



Information Technology for European Advancement

A common reference model for adaptive user interfaces

Application domains:
avionics, home systems and vetronics

Deliverable D3

Public version

User Centred Intelligence: BEYOND (the GUI)

Partners: Philips, BARCO, Delft University of Technology and
Limburg University Centre

*This document is part of the work of the EUREKA 2023 - ITEA 99002 project BEYOND.
Copyright © 1999-2001 BEYOND consortium.*

*Published via the Beyond Website:
www.extra.research.philips.com/euprojects/beyond*

No part of this publication may be reproduced, stored or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without written permission of the BEYOND consortium.

Requests for publications can be send to: al_publicrequest@natlab.research.philips.com.

Contents:

1. Introduction
2. A generic framework for user interface adaptivity
3. Implementation architecture: software agents
4. The avionics domain
5. The home systems domain
6. The vetronics domain
7. Conclusions
8. References

{blank page}

1. INTRODUCTION

User interface adaptivity is the capability that an interface between a user and a software system has to adapt itself dynamically to the current situation of use. The current situation represents all the aspects which affect and are involved in the user-system interaction such as the user's goals and tasks, the internal structure of the tasks, the inter-dependencies between the tasks and the environment, the current state of the environment and so on.

The adaptivity of the user interface provides a means of simplifying the interaction between the user and the system by reducing the amount of information that has to be processed by the user as well as the interactions required to achieve a certain goal. One way in which this cognitive load on the user can be reduced is by using knowledge about the current situation in order to infer the information and functionality most needed by the user at any given time.

In order to model the current situation the system needs to gather knowledge about the users' explicit and implicit intents and their preferences. Moreover the system has to know about the context of use and the current state of the environment. In order to infer the information and functionality needed by the user at a given time, the system has to perform a matching procedure together with conflict resolution strategies for prioritising tasks. Then the adaptive interface has to provide the user with the right information at the right time and in the right format.

The objective of this Work Package is to provide an insight into the technology which is required in order to achieve user interface adaptivity in different application domains. The approach taken is to develop a generic framework for an adaptive human-machine interface, where the modelling of the current situation and the decision of what and how information is to be presented to the user are taken care by system components and their sub-components. The generic framework is then applied to different domains.

This document is organised as follows. The common reference model is presented and discussed in the next section. Next, software agents are discussed as well as possible applications of agents given the proposed architecture of the common reference model. In the sections which follow, domain specific applications are discussed. Three different domains, namely avionics, home systems and vetronics are discussed in the context of adaptivity and the generic framework is applied to each one of them. Each domain develops its own agent architecture, which is compatible to and useful in the domain specific application, and conformance and non-conformance to the common reference model are discussed. Finally, a section with conclusions summarises the concepts presented in the document.

In addition to observing user actions, the system keeps track of the active tasks in the system and their states in order to form a dynamic picture of the user-system interaction. Input obtained from the user is of an implicit nature and therefore the Inference Engine must cope with uncertainty. From that dynamic picture and through the Inference Engine, the system infers and evaluates the goals and intentions of the user - creating in this way a user "representation" - in order to establish the System Goals. They represent the current desired underlying system functionality as inferred by the reasoning engine from user's actions.

Information matching

This component integrates the Systems Goals and the Environment Model output through the Information Manager, which matches the system's goals - as inferred by the Inference Engine - against the present situation or context - as represented in the Environment Model. When doing so, the Information Manager may have to resolve possible conflicts, prioritise actions and tasks and take decisions as, e.g., constraints coming from the Environment Model might interfere with the current system goals of the System Goals module. It must also format the information to be presented to the user according to the current situation. It does so by observing priorities, user preferences, user goals and tasks and the environment state. The Information Manager then outputs the system's integrated knowledge, along with instructions and recommendations on information prioritisation and formatting, to the Dialogue Manager for presentation to the user.

User-system interaction

This component comprises the Physical Interface and Dialogue Manager modules. It covers all aspects of direct interaction between the user and the system. Such direct interaction takes place, from the user's point of view, through inputs to and outputs from the Physical Interface. In order to make this interaction natural and easy, the Physical Interface should have multi-modal input and output properties. Therefore, the present Work Package links very strongly with Work Package 1, which explores the aspect of multi-modality in user interface adaptivity. Once information from the user has passed through the Physical Interface or before information goes through the Physical Interface to be presented to the user, it goes through a deeper layer closer to the system, the Dialogue Manager. Its task is to administrate the information flow between the user and the system so that the user-system dialogue is coherent. On one side, a specific user input must be certified to be passed to the appropriate component. On the other side, the user must be presented with the appropriate information according to the current functionality being accessed at any one moment.

