information are displayed, are they easy to recognize and understand?	4	1	2	3	-		**2.5**	
7. Does the organization and structure of the system fit the user's perception of the task?	3	2	2	4	4		**3**	
8. Does the sequence of activities required to complete a task follow what the user would expect?	3	2	-	4	-		**3**	
9. Does the system work in the way the user thinks it should work?	4	2	3	3	4		**3.2**	
10. Does the system support the evaluator so that the probability of conducting errors is minimized?	3	3	2	3	3		**2.8**	
11. Does the system support the evaluator in carrying out tasks correctly and efficiently?	4	3	3	4	3		**3.4**	

1 = many problems 2 = minor problems 3 = no problem

Question	**1**	**2**	**3**	**4**	**5**	**6**	**M**	**Comments**
Section 7: Usability problems					**2.8**			
1. Working out how to use the system	3	2	3	4	3		**3**	
2. Understanding how to carry out the tasks	3	1	3	4	3		**2.8**	
4. Finding the information you want	2	1	2	4	3		**2.4**	
5. Too many colours on the screen	3	2	3	4	3		**3**	
6. An inflexible, rigid system structure	3	2	3	4	3		**3**	
10. Unexpected actions by the system	3	2	3	3	3		**2.8**	
11. An input device which is difficult or awkward to use	3	3	3	4	3		**3.2**	
8. Having to remember too much information while carrying out a task	2	1	3	4	1		**2.2**	

1 = Very unsatisfactory 2 = fairly unsatisfactory 3 = neutral
4 = fairly satisfactory 5 = very satisfactory

Questions	**1**	**2**	**3**	**4**	**5**	**6**	**M**	**Comments**
Section 1: Clarity of the representation								
How do you assess the system??	4	4	-	5	4		**4.3**	
Section 2: Functionality								
How do you appraise the system in terms of functionality	5	4	-	5	4		**4.5**	
Section 3: Alarms and error messages								
How do you estimate the system in terms of alarm and messages presentation?	4	3	-	5	4		**4**	
Section 4: Feedback								
How do you assess the system in terms of feedback	4	2	-	5	-		**3.7**	
Section 5: Consistency								
How do you assess the system in terms of consistency?	4	3	-	5	-		**4**	
Section 6: Compatibility								
Which is the level of compatibility of the system?	5	4	-	5	4		**4.5**	
Section 7: Usability problems								
How do you evaluate the system usability?	-	-	-	-	-		**-**	

Average = 4.2

General questions

Questions	**Replies**
Which are the best aspects of the system?	The capability to exploit multimediality to display information
	Use of vocal commands can be of some help.
	The possibility to focus on the solution of a problem without worrying to look for the specific information. The capability to interact simultaneously via manual and vocal input The optimal spatial reorganization of the display
	Graphic interface and autoadaptivity Alarm priority pointed out by colours
Which are the worst aspects of the system?	Difficult to say without an actual AMEBICA console
	Iconic presentation of the alarms might be useful (instead of alarm lists)
	Maybe the excessive level of intrusion of the system
	Colour assignment might be changed
Which improvement do you suggest?	Multimedia features might be confusing for some operators
	Some operators don't want an excessive reduction of information. Experienced operator may wish to have al the information at a glance (especially in some instances: relay protection for instance).
	Autoadaptivity should be customisable for each operator

What do you like to add?	All functionalities are enough covered
	Include a Context sensitive HELP function.
	capability to export information to field maintenance crews

ENEL Workshop in Genoa

Results from the Questionnaires

Assessment

Question	1		2		3		4		5		6		M	S/N		
	#	S/N	#	S/N	#	S/N	#	S/N	#	S/N	#	S/N	#	S	N	X
Announcements and notification													**3.8**			
Recognize alarms	4	S	4	S	4	S							**4**	3		
Aware of the storm	3	N	4	S	3	N							**3.3**	1	2	
Vocal input	4	S	4	S	4	S							**4**	3		
Vocal output	4	S	4	S	4	S							**4**	3		
Critical Task Completion													**4.2**			
Track the storm movement	3	N	4	S	3	N							**3.3**	1	2	
Focus on critical alarms	5	S	5	S	5	S							**5**	3		
Establish priorities	4	N	5	S	5	S							**4.6**	2	1	
Verbal command recognition	3	N	5	S	3	N							**3.6**	1	2	
Stabilisation													**3.3**			
Stabilise the crisis situation	3	N	4	S	3	N							**3.3**	1	2	
Restore to a normal state	3	N	4	S	3	N							**3.3**	1	2	

