ITEA

Information Technology for European Advancement

Functional Specification and Architecture

Deliverable D8

Public version

User Centred Intelligence:

BEYOND (the GUI)

Edited by K. Coninx (LUC-EDM)

contents:

1. Management and summary

This deliverable D8, entitled “Functional Specification and Architecture”, has the intention of summarizing the progress in the BEYOND project and clarifying the approach in the process of going from the first to the second milestone.

As such D8 is a so-called common deliverable, beyond workpackage borders. There is no ambition to strive for completeness in this document. All BEYOND project members have been approached and partners themselves have made a selection of appropriate projects that could support the goal of this deliverable.

It was decided that a domain-oriented approach provides a comprehensive structure for this deliverable. It supports the placement of the individual projects in their context, and allows key aspects of the involved workpackages (multimodality, adaptivity, simulation and usability) to be emphasized.

The following contributions are integrated in this public version of deliverable D8 (there are two more contributions in the internal version):

Domain	**Project**	**Authors**
Public	Accesspoints; public information kiosks	Ferdinand Schinagl (APC Interactive Solutions AG)
Home	UI for future in-home electronic systems (Home Experience, Multimodal Jukebox)	Berry Eggen and Boris De Ruyter (Philips Research, USIT)
Home	UI editor for consumer domain	Linde Loomans (Philips Hasselt)
Vetronics	Vetronics UI editor	Johan Devos (BARCO BarcoView) Karin Coninx (LUC-EDM)
Avionics	Intelligent Adaptive Flight Deck	Max Mulder (TUD)

It has to be explicitly stated that due to uncertainties in the German funding situation, the consortium lacked the experienced partners for the usability workpackage. Therefore, the other partners had to find a way to overcome this problem. One consequence is that most of the contributing partners devoted resources to usability engineering activities, as reported in this deliverable. Also, reconsidering the purpose of this deliverable D8 and devoting some time to the usability issues has delayed this deliverable.

The next chapter elaborates on the exact purpose of this document and its relation to other deliverables. The contributions of the indivual project are ordered according to their application domain. Finally, project-wide conclusions concerning the evolution from the first to the second milestone are formulated.

2. Purpose and structure of the document

The purpose of this deliverable D8, entitled “Functional Specification and Architecture”, is twofold:

- summarizing the progress in the BEYOND project
- clarifying the approach in the process of going from the first to the second milestone.

The purpose and approach of D8 has been agreed upon during the team meeting in Delft (Nl), November 2000. The following arguments contributed to the definition of D8, as it is currently presented.

D8, “Functional Specification and Architecture”, can to some extent be considered as a second iteration on D2 "Requirements and Usability Methodology". The rationale behind this conclusion is as follows. After describing "key concepts" in deliverable D1, the BEYOND partners have contributed to D2, in which the requirements have been detailed and the usability methodology has been described. Most of the BEYOND partners planned to have a first prototype at the time of the first milestone (after one year of research in the consortium). This is in particular the case for the partners that are directly active in one of the application domains (Public domain, Home domain, Vetronics domain, Avionics domain). Consequently, the requirements stated in D2 are considered as a kind of checklist for the first prototype. Based on the lessons learnt from the first prototype, ideas for the second prototype have been collected. So we can consider the presented D8 as a description of the functional specifications for the second prototype to be delivered at milestone two (at the end of the BEYOND project, after two years of research in the consortium). It seems acceptable that the relation D2-milestone 1/prototype 1 is the same as D8-milestone 2/prototype 2.

Also it is in accordance with Software Engineering terminology that D8 speaks about Functional Specifications, which are usually closer to the design/architecture than the Requirements in D2, which were most of the time high-level requirements, not very detailed.

Besides this, it turns out that most of the BEYOND partners that are directly active in one of the application domains (Public domain, Home domain, Vetronics domain, Avionics domain) have a prototype/demonstrator that evolves from a version at milestone one to a final version (at least within the BEYOND project context) at milestone two. This as opposed to the realization of new prototypes during the second half of the project period.

As a result it was decided that a domain-oriented approach provides a comprehensive structure for this deliverable. It supports the placement of the individual projects in their context, and allows key aspects of the involved workpackages (multimodality, adaptivity, simulation and usability) to be emphasized. It is indeed important that *all* the key aspects of the workpackages (WP1: multimodality, WP2: adaptivity, WP3: simulation and WP4: usability) are present, because this document is a common deliverable for all consortium partners. However, we opted to let the workpackages take part through projects in application domains in order to obtain a coherent and comprehensive report. As several

partners are active within several workpackages this turned out to be a workable solution.

The contributions that are found in the following chapters are:

Public domain:
Accesspoints; public information kiosks

Home domain:
UI for future in-home electronic systems (Home Experience, Multimodal Jukebox)
UI editor for consumer domain

Vetronics domain:
Vetronics UI editor

Avionics domain:
Intelligent Adaptive Flight Deck

The order in which the domains and the individual projects within the domains are organized is rather deliberate: from the public domain with UIs for everyone in everyday circumstances, over the other application domains to the avionics domain, with UIs for very select users in a professional environment.

It should be emphasized (and it should be obvious after reading this deliverable) that the key aspects multimodality, adaptivity, simulation and usability are independent variables. They are different dimensions along which the UIs are investigated, but they have nothing to do with the order in which the projects are listed. The level to which these aspects have been addressed in the projects differs considerably. Therefore, for each of the projects listed, there will be separate sections devoted to the key aspects in order to (1) show the integrative level as an advantage of cooperation in the consortium and (2) to focuss on particular efforts concerning multimodality, adaptivity, simulation and usability. Also, there are sections on architectural issues.

It has to be explicitly stated that due to uncertainties in the German funding situation, the consortium lacked the experienced partners for a separate usability workpackage. Therefore, the other partners had to find a way to overcome this problem. One consequence is that most of the contributing partners devoted resources to usability engineering activities, as reported in this deliverable.

The above mentioned relation between deliverables D1 (key concepts), D2 (requirements) and D8 (functional specifications) is applicable for all the listed projects. D1, D2 and D8 are all common deliverables. Besides this, there are relations with other deliverables that are the result of particular workpackages. For instance, D3 “Common Adaptivity Reference Framework” from WP2 comes into play in the adaptivity and/or architectural sections of this document. This illustrates that the migration from the first to the second prototype takes into account adaptivity issues, possibly inspired by the cooperation in the context of the adaptivity framework. Similar relations can be found with regard to multimodality and simulation. It is likely that the relationship also extends to future deliverables, intended to provide even more detail when we progress towards milestone 2.

The next chapters list the individual projects and are integrated by the information provided by the project responsibles. Finally, project-wide conclusions concerning the evolution from the first to the second milestone are formulated.

3. Functional specifications and architecture for domain public (public terminals and kiosks)

3.1. Introduction

APC (apc interactive solutions AG) is working on the development of information networks, terminals and kiosk solutions to address people's needs for information in public space. The core component in this project is the information terminal which is called accesspoint. It resembles the interface between users and the network which transports and provides context sensitive information. The accesspoint is higly multimodal and provides a good basis for adaptive services.

APC has devoted part of its development to user interface design. Many features have been implemented in a first multimodal Accesspoint prototype. This is considered to be crucial not only for the accesspoint services but the acceptance of the whole system where essential functionalities are speech controlled dialogs. The second prototype will follow and offer adaptive features as well. It will enable enhanced understanding of voice commands and beter dialog design will be an important step towards natural language understanding.

3.2. Project Status

During the first year, comprehensive analysis and functional specifications have culminated into the first multi-modal prototype accesspoint. Its basic technological implementation has been tested during operations in numerous public space applications as a geographical information system and marketing support media among many others.

By previous customer projects experience and internal evaluations we derived the need for a new extended architecture and to re-implement most of the software solutions in order to comply with originally defined functional specifications, implement usability and new functional requirements, establish a control and distribution network for the terminals and tune overall performance.