In the section 3, software agents will be discussed. A more detailed specification of the system components described above - in terms of sub-tasks organised as multiple agent systems - will be also discussed.

2.2. The information flow through the Dialogue Manager

The information which passes through the Dialogue Manager is of a very diverse nature as discussed below.

The Environment Knowledge represents the meaningful information or knowledge which flows directly between the Dialogue Manager and the Environment Input units.

It can be either knowledge from the units that is directly passed to the Dialogue Manager to be presented to the user - such as (possibly pre-processed) media streams and measurements of the environment and the underlying system properties coming from sensors - or knowledge coming directly from the user which might be used as input to some of the units - direct user input such as user's instructions, preferences, capabilities and explicit communication of goals and intentions.

The Environment Model can be communicated directly to the user through the Model Dialogue, which allows the user to modify or overrule that system's mental model. The System Goals, as established by the system, must also be made available to the user. They can be communicated to the user through the Goal Dialogue, which allows the user to modify or overrule the inferred goals.

In addition to the bilateral modes described above, information between the adaptive system and the user can also flow in an unilateral manner, by means of the Inference Engine and the Information Manager modules.

The information that flows via the Inference Engine is from the user to the system, in an implicit fashion. There, the user's actions considered in the context of the tasks' states are observed by the system and this knowledge is used by the Inference Engine to infer the user's intentions and establish the System Goals.

The information that flows via the Information Manager goes from the system to the user and represents the inferred information and functionality currently most needed by the user (the right information presented at the right time). This information must be also presented in the right format according to the current situation and context of use.

3. IMPLEMENTATION ARCHITECTURE: SOFTWARE AGENT TECHNOLOGY

3.1. Software agents

A common procedure in modelling complex software systems involves the decomposition of the system into smaller parts which are then organised through established relations between them. This standard technique in modelling complex systems includes, for instance, the object oriented approach in software development [Meyer, 1997]. Another approach, component-based software development [Szypersky, 1997], is currently getting a lot of attention, especially for distributed or networked systems. They both aim at achieving “pluggability”, which is the ability to reconfigure objects or components in order to cope with any variation that may be desired in the system. As software systems become more complex and distributed, new paradigms are required in order to model and understand those systems. The object oriented and component-based approaches can be extended by adding on top of them an *agent* layer.

An agent can be defined as a software entity which functions continuously and autonomously in a particular environment, often inhabited by other agents and processes. Agents engage in high level interactions to achieve their goals and there is an organisational relationship between them [Shoham, 1997; Jennings, 1999]. With the agent oriented approach, the complex system can be conceptualised as a society of co-operating and autonomous problem solvers. Agents therefore provide a more natural, more human-like way of modelling, designing and implementing complex, distributed and knowledge-intensive systems. This approach results in systems that can be better understood and, therefore, interact more naturally and easily with the users.

For the reasons mentioned above software agents come immediately to mind when devising a common architecture for the generic framework for user interface adaptivity, as described in the previous sections. However, the immediate association has more to do with the foreseen results of the agent implementation (systems are better understood and can better interact with the user) than with the implementation itself, which might not be so straightforward when examining the proposed model presented in the previous section.

The reason for that uncertainty is that agents in the strict sense of the term must have attributes such as internal mental states, beliefs, goals, intentions and plans. With respect to the interaction with other agents, they must be able to pass high level information. Moreover, the interactions are not necessarily defined at design time and can change at run time (flexibility). Therefore an agent can search to find other agents that it needs in order to achieve its goal (autonomy, pro-activity), even if those agents did not yet exist when the agent was created [Bourne, Excelente-Toledo and Jennings, 2000].

In order for the proposed generic framework to support a common agent architecture - given the above specifications - the various components and modules in the model would have to possess a similar level of complexity across the different application domains. In this case generic agents could be, for instance, associated with those

components. However this is not the case, as the discussion on domain-specific agent implementations presented in the following sections will show. Therefore, only some very general remarks about possible agent assignments to system components will be made in this section.