INTERVIEW

Question	1	2	3	4	5	6	M	Comments
Part 1						**5.8**		
Display of information						**6.0**		
Which is your overal impression of the system? (1)	6	6	6				**6**	
Do you think the system provides information about the process state in a good way? (2)	7	6	6				**6.3**	
Does too much information exist in the display? (2)	4	6	5				**5**	
Is the amount of information sufficient? (2)	6	6	6				**6**	
Is the most important information easy to find? (3)	5	6	5				**5.3**	
Do you find the way of presenting the information logical? (3)	6	6	6				**6**	
Is information presented in a logical arrangement? (3)	7	7	7				**7**	
Effectiveness 5.2								
Is it possible to carry out tasks easy and logically? (3)	4	6	5				**5**	
Are you able to carry out your tasks efficiently when using the system? (4)	6	6	6				**6**	
Is it easy to prevent errors? (5)	4	6	4				**4.7**	
Does the system provide you with a good overview of the process? (6)	6	6	6				**6**	
Do you find the information you need to work in the system? (3)	7	6	6				**6.3**	
Is the amount of	6	6	6				**6**	

Question	1	2	3	4	5	6	M	Comments
information sufficient? (3)								
Subjectively pleasing					**5.9**			
Do you like working with the system? (6)	5	6	5				**5.3**	
Do you like the way information is presented? (6)	6	6	6				**6**	
Do you find the system pleasing to interact with? (6)	6	6	6				**6**	
Do you feel in control of the system? (6)	6	6	6				**6**	
Is the information presented supporting your main tasks? (3)								
How easy is it to learn to work with the system? (3)	6	6	6				**6**	

Question	1	2	3	4	5	6	M	Comments
Part 2						**5.6**		
Alarms						**5.6**		
How easy is it to diagnose faults by using alarms? (3)	7	6	6				**6.3**	
Is the alarm system making you attentive when deviations occurs in the system? (3)	7	6	6				**6.3**	
Does the alarm system provide sufficient information on the priority and cause of the deviation? (3)	4	4	5				**4.3**	
Are the feedback you get from the alarms helping you to decide if you carried out the right action? (3)	6	5	5				**5.3**	
Are the alarms presented consistent with other information on the screens? (7)	5	5	6				**5.3**	
How do you evaluate presentation of alarms? (3)	6	5	6				**5.7**	
Is the alarm text understandable? (3)	6	5	4				**5**	
Is vocal input/output helpful? (3)	5	6	4				**5**	
Do you believe that multimedia are a good support fro the network control? (3)	7	7	6				**6.7**	

(1) Very Bad – very good (2) False – True (3) No – Yes (4) Very little– A lot (5) Very difficult – very easy (6) Little – Much (7) Bad – good

The evaluation checklist

1 = never 2 = some times 3 = often 4 = always

Question	1	2	3	4	5	6	M	Comments
Section 1: Clarity of the representation					**3.2**			
1. Is important information highlighted on the screen?	3	4	3				**3.3**	
2. Does information appear to be organized logically on the screen?	3	4	3				**3.3**	
3. Are bright or light colours displayed on a dark background and vice versa?	3	4	3				**3.3**	subjective
4. Does the use of colour help to make the displays clear?	4	4	4				**4**	
5. Where colour is used, will all aspects of the display be easy to see if used on a monochrome or low-resolution screen or if the user is colour blind?	3	3	3				**3**	
6. Is the information on the screen easy to see and read?	3	3	3				**3**	
9. Is it easy to find the required information on a screen?	3	3	3				**3**	
10. Are voice messages useful and clear?	2	4	2				**2.7**	
12. It is easy to interact with the system using vocal command?								
11. Do vocal commands help in fulfilling the task?	2	4	3				**3**	
Section 2 Functionality					**2.7**			
1. Is the way in which information is presented appropriate for the tasks?	3	4	3				**3.3**	
2. Does each screen contain all the information that the evaluator feels is relevant to the task?	2	4	2				**2.7**	Often to much
5. Is system feedback appropriate for the task?	3	3	3				**3**	
6. Does the system help the evaluator to understand the state of the process?	2	3	2				**2.3**	
8. Does the nature of adaptation ensure that the evaluator is warned of a process disturbance in time?	3	4	2				**3**	
9. Are the adaptation strategies consequent?	2	3	3				**2.7**	Integration with ENEL procedures required
10. Is the media chosen for adaptation appropriate?	2	4	3				**3**	
11.Are the voice media effective?	2	3	2				**2.3**	
11. Does the system decrease the users' workload?	2	-	3				**2.5**	Difficult, but it would improve efficiency
14. Does the system ensure that important information is presented for the evaluator at appropriate times?								