During the second year, development of modular component based modules will address the lack of functionality and interoperability that are currently missing. Goals for the second prototype are guidelines for basic user interface design and the simulation of a sample application. Another focus is to comply with open architecture standards and maintain accessibility of web-hosted data.

3.3. Methods and results concerning usability

As indicated in the project status, usability data has been acquired by log file analysis and video surveillance of user sessions. Furthermore, users have been asked to participate in opinion polls and fill in questionnaires.

Pre-evaluation of the user interface has been carried out together with design experts by APC corporate partners. Results were mostly focused on acceptance of the system in public space, and address the design of the solid, speech and graphical user interface. It has been found that a major issue in the design of the solid user interface is ergonomics.

The limiting factor is usability of the graphical user interface. It is followed by the integration of speech control features into the user interface.

3.4. Functional Specifications

The most important requirement is the stability of the system on a high-level user application level, which is a consequence of the availability of basic network and terminal functionality.

System failures caused by user operation or network malfunctions have to be avoided. Dead ends in dialogs have to be addressed by a special help system and contextual understanding has to be provided. The system must guide users throughout the interaction process quickly and effectively and it has to be designed to avoid cognitive overloads caused by information overflow.

The accesspoint has to be controllable through different modalities. A switch between modalities must not affect the system operations, consequently there should be no need to re-initiate the dialog status. Likewise, it should be possible to switch the current context without loosing perspective in case the system does not estimate the context properly.

3.4.1. General changing requirements and extensions

There is an urgent need for extensibility of the software system. The monolithic implementation which is used at the moment does not support modelling of upcoming functionality requests and it is very difficult to maintain and extend. Although the first prototype complies with the specified functionality we have decided to re-implement it from scratch and carefully design all application interfaces for process communications.

Better development tools to handle the speech system on an abstract level are required to improve the process of extending its rule database. For better extensibility, context engines need to be implemented rather than interfaces and the method of choice are neural networks. As for training purposes new tools are needed and should be able to derive input from rule databases which are used at the moment.

3.4.2. Usability related specifications

There are two major aspects which are of vital importance for the acceptance of the terminal in public space applications. Firstly the solid user interface requires high standards in ergonomics and robustness. Secondly the system needs to have a ``subtle notion'' of its services to guide users quickly and effectively.

The terminal pro actively offers its services i.e. must be able to attract and help people. It must be self explicable and understand users' needs. It must also be able to handle context specific dialogs. Dialogs which it doesn't understand clearly should be handled properly to limit user frustration. Context switches must be supported at any state of the session and the system should offer alternatives and rank them by guessing its likelyhood.

The graphical and speech user interface need to complement each other. They represent input and output channels and provide access to the state of the context

engine. The information flow must be compatible with average user capabilities to avoid cognitive overload. The graphical user interface model presents information in an abstract two dimensional way and supports virtual characters.

Further points address training of the context engine of which some have already been mentioned above.

3.4.3. Adaptivity related specifications

The following adaptivity specifications are limited to single user sessions. It is assumed that multiple user sessions are not very likely to occur in public space applications. They are treated elsewhere.

A general single user session occurs when one person in front of the terminal interacts with the system. An onlooker, or person inside the interaction area of the terminal is already considered to be a user.

Some adaptivity related specifications are partly indicated in section ``Usability related specifications'' above. Contextual understanding is an example and support of context switches without explicit requests are another. The interaction process should be as natural as possible and the development should ultimatly enable natural language understanding.

``Technically'' speaking, the system should be able to detect people automatically and doesn't need to be approached by a subject. It should be able to initiate the first step and react adaptively because it can recognize people and distinguish humans from other life, e.g. dogs.

Another requirement is speaker independent voice recognition which is essential in public space applications, i.e. every single user is understood instantly irrelevant of age, gender or other distinct characteristics.

3.4.4. Multimodality related specifications

The terminal is designed to support classical input-output channels like typing or reading, and complement them with speech and visual capabilities.

Except for high level features, most of the above mentioned adaptivity specifications primarily rely on multimodality related features. Visual and audio surveillance enable user detection and language understanding through face and speech recognition. At the moment, higher level functions and more powerful features for enhanced sensing can be implemented by combining speech and visual information.

Mode switches which are not initiated by the system are only possible through input channels. Output presentation is determined by the system and the user has no direct influence in this case. If desired the user can override default settings, but because of privacy issues, certain output channels are preferred.

3.4.5. Simulation related specifications

Certain new usage scenarios require extended functionality which has not been implemented yet. Simulation and testing will provide further insight into the integration process and give more practical hints on usability, before new features

materialize. The system functionality should be extensible and able to be integrated into the simulation environment to interface existing and new features at the same time.

To describe the newly defined specification, the simulation environment should provide an abstract high level scripting language thus providing an interpreter.

3.4.6. Architectural issues

To ensure extensibility of the software system, an open architecture has been chosen. All functionalities will be implemented using components or even distributed components in a client-server oriented approach.

Because of multimedia capabilities operating system selection is virtually limited to the MS Windows based platform.

The implementations of the context engine with near natural language capabilities, the multiple modality support functions and the speech and graphical interfaces are heavily component based.

4. Functional specifications and architecture for domain Home (1) (Home Experience and Multimodal Jukebox)

4.1. Introduction

Deleted from public version

5. Functional specifications and architecture for domain Home (2) (UI editor for the consumer domain)

5.1. Introduction

Home entertainment systems are embedded-computer systems that deliver entertainment content to consumers in their homes. The home environment places special constraints on entertainment systems, such as audio-video systems.

The expected evolution is that home systems can adapt to suit to individual preferences, different contexts of use, and different types of content.

The user interfaces of adaptive home systems will have to communicate this adaptivity to the user.

A first step towards adaptive user interfaces is to have an easy way to create customisable or “flexible” user interfaces.

Flexibility means here: adaptivity in the development phase. However, the process to develop these flexible user interfaces needs to be improved and supported by tools. Currently, user interfaces are mostly specified on paper with sometimes-limited tool support, and it takes many man-years to develop them. Prototypes or simulations, if developed well, are far more comprehensive than lots of pages of description and they allow early validation of the usability in a cost-effective way.

In the computer world, authoring tools or Rapid Application Development tools are used to prototype and build the graphical parts of an application. These parts are compiled into executable code.

An authoring tool for the consumer world can use a similar approach as the user interface development tools in the computer world. However, the consumer world has extra constraints that have to be taken into account: in particular, these are limited RAM, lack of hard disk and comparably slow processors.

Philips DVS aims at developing a prototype of an authoring tool for visualisation, specification, design, code generation of user interfaces for consumer products, with particular emphasis on screen based user interfaces and control, and taking into account the constraints of the consumer world as mentioned above.

The high level requirements for this authoring tool are:

- The authoring tool should allow easy modification of the behaviour of user interfaces.
- The authoring tool should support easy development of adaptive user interfaces.
- The authoring tool should allow the functional requirements specification of the next generation of consumer products to be developed in a much faster and efficient manner.

The first prototype of the authoring tool offers a basic functionality. This functionality has been testing extensively by expert users in a pilot project. Their experiences resulted in more knowledge, needed to specify new requirements.

In the mean time, a common reference model for adaptive systems has been developed.

The second prototype should offer a more extended functionality based on the new requirements and it should take into account the common reference model.

The specific requirements are listed in section 5.4.1.

The aim of the two prototypes is to give us insight in the feasibility of the development of an authoring tool for user interface development of consumer products.

5.2. Project status

In the first year of the project, we started with an early version of the first prototype and focussed on the specification of the requirements for this prototype, by doing usability tests. This resulted in a list of requirements for the first prototype as described in section 5.4.1.

Also our contribution to the reference model for adaptivity (document D3) gave us a better insight in the domain of adaptivity.