3.2. Agents in the common reference model

As discussed in the previous section, the common reference model encompasses four main components - environment perception, goal inferring, information matching and user-system interaction - each of which can be decomposed into modules. Those components and modules must work in an integrated and co-ordinated manner so as to provide the adaptivity functionality to the system.

The proposed approach is to regard the system as a multi-agent system (MAS) where agents incorporate the reasoning elements of the system and each agent is responsible for a very specific task. Each agent acts autonomously but communicates and co-operates with other agents and the user. Functions which do not involve reasoning are assigned to objects, according to an object-oriented approach. These objects might still be generically called "agents", but it must be kept in mind that they might not conform to the strict definition of agents as discussed previously. They will be referred to as "lower-level" agents. Therefore the integration and co-ordination in the system can be achieved through agent communication and co-operation in the higher and lower levels.

Environment perception

This component, which is responsible for establishing and maintaining situation awareness through the Environment Model, might involve reasoning - which would take place in the Environment Input units - although reasoning might be altogether absent or present in a simpler form. Therefore an agent might be assigned to each unit which performs reasoning. Moreover, low-level agents might perform simpler tasks, such as pre-processing or integration of information before that information goes into the Environment Model. The maintenance of the model itself might be assigned to an agent which communicates to the user through the Model Dialogue.

Goal inferring

This component, responsible for maintaining knowledge about the current system goals, can be implemented as a multi-agent component. An agent might be assigned which will reason about the observed user's actions over time. Another agent might keep track of the active tasks in the system and their states in order to form a dynamic picture of the user-system interaction. Through communication, those agents can infer and evaluate the user's intentions and establish the current system goals based on the inferred information. Moreover, maintenance of knowledge about the current system goals and communication to the user through the Goals Dialogue might be assigned to another agent.

Information matching

This component, which is responsible for the main integration of information in the system, promptly invokes the multiple agent approach. Independent tasks are performed which must be intelligently integrated. The task of matching the system's goals against the present situation or context may be very complex and can be

performed by one agent or, perhaps, multiple agents using different matching procedures. When performing the matching, those agents may have to perform sub-tasks such as to resolve possible conflicts, prioritise actions and tasks, select and optimise actions and take decisions. These can be achieved through agent communication and co-operation. The management of the information which is to be presented to the user can be also assigned to multiple agents. They must prioritise the information presentation, format the information according to the current situation (priorities, user preferences, user goals and tasks, the environment state, etc) and map the information to the right medium.

User-system interaction

This component, responsible for the direct interaction between the user and the system, can also be implemented through interface agents. Interface agents such as anthropomorphic animated characters may inhabit the Physical Interface and interact directly with the user. As mentioned previously, multi-modal properties of such an interface are essential in order to fully explore the natural interaction paradigm provided by agents. The task of the Dialogue Manager of administrating the information flow between the user and the system may be implemented with lower-level organising agents. They communicate with the agents of the Information Manager module which are responsible for the prioritisation and formatting of the information and, through their recommendations and instructions, they can perform their task.

In the following sections, domain specific applications will be discussed in the context of user interface adaptivity. The generic framework will be applied to each one of the domains, by means of specialisation of its components and modules, and domain-specific agent implementations will be discussed.

4. THE AVIONICS DOMAIN

4.1. Introduction

During the last two decades, more and more automation was introduced in the aircraft cockpit to cope with the increasing complexity of traffic situations and aircraft systems. In spite of these automation efforts, overall aviation safety did not increase. Accident analysis shows that about 65% of the aircraft incidents are attributed to human error [RAND, 1993]. While the automation was intended to eliminate human errors, it seems that often the automation itself is the underlying cause of the human error.

Conventional automation has taken over some tasks of the flight crew, placing the human pilot in the position of being a monitor of automation, a task for which humans are not particularly well suited [Wiener et al., 1980]. In low workload situations, the pilot might get bored, while in the event of a time-critical situation, pilots often do not have the resources to direct and monitor the automation, making the automation useless when the operator needs it the most. In addition, at the present state of the art, automation systems can only guarantee the correct reaction to a situation for which they have been explicitly designed and which they have learned. In the event of an unusual situation, for which the system was not designed, it is the pilot that has to take over and control the situation.