Question	**1**	**2**	**3**	**4**	**5**	**6**	**M**	**Comments**
13. Is vocal information appropriate?	2	3	2				**2.3**	to be improved, but in compliance with ENEL's requirements
Section 3: Alarms and error messages					**3.4**			
1. Does the system clearly warn the evaluator about a disturbance or a deviation from a normal situation?	4	4	3				**3.7**	
2. Is the alarm of such a importance that it interrupts the evaluator from its work?	4	-	3				**3.5**	
3. Is a normal situation indicated by the absence of colours and sounds that demand attention?	4	4	4				**4**	
4. Is the system designed so that alarms that demand immediate attention are presented in a way that they are perceived and understood in all situations, including situations with a high degree of disturbance?	3	3	3				**3**	
3. Is a normal situation indicated by the absence of colours and sounds that demand attention?	2	4	3				**3**	
9. Are voice alarms helpful to call for operator attention?								
6. Can audible alarms be heard despite background-sound?								
7. Does the system provide the possibility of turning an alarm sound off from several places in the control room?								
5. In the alarm display, is there enough space to show all high priority alarms at the same time?								
8. Are voice commands helpful in carrying out parallel activities	2	4	4				**3.3**	Only in conjunction with other methods
Section 4: Information Feedback					**2.7**			
1. Are instructions and messages displayed by the system concise and positive?	-	4	3				**3.5**	
2. Are messages displayed by the system relevant?	2	3	2				**2.3**	
3. Do instructions and prompts clearly indicate what to do?	3	3	2				**2.7**	
4. Is it clear what actions the user can take at any stage?	4	3	3				**3.3**	
5. Are status messages informative?	3	3	2				**2.7**	

Question	1	2	3	4	5	6	M	Comments
6. Does the system clearly inform the user when it completes a requested action?	3	3	3				**3**	
7. Does the system promptly inform the user of any delay, making it clear that the user's input or request is being processed?	2	3	3				**2.7**	
9. Is it clear to the user what should be done to correct an error?								
10. Do alarm messages inform the evaluator about the priority and nature of the deviation?								
11. Do alarm messages guide the evaluators' initial actions?								
12. Do alarm messages verify whether the evaluator's response corrected the deviation?	2	3	1				**2**	
Section 5: Consistency					**3**			
1. Are the different colours used consistently throughout the system?	3	3	3				**3**	
3. Are icons, symbols, graphical representations and other pictorial information used consistently throughout the system?	3	3	3				**3**	
4. Is the same type of information displayed In the same location on the screen? b) In the same layout?	2	3	3				**2.7**	
5. Is the same item of information displayed in the same format, wherever it appears?								
6. Is the format in which the user should enter particular types of information on the screen consistent throughout the system?								
8. Is the method of selecting options consistent throughout the system?								
Section 6: Compatibility					**2.8**			
1. Are colours assigned according to conventional associations where these are important? (e.g. red = alarm, stop)	3	3	3				**3**	
2. When abbreviations, acronyms, codes and other alphanumeric information are displayed, are they easy to recognize and understand?	3	3	2				**2.7**	With the manual On Line HELP would be useful
7. Does the organization and structure of the system fit the user's perception of the task?	3	3	3				**3**	
8. Does the sequence of activities required to complete a task follow what the user would expect?	3	2	3				**2.7**	Integration with ENEL required
9. Does the system work in the way the user thinks it should work?	3	3	2				**2.7**	
10. Does the system support the evaluator so that the probability of conducting errors is minimized?	3	3	2				**2.7**	
11. Does the system support the evaluator in carrying out tasks correctly and efficiently?	3	3	2				**2.7**	

1 = many problems 2 = minor problems 3 = no problem

Question	**1**	**2**	**3**	**4**	**5**	**6**	**M**	**Comments**
Section 7: Usability problems								**3**
1. Working out how to use the system	3	3	3				**3**	
2. Understanding how to carry out the tasks	3	3	3				**3**	
4. Finding the information you want	4	3	3				**3.3**	
5. Too many colours on the screen	3	3	3				**3**	
6. An inflexible, rigid system structure	3	3	3				**3**	
10. Unexpected actions by the system	2	3	3				**2.7**	
11. An input device which is difficult or awkward to use	4	3	3				**3.3**	
8. Having to remember too much information while carrying out a task	3	3	3				**3**	

1 = Very unsatisfactory **2 = fairly unsatisfactory** **3 = neutral**
4 = fairly satisfactory **5 = very satisfactory**

Questions	**1**	**2**	**3**	**4**	**5**	**6**	**M**	**Comments**
Section 1: Clarity of the representation								
How do you assess the system??	4	4	4				**4**	
Section 2: Functionality								
How do you appraise the system in terms of functionality	3	4	4				**3.7**	
Section 3: Alarms and errors message								
How do you estimate the system in terms of alarm and messages presentation?	4	4	4				**4**	
Section 4: Feedback								
How do you assess the system in terms of feedback	4	4	3				**3.7**	
Section 5: Consistency								
How do you assess the system in terms of consistency?	4	4	4				**4**	
Section 6: Compatibility								
Which is the level of compatibility of the system?	4	4	4				**4**	
Section 7: Usability problems								
How do you evaluate the system usability?	-	-	-	-	-		**-**	

Average = 3.9

General questions

Question	**Replies**
Which are the best aspects of the system?	Immediacy of alarms
	Multimediality as complement to autoadaptivity to support operator
	Operator guide
Which are the worst aspects of the system?	No tool to include different procedure (on line)???
	Too much information in relation to the workload
	Sometime difficult to understand
Which improvement do you suggest?	Selective filtering of information
	Try to simplify the operating procedures Reduce effects Operator customisable
	Improved readability
What do you like to add?	On line configuration of the AT network
	Possibility to trace the execution of standard ENEL command procedures (for example load transfer)