In the first half of the second project-year, the first prototype has been used in a pilot project, where a user interface for a specific consumer system, a DVD (Digital Versatile Disc) player was developed. This usability testing by expert users in a real project was needed to gain better knowledge on what we expect from an authoring tool. The requirements for a second prototype, specified in section 5.4.1 are based on the knowledge gathered in these activities.

5.3. Methods and results concerning usability

The first prototype has been tested extensively by expert users: the user interface of a DVD player has been simulated, a prototype UI has been developed and tested by UI developers. Although the first prototype offered a limited functionality, it gave us the possibility to gain experience on the desired functionality of an authoring tool.

5.4. Functional specifications

5.4.1. General changing requirements and extensions

In general, an authoring tool is a software development tool which aims at the construction of application programs in a user-friendly way, and which allows a significant reduction in development lead-time.

Our authoring tool specifically targets specification, design, simulation and code generation, for the development of on-screen-based user interfaces and control of consumer products.

The first prototype offers a basic functionality: specification, design and simulation on a PC are supported. Code generation for a target platform is not yet supported. The requirements of the first prototype are described by the requirements listed below:

- The authoring tool provides a set of standard widgets. A widget is a graphical entity representing the interface between a user and the application software. The set of standard widgets contains widgets of the following types:
  - screen, to define properties for a whole screen;
  - dialog, to provide a container for other widgets, excluding screens;
  - text, to provide a field to display a text string;
  - button, to trigger specific actions on a user request;
  - slider, to show the current state of an analogue value, e.g. volume;
  - picture, to display a bitmap, e.g. a logo.
- The authoring tool provides for each project a project repository containing a font definition, a colour palette, bitmap images and strings.
- The authoring tool allows the user to manage a palette and its colours. Colours are organised in a palette.
- The authoring tool allows the user to import bitmaps from an external source.
- The authoring tool allows the user to simulate a user interface on a PC.

An extensive evaluation of the first prototype, by trying out the tool for the development of the on-screen-based user interface of a DVD player resulted in new requirements for a second prototype.

From the UI developer's point of view, the main need was the addition of the ability to create menus in an OSD UI.

Besides this important user requirement, the second prototype of the tool should demonstrate code generation for a target platform, in this case the DVD player.

The extension - requirements for the second prototype are:

- The authoring tool provides a complex widget containing a menu structure and a navigation function.
- The authoring tool should provide a menu editor. This menu editor allows the user to define items in a menu.
- The authoring tool allows the user to generate C code for a target platform, e.g. a DVD player.

5.4.2. Usability related specifications

The general requirements, listed in section 5.1, all stress the usability aspects of the tool.

A possible usability requirement that will be considered for future extensions is: “the authoring tool should be able to assist the user in the authoring process by a wizard”.

5.4.3. Adaptivity related specifications

The first as well as the second prototype focus on off-line adaptivity: a user interface developed with the tool is created, simulated and modified on a PC. Once the UI satisfies, the authoring tool can generate C code for the target platform, e.g. a DVD player. The compiled C code can then be loaded in the target system.

Off-line adaptivity is a first step towards the development of more adaptive user interfaces. We called this flexibility instead of adaptivity. The tool offers the UI developer a more flexible way of developing a UI.

5.4.4. Multimodality related specifications

The target platforms for which user interfaces are developed with this tool, e.g. a DVD player, have several types of in- and output interfaces. Thus the authoring tool should be able to support specification, simulation and code generation of UI aspects related to these i/o peripherals. A Multimodal user interface for the authoring tool is a possible requirement for future versions.

5.4.5. Simulation related specification

As specified in the requirements of section 5.4.1, the first as well as the second prototype allow to make simulations of a user interface on a PC. Simulations are a way to get early feedback on a UI that is being developed.

5.4.6. Architectural issues

Until now, Philips Hasselt concentrated for the Beyond project on usability testing and on requirements specification. A lot of effort will be done to design the second prototype conform the requirements specified for the second prototype of the authoring tool.

6. Functional specifications and architecture for domain Vetronics (Vetronics UI editor)

6.1. Introduction

6.1.1. Requirements imposed by Vetronics Application Domain

BARCO and the LUC collaborate to realize ruggedized displays for use in several types of vehicles, for which the user interfaces can be defined in a flexible way.

Vetronics (vehicle-electronics) are computer systems embedded in special-purpose vehicles, e.g. ships, trains, trams, airplanes or vehicles used in construction industry. Examples are global positioning systems in cars, trajectory guidance in busses and metro, systems allowing the monitoring of a boat in its environment etc.

The environments in which they are used place special constraints on the displays of the Vetronics systems. "Rugged" displays are needed, which satisfy specifications such as being resistant to vibrations, extremely high or low temperatures, shocks, high degree of humidity, magnetic fields etc. These specifications must be satisfied in addition to common image specifications with regard to resolution, colors etc.

Because conclusions are taken based on the displayed image, a special demand is to display the video in very high quality without artifacts, supporting several types of video sources or graphics inputs with their specific timing parameters. In addition, the possibility to add both textual and graphic overlay to the displayed video is necessary to show navigation information, diagnostics etc.

On top of that hardware part we desire to have an ‘intelligent’ software layer having the most important feature that the User Interface is User Definable. BARCO collaborates with the LUC for the Vetronics software.

The first prototype of the UI-Editor was meant to perform some research about the general structure of the Editor. This prototype gave us an impression of how to implement such kind of program and enabled us to learn about the do’s and don’ts in this kind of application.

Another important target is to get some feedback of the market about the new concept we are introducing.

The second prototype should take into account the remarks and feedback we received on the first prototype in order to finalize the conceptual phase of the program so we can start the effective development and exploitation afterwards.

6.1.2. High-level requirements

The previous section emphasized the fact that the Vetronics domain demands ruggedized displays. In this section we will not mention general criteria for ruggedized displays, as they are well-known by BARCO from earlier developments. We will rather focus on specific targets for BARCO and LUC in the second prototype based on the requirements of the first prototype. With the feedback on this first prototype we can refine our requirements so they actually reflect our needs. Knowing that our first prototype was based on a prototype of the hardware, we can now say

that the hardware is reaching its definite form, being compliant with all the requirements. However the software part (UI-Editor and firmware support) are still in prototype phase because of the new approach for the UI-Editor.

For the Vetronics hardware and firmware:

- Optimum Video performance
- Ruggedness : Environmental Specification compliance.
- The software structure of the firmware should be suitable for adding a 'UI-interpreter' (to interprete and visualize the UI code generated by the UI Editor; so a very important component in this project). For this aspect we have chosen to add/implement a kind of virtual machine which performs the interaction between the UI-Code and the effective hardware/firmware. For performance reasons this VM is closely matched with the firmware.
- The software (firmware) should be able to control all the available hardware and the hardware state should be accessible to the firmware and be exposed to the UI-Code.
- The software architecture of the firmware should allow the extension with additional functionality.

For the User Interface Editor:

- The UI Editor should allow flexible creation and modification of the user interface of the Vetronics application. This must be done in an interactive and visual way.
- The UI Editor should be able to communicate with the Vetronics hardware, to download the developed UI to the Vetronics system for real use. This communication is done using different kinds of interfacing (Serial port, USB, FireWire, IrDa, CAN, …). This communication module is also needed to obtain the system configuration from the Vector Unit.
- The UI Editor should not only allow the management of the UI of the Vetronics application, but should also support definition of the behavior of the Vetronics system through the connection of real actions to UI elements.
- The UI Editor itself should be a powerful and user friendly tool. Because we are focussing of Windows-users, the Windows guidelines are taken info account. However, we do not want to obtain the "Designed for Windows …." Label. It should be compatible with the common versions of Windows (98/ME/2000) and near-future versions.
- The UI-Editor should be an open-application so we can add new functionality to support new hardware configurations. To obtain this we must create an open framework on which we can Plug-In all functionality we desire.