Another cause of human error is a lack of situation awareness. According to [Endsley, 1995], 'situation awareness is the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future'. Situation awareness is a necessity for good decision making. In order to form an internal representation of the flight situation, currently the crew has to integrate a large amount of information. Flight, environment and system state information are presented to them on numerous head-down cockpit displays in different formats. In addition, the auditory channel is used to alert the crew in case of a hazardous situation. With the increase of cockpit complexity and automation, pilots have difficulties to obtain situation awareness, especially during critical situations when this awareness is of greatest importance. In the future Air Traffic Management system, the airspace will be much more crowded, traffic situations will become even more complex, and the pilot will need an even better situation awareness, as the traffic will no longer be channelled along fixed routes and the pilot will be responsible for defining a safe and most efficient trajectory.

Therefore, in the international aviation society there is an urgent call for a new kind of cockpit automation. Recent technological evolutions provide a great potential for cognitive automation, intelligently controlling the information presentation, warning systems, and a dynamic task allocation. It is expected that an intelligent flight deck that works alongside the human operator can significantly increase the performance of the total human-machine system in terms of safety and efficiency.

4.2. The need for adaptivity

In the introduction, it was indicated that human errors are made because of:

- Limitations of conventional automation
- Information overload
- Too high or too low workload

The intelligent flight deck has to work alongside the human pilot in order to avoid human errors. Therefore, the following functionalities are needed:

- Information management, including information generation
- Error detection and correction, including intent inferencing
- Dynamic task allocation between the human operator and the machine
- Optimisation

Information management

The intelligent flight deck should present the right information in the right format at the right time to help the pilot in building the situation awareness they need for that particular situation. Based on knowledge about the flight situation, environment, pilot behaviour, goals, plans, etc., the automation first filters out the relevant information. Then, it determines how long the information should be presented and in what order. Finally, the presentation mode or modality is chosen. In addition to information management, the intelligent flight deck will generate information, such as warnings, judgements about the situation, alternatives, and recommended responses.

Error detection and correction

To be able to detect errors, the system first has to find out what the pilot's intentions are (by explicit communication or (implicit) intent inference). These intentions can be compared with the actual crew behaviour. Thereby, discrepancies in behaviour and their reasons have to be determined. In case of deviation from the expected behaviour, the system has to discriminate between possible unknown changes of pilot intents and erroneous behaviour. When an error is detected, the system gives a warning to the crew and suggests a correction. In the event of a change of intent, the system tries to figure out what the new intention might be.

Workload management

Based on the intended and actual crew behaviour, a model determines the actual and predicted tasks. This can be used to estimate the current and future pilot workload and to identify possible pilot under-load or overload. This task loading information can be used by the intelligent flight deck to dynamically allocate tasks between the human operator and the automatic systems.

Optimisation

The human pilot is not a good optimiser. In comparison with the intelligent flight deck the human pilot has limited overview of the situation. If the human pilot expresses an intent, the intelligent flight deck can propose optimised alternatives.

4.3. Applicability of the common reference model

The common reference model, as presented in Fig. 1, is very representative for the model of the intelligent flight deck that is discussed in [van Paassen et al, 2000].

Some comments starting at the bottom of the figure:

- The Environment Input units:

They are Functionalities in the Avionics domain, implemented as agents in the intelligent flight deck. These function-specific agents provide the interaction with the environment, the aircraft and the pilots. They receive raw data from sensors or already interpreted information from other agents on a lower level in the multi-agent architecture. The output of the functionalities consists of information with a meaning. They supply the content of the information (Environment Knowledge) directly to the Dialogue Manager. Every functionality covers a different aspect of the environment. For example, there is one functionality for surveying surrounding traffic, one for guarding ground proximity, one for aircraft system failure, and so on.

- **The Environment Model:**
It includes the human operator or the piloting crew.
- **The Model Dialogue:**
It allows the communication of the Environment Model to the user and modification of the Environment Model by the user.
- **The Information Manager:**
It combines information about the system goals with the knowledge of the system and the environment.
- **The System Goals module:**
It maintains knowledge about the current system goals.
- **The Goal Dialogue:**
It is the explicit communication about system goals between the pilot and the intelligent flight deck.
- **The Inference Engine:**
It infers pilot intentions without any explicit communication about those intentions.
- **The Dialogue Manager:**
It decides how to use the information channels between the pilots and the intelligent flight deck. Content of the information is supplied by the different functionalities. Selection of format and priority of information is done on the basis of recommendations by the Information Manager.

4.4. The avionics domain model

The model for the intelligent flight deck is shown below.

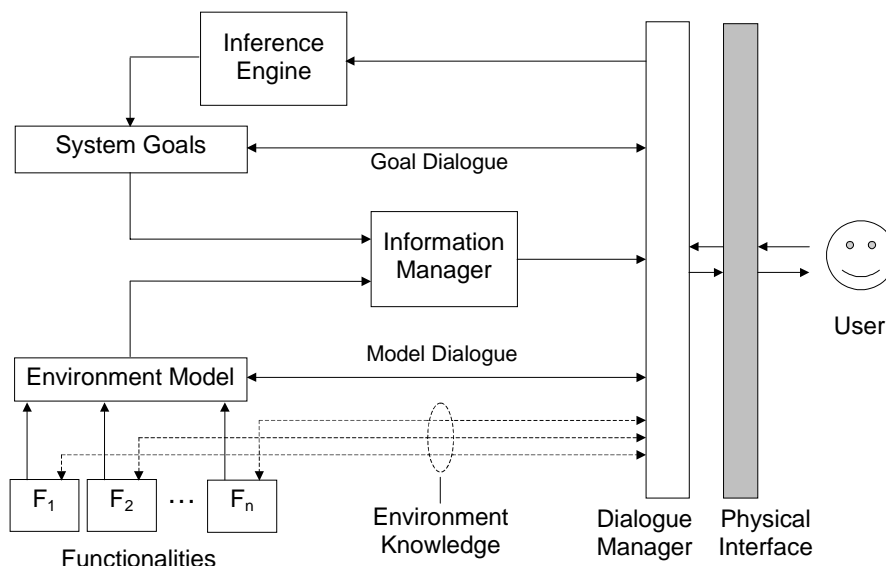


Fig. 2. Architecture for adaptivity in the intelligent flight deck.

4.5. Implementation with an agent architecture

The current aircraft systems function largely independent from each other, and all of these systems have their individual channels to communicate with the pilot. To come to a more integrated, intelligent avionics system, a new avionics architecture is needed. It is believed that this can be achieved by an avionics system in the form of a multi-agent system [Mulder et al., 2000]. The technology of co-operating intelligent agents [Wooldridge and Jennings, 1995] allows the development of a modular system, where software agents each handle a specific function in the system. Each agent is a relatively simple, but highly specialised entity. The agents can reason about themselves, their goals, their (perceived) environment, other agents, etc. They communicate their advice, needs, goals, etc. to each other, and jointly communicate with the human operator [van Paassen et al., 2000]. It is this co-operating behaviour that results in system intelligence.

This leads to a system as depicted in the figure below.

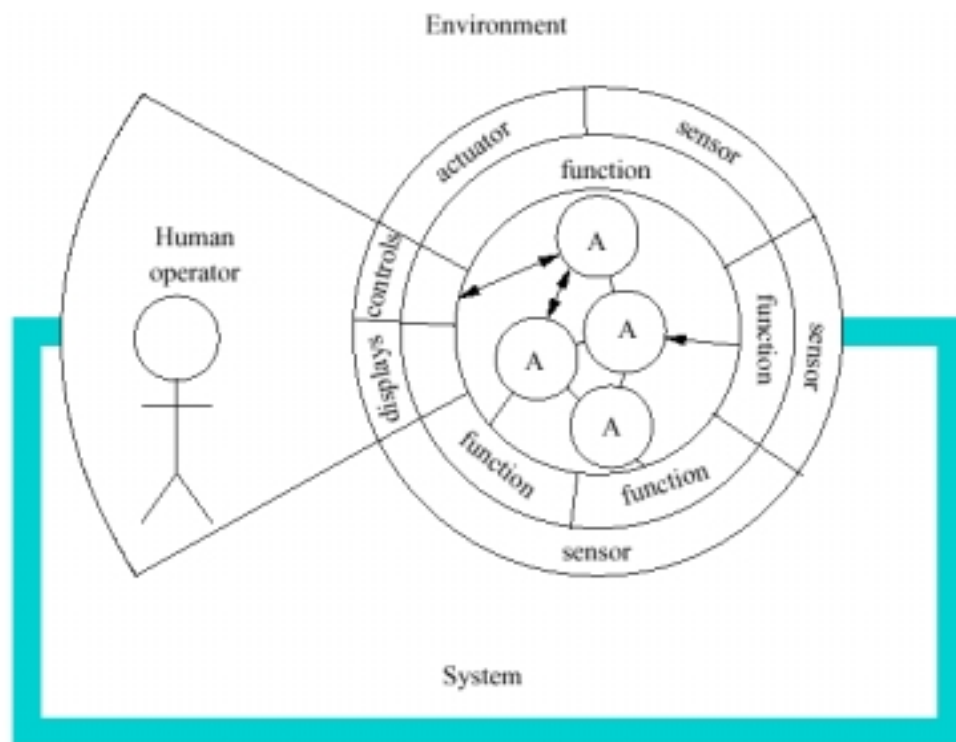


Fig. 3. Human operator supported by intelligent agents (A) in the control of a system. Through actuators/sensors and monitoring and automation functions the agents observe and influence the environment and the system.