6.1.3. Flexibility versus Adaptivity

We define flexibility here as a level of adaptivity of the system, from the point of view of the UI developer. This way of working assumes that we, in a first stage, only focus

on ‘Off-Line Adaptivity’. This means that we must develop/modify the User Interface using a separate program (UI-Editor) and download it into the system in order to use the new User Interface. The adaptation takes place in between sessions with the Vetronics system. Even this off-line adaptivity is a step forward in the Vetronics domain as in most cases, the UI is realized in firmware.

6.2. Project status

During the first year of the project, a lot of analysis and research have been performed. All this research resulted in the development of a first prototype. This prototype was the first milestone in the BEYOND program.

The prototype consists of a User Interface Editor, a Vetronics system and some peripherals to try the complete system as such.

This prototype has been evaluated internally on the following basis:

- Functionality
- Compliance with requirements
- UI and ease of use
- Performance
- Major bugs/errors

This expert review resulted in some kind of score by which we could validate the prototype (or reject it).

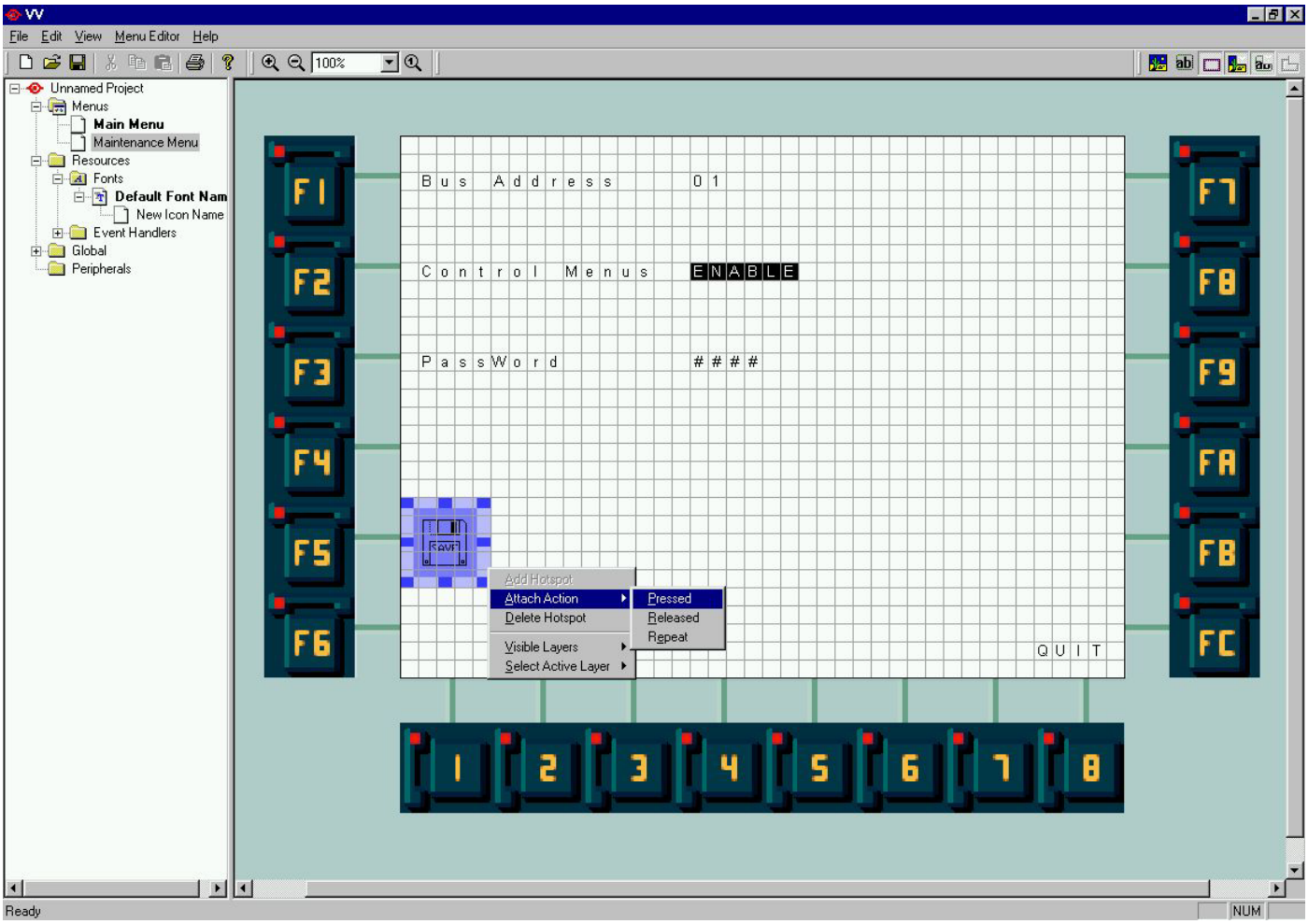
This is a summary of the remarks that have been made on this prototype:

- The user interface of the editor makes the program quite easy to use, which is very important before any definite product can be made and sold.
- The functionality of the prototype is more or less what was required
- The graphical performance could be better. Some screen updates were not OK
- A list of bugs/malfunctions has been made. Most of these have already been corrected.

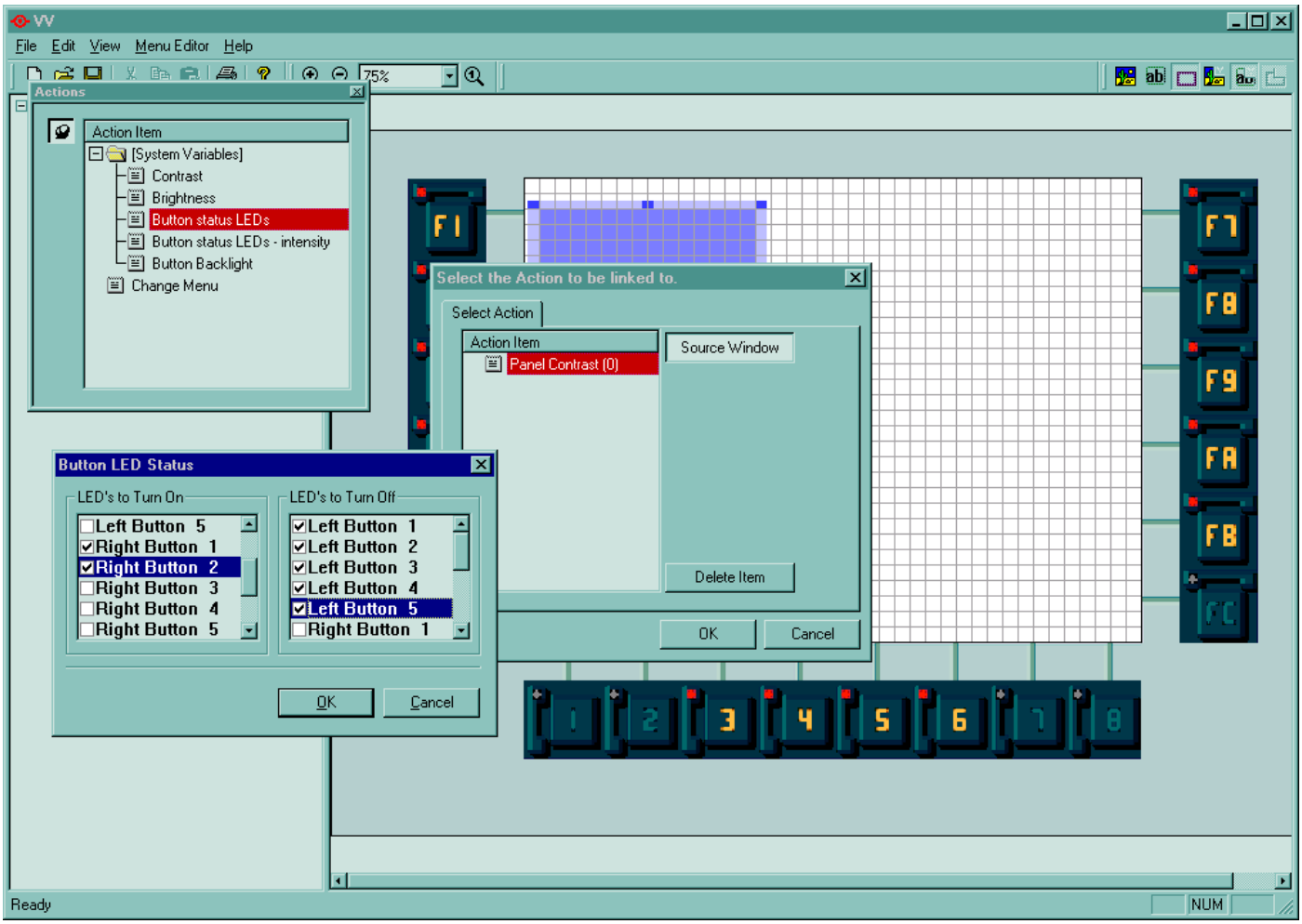
In fact, all functional requirements were fulfilled, but we noticed a major lack in the approach of the software: PlugIn-Capability.