In this multi-agent system, the following can be distinguished [van Paassen et al., 2000]:

- Sensors (outer layer of the circle) gather and measure signals from the environment of the aircraft, or from the aircraft itself.

- Monitoring and automation functions (in the middle layer of the circle) use the signals and interpret these to detect meaningful events.
- Each agent (in the inner circle) uses one or more functions to interpret the meaningful events in the context of the goals of the aircraft/pilot/automation system.
- The pilot(s), on the left side of the diagram, communicate with the system through interfaces.

One can see that information, as it travels from the world external to the support system to its core, becomes more meaningful. Using the terminology of [Rasmussen, 1983], the outer layer handles signals, the middle layer converts these into signs, and the agents handle symbols. The modules of the adaptive system as presented in Fig. 2, except for the Physical Interface and the User, fit into the agent organisation in the inner circle of Fig. 3.

5. THE HOME SYSTEMS DOMAIN

Removed from the public version

6. THE VETRONICS DOMAIN

6.1. Introduction

As traffic becomes more dense and complicated, the cockpits of special-purpose vehicles - e.g., ships, trains, trams, airplanes and vehicles used in the construction industry - are continuously evolving with an increasing number of embedded electronics devices (vehicle electronics or *vetronics*). As a consequence, more and more mechanical navigation instruments are being replaced with multi-functional displays which can be more selective with the presentation of the available information.

In order to assist the vehicles' operators in their increasingly complicated task, automation levels have been growing. As a consequence, the operator assumes more and more a role of an observer of the system's behaviour and automation. It is a fact, however, that users do not like to lose control of the system. Therefore, a certain level of interaction between the human and the machine is still needed.

The vetronics display is a multi-function display that BARCO offers to system integrators - e.g., train builders - so that they are able to build the display into the system - e.g., a train. In this way, system users - customers of system integrators - can use the complete system. This means that at least three parties are involved:

- display manufacturer (e.g. BARCO)
- system integrator (e.g. Train builder)
- user (e.g. Railway company)

In current systems, the 'intelligence' is fixed into the system and cannot be easily modified without a service intervention. This has several disadvantages:

- As the intelligence is in the display itself, BARCO has to have knowledge of the specific application (e.g., a train).
- The system integrator cannot easily adjust the functionality to their wishes and specifications. They always need to contact the display vendor to modify that functionality.

BARCO wants to create a User Interface Editor to be used with its vetronics display range. The User Interface Editor allows the system integrators themselves to make their own user interface according to their specifications and wishes. This approach gives them the opportunity to modify the interface whenever there is need to do so. This gives the system integrator much more freedom and flexibility in the development of their system.

The desired functionality, as requested by the user of the system, can now be communicated to the system integrator, who, by means of the User Interface Editor, is able to modify and adjust the current system's functionality.

6.2. The need for adaptivity

As mentioned in the introduction, a multi-function display should be able to process a large amount of information. This means that some filtering mechanism must be

applied to the information in order to give the operator access to all relevant information while hiding the irrelevant data.

The relevance of the data depends on several factors, amongst which:

- the complete system state,
- the display date, and
- the information itself.

On the other hand, the filtering and presentation of relevant information must lead to:

- less workload,
- more safety, and
- higher efficiency.

Generally speaking, in order to generate a user interface by means of a tool such as a User interface Editor, we need to create a mechanism which associates the displayed image (that is, video plus information) to the supplied information. This means that what the operator sees at any one time is not fixed, but depends on the circumstances. Therefore some type of matching procedure must take place, which links the present information to the user interface functionality, which was previously established with the User Interface Editor.