All functional parts were statically linked to each other without any way to add additional functionality; although not without writing lots of code.

So, we have decided to start a new cycle with a well-analyzed framework so we will be able to easily plug-in new functionality onto the application. Every plug-in will then represent a piece of functional hardware.



Picture 1: Attaching an event to a hotspot for a touchscreen



Picture 2: The Event Handler Editor

6.3. Methods and results concerning usability

As already mentioned in the project status, there has been an expert review on the first prototype. This review considered both functionality and usability aspects of the program.

The expert team consisted of people from different divisions within BARCO:

- Software Quality Assurance
- Marketing
- Sales
- Software Developers
- Data Development Team

These members have been chosen for their different knowledge of software engineering and their specific use of software in general.

6.4. Functional specifications

6.4.1. General changing requirements and extensions

The list of requirements of the first prototype can be found in the D2-document. For completeness, the most important requirements are repeated below, with extensions/changes towards the second demonstrator:

Hardware and firmware requirements:

- Optimal Video performance on most popular video signals.
- Environmental Specifications Compliance.
- Event Driven Approach
- The firmware should allow extensions of the functionality.
- The software structure of the firmware should be suitable for adding a 'UI-interpreter' (to interpret and visualize the UI code generated by the UI Editor). This UI-Interpreter has now a more concrete form by means of the dedicated Virtual Machine.
- The UI interpreter (=VM) should show a "start menu" (defined in the UI by the UI Editor) when the system is powered up. This start menu is the root item for the complete UI-menu tree.
- The UI interpreter should interpret the downloaded UI code to navigate through the menus and perform defined actions. Because of the event driven firmware, the UI is built on Event-Action associations
- The UI interpreter should control the interference between the defined UI and the environmental conditions, such as overruling user-defined UI elements and actions by an "emergency" UI and behavior. This is the decision-agent we need for content prioritization.

User Interface Editor:

- System Configuration Wizard
- Font authoring modalities
- The UI Editor should allow the user to create/modify menus. Menus are in fact the combination of Overlay (OSD and Symbol Generator), HotSpot zones and Event-Action Associations.
- The UI Editor should enable the user to associate an event to an action as defined in the previous point. An event-driven approach is envisioned in the first prototype to evaluate its feasibility. Hereby the VM must support this event-driven approach and act as a transparent link between the generated UI-code and the hardware.

- The UI Editor should also dispose of a facility to associate an event to more than one action. This extension of the event-driven approach raises design issues to be tackled.

- The UI Editor should provide a compile step to generate code on the development platform. The Editor also allows downloading the generated code in the embedded Vetronics system (where it will be interpreted by the firmware). This is the way the OffLine adaptivity of the Vetronics system is based on.

- In general, the UI Editor should be a user friendly and powerful development tool. This requires:

    - Attention for visualization and simulation issues (WYSIWYG approach, e.g. Visualization of the designed UI, menus, icons, Simulating navigation etc.). The Windows guidelines should be followed.

    - Attention for solutions that speed up the development of the UI (e.g. investigate the use of drag-and-drop tools such as in the context of action definition).

    - Attention for general usability issues and user interface design.

Our main extension of the requirements for the UI-Editor is the Plug-In- approach of the program. In order to meet this extra requirement a lot of investigation on the architecture of the UI-Editor has been done and seriously evaluated. This resulted in a component-based approach where all functionality could easily be 'Plugged' into the editor. As a result, we are able to add the Symbol Generator Functionality to the Editor, even with expanded possibilities like macro definition, text capability and bitmap operations. All these important features exposed by the hardware will now be fully supported in the UI-Editor, which gives us expanding possibilities.

6.4.2. Usability related specifications

The usability has been verified by the expert review of the first prototype. Because of the variety of SW-experience and interest we covered a wide range of usability issues, from non-technical, commercial to pure technical.

Every member had his specific remarks on the software, as expected, some were more about the Look-And-Feel, and others were more technical about bugs and inconsistencies. However, all remarks were very useful for the evaluation of this first prototype:

- Tree-Type overview looks very familiar to most users, might seem complicated at first sight, it gives fast access to all modalities of the editor

- An important usability requirement with respect to the UI-Editor is the WYSIWYG layout using the Windows Guidelines. The hardware layout is always visible during development. But, this layout is now very static, it does not provide the ability to modify the layout (e.g. button layout) towards the real application. Also derived Vetronics products, with different housing, LCD, … are not covered.

- Font creation/modification is really easy and performing well.

- Some artifacts are noted during screen updates.

- Communication ports are statically implemented into the editor. When new hardware is available, a lot of additional implementation work has to be done to support the hardware (e.g. CAN-bus, USB, …).

- The complete UI-Editor has to be updated if modifications are made. This is an anomaly when we want to give support/updates via the Internet.

- Finally we concluded that the prototype met its functional requirements, but it was too static. From our hardware development and customer requirements we had learned that all have very specific needs concerning not only on User Interface but also real Hardware Interfacing to several peripherals. So we must be able to support this wide (and growing) range of interfacing capabilities of our Vetronics hardware in our UI-Editor as kind of Plug-Ins. This will also solve our maintainability issue.

6.4.3. Adaptivity related specifications

The generated User Interface is only off-line adaptive. Which means the UI is modified on a separate computer and then reloaded into the Vector System. The Vector system has some agents inside to prioritize the information available (UI-information, system status, Environment data, …). The most important data must be shown at the topmost level!

Though using off-line adaptivity, the running Vetronics application (based on code realized by means of the UI Editor) can demonstrate “intelligent” behavior to some extent.

The User Interface has access to its own data (directly UI-related), but also to data provided by the firmware concerning the current system status and its environment.

As a consequence, there is a need for a kind of decision algorithm (=agent) in order to decide whether the UI is more important than the system/environment data or vice-versa, depending on the actual content. This will reflect in the information the user gets to see. It is possible that in dangerous conditions the environmental data force the defined UI to be overruled by some built-in UI elements (e.g. warnings and error messages).

The UI-Editor itself enables us to create and modify the User Interface off-line and then reload it. In the definite program we can implement wizards in order to guide the user through several tasks or warn him for contradictory situations. These self-learning wizards will recognize the user’s behavior and identify occasions to interfere and help. In this case, we need a kind of user profile and a behavior database in order to take the right decisions/predictions.

In this context, we will use the reference model in which we collaborated with the other members of WP2 for the Deliverable 3. This reference model will be used in the agent approach of both the firmware and editor. Although extension will be rather limited in first, we want to do some research about extending its implementation in both the Firmware and UI-Editor.

6.4.4. Multimodality related specifications

The relation of this project to multimodality is rather limited. However, our Vector system should be able to communicate with all kinds of peripherals. Thus, not only the hardware should support these peripherals, but also the UI-Editor should contain the possibility to define protocols, identify several types of communication ports, …

The integration of additional I/O possibilities like speech output, sound generation, … is still under consideration. In Vetronics markets, probably due to the heavy environmental conditions (temperature, vibrations, noise, …) we cannot really use fine multimodal devices.