The ability to use a separate tool to make the link between the present information and the image which the operator sees at any one time gives the complete avionics line an enormous added value compared to traditional displays. Provided that the tool be easy to use and compatible with standard computers, we can nearly talk about an adaptive system. However, since the display itself has no user knowledge but only task knowledge and adaptivity to user's specifications and wishes takes place before and in-between sessions of use of the system (through the use of the User Interface Editor), we can talk more appropriately about an off-line adaptive system.

In order to achieve more safety, all requirements can be implemented in the user interface at the time of its development. This can be done in a very flexible and modular manner. For verification purposes, we can simulate the complete user interface before it is loaded into the system.

6.3. Applicability of the common reference model

The model for off-line user interface adaptivity discussed above can very easily fit into the generic framework for user interface adaptivity, as presented in Section 2. When looking at the common reference model we find all components which are needed in our domain specific adaptivity model.

The function of the modules is as follows:

- **Environment Model:**
It gathers information about environmental conditions which are relevant to the vehicle operation as well as availability and constraints information - e.g., available communication ports and connected video types - in order to establish situation awareness.
- **System Goals:**

It represent the desired system functionality as established through off-line adaptivity by means of the User Interface Editor.

- **Inference Engine:**
It is absent in our model, since the vetronics system does not retain user knowledge, at least not for the time being.
- **Information Manager:**
It represents the built-in intelligence in the system, which links the present information about the environment to the user interface functionality, which was previously established with the User Interface Editor.
- **Dialogue Manager:**
It decides how to use the information channels between the vehicle’s operator and the system in order to present all relevant information.

6.4. The vetronics model

The model for the off-line adaptive vetronics system is shown below.

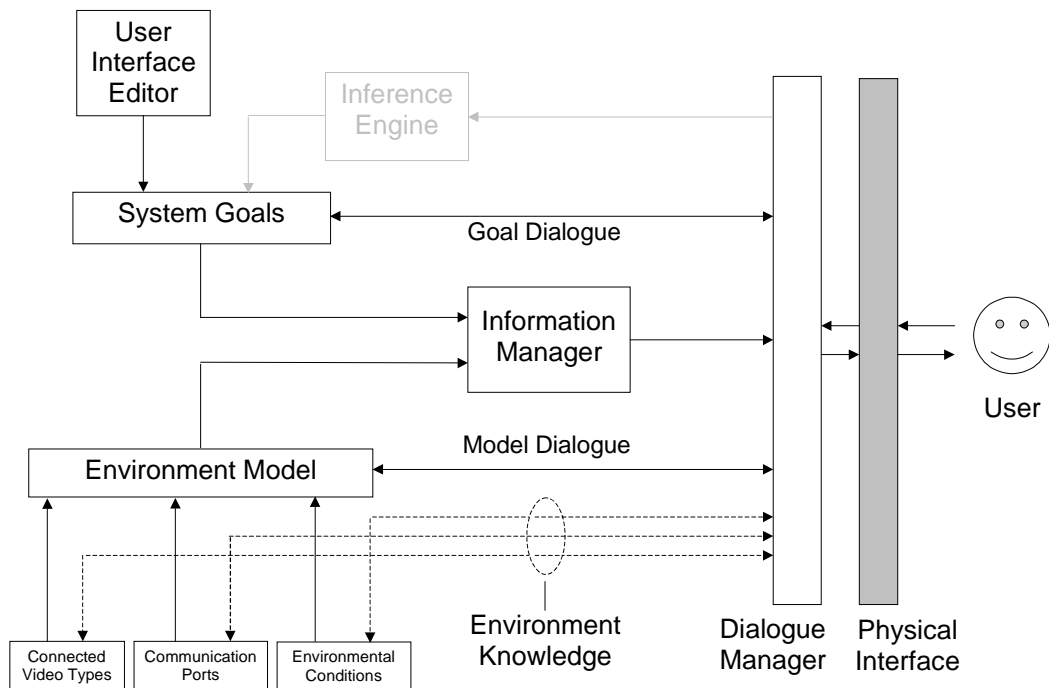


Fig.7. Architecture for off-line adaptivity in the vetronics domain.

6.5. Implementation with an agent architecture

As the main focus of the vetronics domain, in the context of the Beyond project, is off-line user interface adaptivity and flexibility, agent technology is not envisioned immediately. However, hardware system evolution will continue to produce vetronics systems with fewer constraints (e.g., constraints in memory and processing power) and, thus, on-line adaptivity supported by an agent architecture will be possible in the future.