However, auditive information is one of the most used feedback methods in train applications. Also Touch Input devices are rather frequently used because they do not require additional place.

Currently, support for a touch screen is provided in the UI-Editor.

6.4.5. Simulation related specifications

To make our UI-Editor complete, it must have a simulation module incorporated so we can try the complete system off-line on our development computer before loading the generated code into the Vector System. As a consequence, the Vector system does not have to be available while developing and testing the UI. This is a very important demand because the systems are mostly built in into consoles where they are not easily accessible for downloading etc.

An important result of the simulation may be the ability to certify the generated UI for use in life critical conditions. This can be achieved when using an identical Virtual Machine in both the simulator and firmware. This is to be investigated after the finalization of our application and is certainly beyond the context of this BEYOND project.

6.4.6. Architectural issues

Because this new approach demands flexible functionality by using Plug-Ins, we needed to redesign the framework from a static one to a component based one.

In order to be sure we can cover all flexibility requirements a lot of effort is now being done in the complete architectural redesign of the UI-Editor. This architecture is thus considered as a very open framework (the base application) where we can easily plug in all Plug-Ins we need. To accomplish this, we have chosen to use an component based approach by using the latest SW-technologies that are usable within this type of application/platform, namely COM & COM+.

An idea of the application structure and contents is illustrated by the figure below:

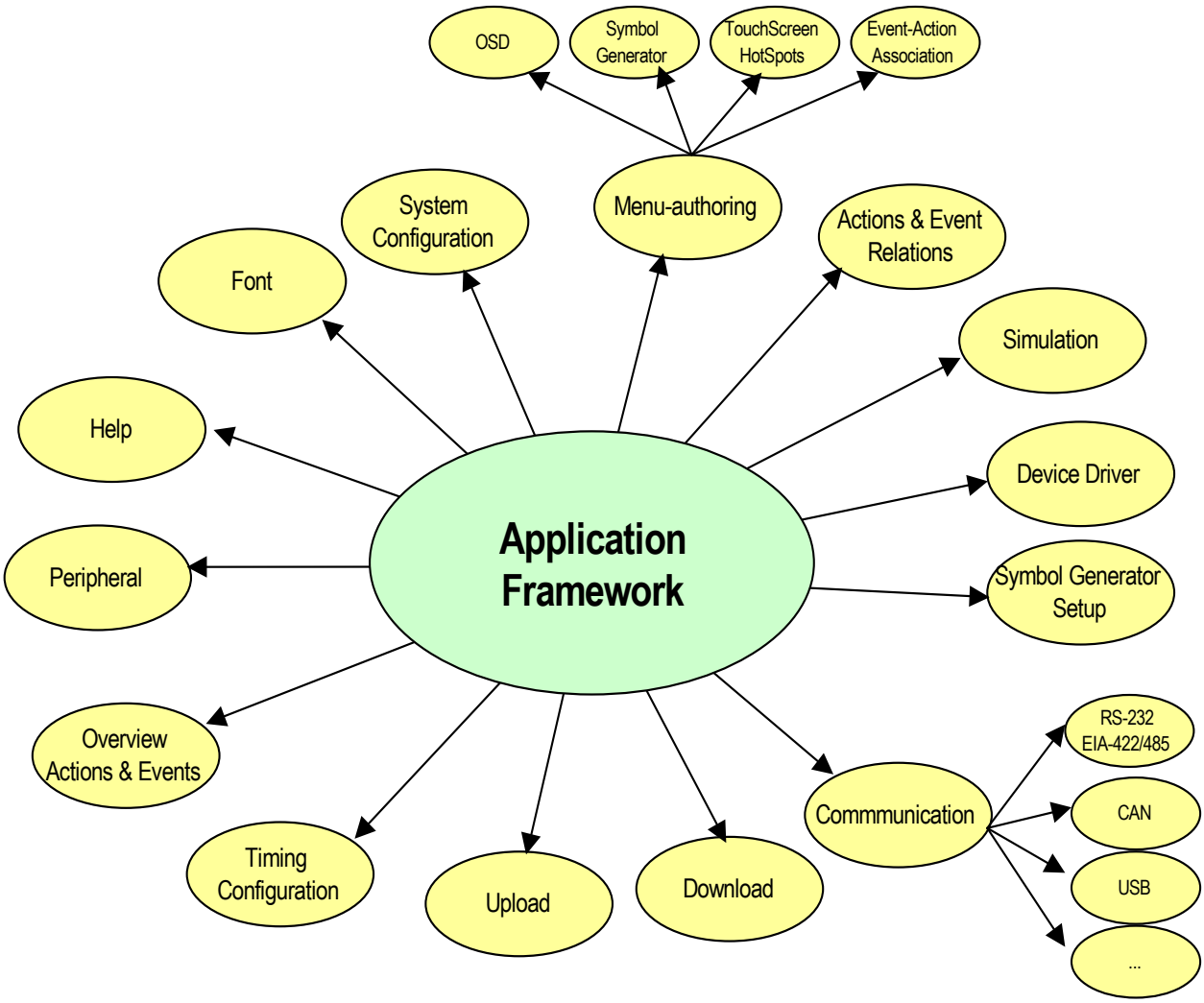


Figure 3

7. Functional specifications and architecture for domain Avionics (Intelligent Adaptive Flight Deck)

7.1. Introduction

7.1.1. Purpose of the intelligent adaptive flight deck

BARCO and TUDelft collaborate in this context.It is expected that a well-designed intelligent adaptive flight deck can significantly increase the flight safety and efficiency compared to the collection of static (but often adaptable is *not* adaptive) and diverse displays in the cockpit of today. Accident analysis has shown that 65% of all aircraft incidents are caused by human error. Two kinds of human errors can be distinguished: *slips* and *mistakes* [Reason, 1987]. A slip occurs when the intention is right, but a deviation of that intention occurs. A mistake occurs when the actions are according to the intended plan, but the plan is inadequate to achieve the intended goal. It is expected that slips and mistakes in aircraft operation can be avoided by an intelligent adaptive flight deck that assists the flight crew in performing their tasks. An intelligent system that knows the human intentions and the actual and predicted flight status can detect slips from the intended plan. If the system is aware of the goals of the operation, the intelligent system can even give a recommendation to the pilot to correct for the error.

In most cases, mistakes are not simply due to a bad knowledge of the theory or procedure, but are rather a logical consequence of a lack of *situation awareness*. The flight deck communicates to the pilot information about the flight status (flight plan, position, velocity, etc.), the environment (weather, traffic, airports, etc.), and system status (aircraft system failures, etc.). To form an internal representation of the flight situation, the crew has to integrate all this information presented to them on numerous head-down cockpit displays in different formats. In addition, the auditory channel is used to present advice, warnings and alerts to the crew. Especially during critical situations, where the situation awareness is of greatest importance, pilots have difficulties building up a representation of the flight situation. At the same time, several buttons can lighten up, various alerts can be given, and information is highlighted on the display. The flight crew then has to determine the most critical problem and the right procedure to solve it, not an easy task for the human being with only a limited view of the situation and it is clear that situations like these contribute considerably to pilot workload. Mistakes can be prevented by assisting the pilot in building situation awareness. It is expected that an intelligent adaptive interface that presents the right information in the right format (integrated and intuitive for that particular situation) at the right time (with an appropriate alerting strategy) can significantly increase flight safety.

Another contributor to the high workload during critical situations in the cockpit is the task of decision making. Even if the flight crew is able to determine a good resolution to solve the problem, they have too little overview to optimize it. An intelligent flight deck can help the pilot by proposing optimized alternatives out of which the pilot can choose one. In such a co-operation, the pilot and intelligent flight deck share their knowledge and capabilities to improve the safety and efficiency of the overall system.

7.1.2. Levels of the adaptation process

Four stages can be distinguished in the adaptation process: initiation, proposal, decision, and execution [Dieterich et al., 1993]. The agents performing or controlling these stages are the pilot of the automation. The following levels of adaptation are applicable to the intelligent adaptive flight deck [Abeloos et al., 2000]:

- Self-adaptation: the automation performs the tasks on all stages of the adaptation. This adaptation is contradicting with the human-centered design philosophy and *seems* therefore unacceptable. In certain time-critical, high workload situations, however, it may be necessary or even mandatory to automatically adapt the interface to draw the crew's attention and to allow an *immediate* but sound response. Because the adaptation is unexpected and may cause confusion, its occurrence must be well known and trained by the cockpit crew.

- User-controlled self-adaptation: the decision to adapt is taken by the user, while all other tasks are automated. If the automation thinks that for the current situation a different presentation may be more efficient, it may suggest an adaptation that has to be agreed upon by the pilot.

- User-initiated self-adaptation: the user takes the initiative, the automation proposes, decides and executes. This is in fact self-adaptation, but allowing the pilot to take the initiative.

- Computer-aided adaptation: on a user's initiative, the automation proposes an adaptation, which it will execute after the user's approval.

- System-initiated adaptation: on a system's initiative, the user proposes, decides and executes the adaptation. The pilot is then informed if it seems reasonable to tailor the system.

- Adaptation: the user performs the tasks on all stages of the adaptation. Simple adaptation gives the opportunity to users to tailor the system to their own needs and preferences. This already exists in the cockpit of today, where the pilots can control the brightness, contrast, etc. of the display directly. This type of adaptation can better be described as *flexibility*.

For a full review of the applicability of adaptivity in a future intelligent flight deck, the reader is referred to [Abeloos, 2000].

7.1.3. Proposal for the first prototype

The first prototype of the intelligent adaptive flight deck has to assist the pilot in:

- Establishing and maintaining situation awareness by presenting the right information in the right format at the right time.

- Detecting and correcting errors by comparing the overall system's goals, the pilot intentions, and the actual and predicted flight state.

- Decision making by proposing optimized alternatives.

- Carrying out some simple actions by comparing the active procedures with the pilot's actions.

The first prototype should be able to solve conflicting hazardous situations in an optimized manner. It does so by supporting all levels of adaptation that have been discussed above. In the case of self-adaptation, the pilot shall always be offered the opportunity to overrule the adaptation. These issues have all been worked out in more detail in the requirements document D2. As will be described below, the first three levels of assistance have been implemented in the first prototype.

7.2. Project status

Description of the first prototype

To observe the environment, the first prototype of the intelligent adaptive flight deck consisted of the implementation of two aircraft warning systems, the Ground Proximity Warning System (GPWS) and the Traffic alert and Collision Avoidance System (TCAS). The agent-oriented system architecture that is applied allows these systems to be integrated on a system level. This means that the two systems can communicate on all levels of their functioning. They can exchange rough environmental information, the detection of hazardous situations, and resolution advisories to solve for these critical situations. This data exchange allows the *system*:

To form one integrated picture of the environment,

To present alerts in the right order, to apply a good alerting strategy, and

To negotiate resolution advisories, so that the resolution of the one critical situation does not induce another critical situation.

The intelligent interface itself has been implemented as a group of agents collaborating with the various system agents in a shared ontology (Figure 4). This means that all functions on all levels of the interface could communicate with each other. The first three levels of assistance (establishing situation awareness, detecting and correcting errors, decision making) listed above have been incorporated in the first prototype.

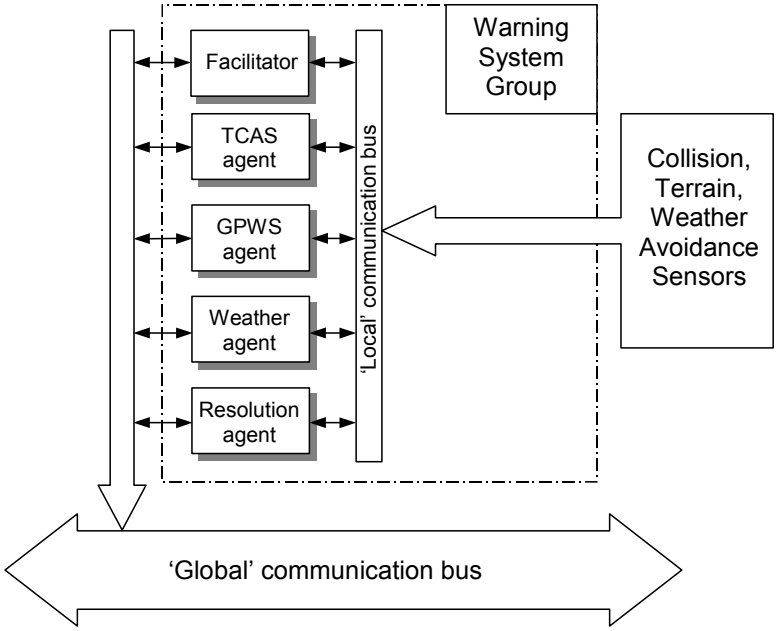


Figure 4 : Warning system group communication organization (first prototype)

To help the pilot establishing and maintaining situation awareness (assistance level 1), the pilot interface consisted of two advanced displays showing the pilot a three-dimensional synthetic view of the environment (the Primary Flight Display (PFD)) accompanied by a two-dimensional bird's eye view of the environment (the Navigation Display (ND)). To detect and correct for errors, and to help the pilot in his/her decision making (assistance levels 2 and 3), the warning-system adaptation was implemented in particular on the Navigation Display, including the GPWS and TCAS systems introduced above. The range in the ND was automatically adapted to show the pilots the cause(s) for the warning signals. Furthermore, display features that were not useful when a threat did not occur were 'darkened' automatically, i.e. the dark cockpit concept, or removed automatically. A realistic scenario was developed to test the functionality of the first prototype.

7.3. Methods and results concering usability

Testing the first prototype

The first prototype has been tested in a future airspace environment, consisting of other traffic and terrain. Other environmental factors, such as weather and Air Traffic Control were not considered. The first prototype has been tested using a *questionnaire*. A fixed demonstration has been developed that showed all features of the adaptive display in a realistic scenario. No user interaction was possible in this stage. The demonstration has been shown to a group of 23 experts in the field of avionics: 17 avionics and software engineers, 3 flight test engineers and 3 commercial pilots. The participants were first briefed extensively to introduce the intelligent flight deck and explain the scenario. After the demonstration the participants were asked to fill out an extensive questionnaire, addressing pilot acceptability, workload, situation awareness, and many others. The results of the questionnaire analysis have been described in a report [Steentjes & Mulder, 2000]. Some of the main outcomes are described below.

The questionnaire evaluation can be concluded successful, and guides the future development of the second prototype. For the near future (around february 2001) a second group of experts, consisting of pilot trainees, has been invited to see the demonstration and fill out the questionnaire addressing the first prototype.