Therefore, it is important even at this stage to map the off-line adaptivity model onto the common reference model, so that integration of agent technology will be accomplished based on a generic framework for user interface adaptivity.

7. CONCLUSIONS

A generic framework for an adaptive human-machine interface has been introduced and discussed in section 2. In this common reference model, the modelling of the current situation and the decision of what and how information is to be presented to the user are taken care by system components, each of which can be decomposed into modules. Components and modules must work in an integrated and co-ordinated manner so as to provide the adaptivity functionality to the system.

An agent architecture which can be supported by the common reference model was discussed in section 3. The approach adopted was to regard the system as a multi-agent system with higher and lower levels of agency. Each agent is responsible for a specific task and acts autonomously but communicates and co-operates with other agents and the user. The integration and co-ordination of the system's components and modules can be achieved through agent communication and co-operation.

Domain specific applications of user interface adaptivity were discussed in sections 4, 5 and 6 for the avionics, home systems and vetronics domains, respectively. The need for adaptivity was discussed as well as the applicability of the generic framework to each one of those domains. Next, the domains, each, proposed their own version of the common reference model which is compatible to and useful in their application. Finally, an implementation of their model with an agent architecture was discussed.

The domain specific applications share the global features of the common reference model. This is due to the need for adaptivity in all those domains which fundamentally requires the modelling of the current situation as a set of goals (what is wanted) and a set of constraints (what is possible), which must be matched against each other through an information matching mechanism. The latter, together with a mechanism responsible for the direct user-system dialogue, define what and how information is to be presented to the user.

The differentiation according to the specific domain of application happens at the level of the components and modules of the common reference model. As it can be seen from the domain specific models, the function, complexity degree and relative importance of components and modules as well as their very presence are generally domain dependent. In addition, new modules may be added to represent functions which are domain specific. Connecting lines and arrows, which indicate the flow of information between modules, behave accordingly.

The common reference model proposed in this document was successfully applied to three different domains. It is generic enough to encompass all the global features required in order to achieve user interface adaptivity, at the same time allowing specificity through its task-oriented components and modules. Therefore, the proposed model can be applied to any new domain in which user interface adaptivity is required.

8. REFERENCES

- Bourne, R.A., Excelente-Toledo, C.B. and Jennings, N.R. (2000). "Run-Time Selection of Co-ordination Mechanisms in Multi-Agent Systems", Proceedings of the 14th European Conference on Artificial Intelligence (ECAI-2000), Berlin, Germany (to appear).
- Endsley, M.R. (1995). "Toward a Theory of Situation Awareness in Dynamic Systems", *Human Factors*, 37(1), pp. 32-64.
- Jennings, N.R. (1999). "Agent-Oriented Software Engineering" Proceedings of the 12th International Conference on Industrial and Engineering Applications of AI, Cairo, Egypt, pp. 4-10 (Invited paper).
- Meyer, B. (1997). *Object-Oriented Software Construction*, Prentice Hall Professional Technical Reference.
- Mulder, M., Pedroso, P., Mulder, M. and van Paassen, M.M. (2000). "Integrating Aircraft Warning Systems", Proceedings of the 19th European Conference on Human Decision Making and Manual Control, Ispra (Italy), June 26-28, pp.162-166.
- van Paassen, M.M., Mulder, M. and Abeloos, A.L.M. (2000). "A Model for Cooperation between Humans and Intelligent Systems", Proceedings of the 19th European Conference on Human Decision Making and Manual Control, Ispra (Italy), June 26-28, pp.62-66.
- RAND (1993). *Airport Growth and Safety. A Study of the External Risks of Schiphol Airport and Possible Safety-Enhancement Measures*. Santa Monica (CA): RAND.
- Rasmussen, J. (1983). "Skills, rules and knowledge; signals, signs and symbols, and other distinctions in human performance models", *IEEE-SMC* 13(3), pp. 257-266.
- Shoham, Y. (1997). "Agent Oriented Programming: a Survey", in *Software Agents*, J. M. Bradshaw (ed.), MIT Press.
- Szypersky, C. (1997). *Component Software*, Addison-Wesley.
- Wiener, E.L. and Curry, R.E. (1980). "Flight-Deck Automation: Promises and Problems", *Ergonomics*, 23(10), pp. 995-1011.
- Wooldridge, M. and Jennings, N.R. (1995). "Intelligent Agents: Theory and Practice", *The Knowledge Engineering Review* 10 (2), pp.115-152.