7.4. Functional specifications

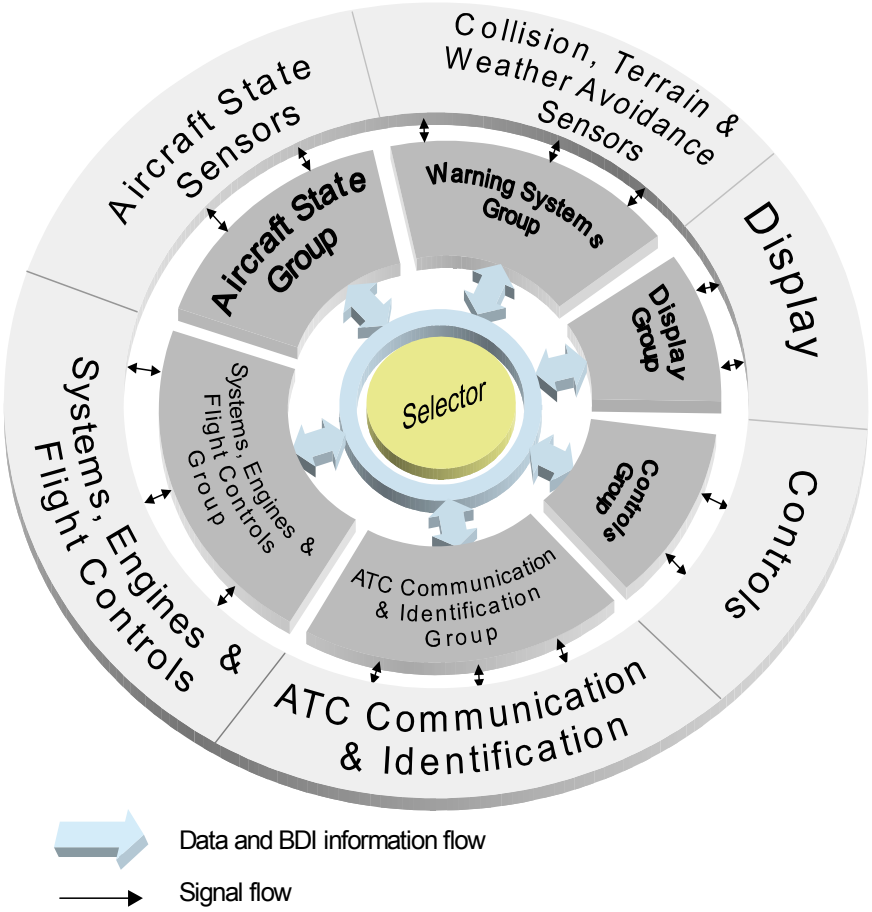
7.4.1. General changing requirements and extensions

General

The first prototype of the intelligent flight deck is able to build and maintain an internal model of a relatively small sub-set of the environment (objects, possibilities, constraints) and of some of the interactions within the human-machine system itself (goals, capabilities, preferences, etc.). The second prototype will extend the first one considerably. The questionnaire revealed that especially the way of communicating this internal situation awareness (the 'system' awareness) to the pilot should be improved. Experts expressed a need of 'seeing through' the automation and adaptation, and the user interface should have an option to provide them that possibility. Furthermore, experts claim the opportunity to overrule the system mental model, and in this respect the user interface needs to be altered. But these findings do not call for changing the requirements as such. The findings from the pilot questionnaire were in general in line with, or were envisaged in the requirements as specified in Deliverable D2 of this project. Hence, the number of changes in the general requirements is small.

The existing multi-agent system architecture of the first prototype (Figure 4) will not be altered, but rather be upgraded and significantly extended (Figure 5). The extensions will be directed almost exclusively towards the automatic, adaptive management of content information on the display (Display Group and Control Group in Figure 5), as defined by the integrated warning system architecture. Questions to be answered include the manner in which information that is not critical but still valuable should be handled (dark cockpit?). The multi-agent subsystem that deals with managing content and inferring user intent needs to be extended, an effort that will require considerable attention.

Figure 5 : Multi-agent system architecture of the second prototype (the warning system architecture is included in this figure as the Warning Systems group)



Changed requirements

One requirement from D2 is dropped:
The second prototype will incorporate the use of touch-screen technology.
This will not be feasible within the available time.

7.4.2. Usability related specifications

The use of a pilot/expert questionnaire in the evaluation of the first prototype has been successful. It has lead to a significant insight into how the user group perceives the purpose and use of the adaptive system. Therefore, also the second prototype will be evaluated using a questionnaire. The user group will again consist of a mixed group consisting of pilots, flight test engineers and avionics specialists.

7.4.3. Adaptivity related specifications

The adaptivity related specifications of D2 need not be changed at all. The adaptive interface shall adapt to the situation and the pilot's state and task. The adapted constituents are modality (see below), display configuration, information presented, level of detail, and timing strategy.

Special interest will be given to the aforementioned issue that the adaptation should leave the pilot in command, and should be non-intrusive. Furthermore, the consequences of the adaptation on the user interface itself should be intuitive, and

support visual momentum. The usability analysis will put extra attention to these aspects.

7.4.4. Multimodality related specifications

As indicated in the requirements, the second prototype shall incorporate an auditory warning signal, augmenting the visual channel. The warning signal will be the result of the integrated warning system. In other words, the auditory warning will present the pilot a resolution that all participating warning systems (intelligent agents) agree upon.

7.4.5. Simulation related specifications

The multi-agent adaptive intelligent flight deck system will be programmed in Java, using JACK Intelligent Agents [tm], a product of Agent Oriented Software Pty. Ltd, Victoria, Australia. Furthermore, all user interfaces (displays, controls) are programmed in OpenGL. The platform is a Windows NT workstation. No efforts are being conducted to generalize the real-time simulation software for product simulation purposes.

7.4.6. Architectural issues

These have been covered above.

References

[1] Abeloos, A.L.M. (2000). The Intelligent Adaptive Flight Deck. Technical Report, Delft University of Technology in co-operation with Barco Display Systems.
[2] Abeloos, A..L.M., Mulder, M., and van Paassen, M.M. (2000). The Applicability of an Adaptive Human-Machine Interface in the Cockpit. 19th European Annual Conference on Human Decision Making and Manual Control, Ispra, Italy, June, 26-28, 2000.
[3] Dieterich, H., Malinowski, U., Kühme, T., and Schneider-Hufschmidt, M. (1993). State of the Art in Adaptive User Interfaces. In Schneider-Hufschmidt, M. et al. (Eds.), Adaptive User Interfaces, Principles and Practices, North-Holland, pp. 13-48.
[4] Reason, J. (1987). Generic Error-Modelling System (GEMS): A Cognitive Framework for Locating Common Human Error Forms. In Rasmussen, J. et al. (Eds.), New Technology and Human Error, John Wiley & Sons Ltd., pp. 63-83.
[5] Steentjes, A. and Mulder, M. (2000). Analysis of a Questionnaire addressing Free Flight's "Big Picture" Displays. Report, Delft University of Technology in co-operation with Barco Display Systems.

8. Conclusions

In this document we have summarized the progress in the BEYOND project and clarified the approach in the process of going from the first to the second milestone. In particular, functional requirements and possibly their translation into architectural issues have been listed. This has been illustrated through the individual projects within the targeted application domains. In the description the key aspects multimodality, adaptivity, simulation and usability have been considered as independent variables, allowing to show the progress for all workpackages in the project.

While all of the projects evolve in one way or another from the first to the second prototype, there are big differences in that evolution. Between the alternatives we find extremes such as performing a second iteration with different points of attention, or looking for alternative solutions for the same research questions as asked earlier in the project.

In correspondence with the allocation of resources in individual workpackages, multimodality and/or adaptivity and/or simulation and/or usability are stressed differently in the migration to the second prototype. It catches the eye that several projects envisage architectural changes to accomodate the desired extensions.

Aspects of multimodality and simulation have only been focussed upon in this document as far as this contributes to the overall functional specifications of the presented projects. Details of these aspects will appear in deliverables per workpackage.

With regard to adaptivity, the diversity in feasibility and approach stated in deliverables D2 (requirements) and D3 (adaptivity reference framework) are confirmed in the functional specifications.

It has been mentioned already that in the “emergency scenario” (no subsidy for German partners and no replacement), most partners had to tackle the usability issues by themselves. Wherever appropriate, individual prototypes have been subject to usability studies. This was not always possible: the limited nature of the first prototype and time constraints (BEYOND is a short term project) are mentioned amongst other restrictions. However, even if usability studies are carried out it is remarked that more longitudinal studies are required in this context, e.g. with regard to adaptivity issues. Questionnaires and expert reviews have been applied with success within different projects and domains.

Depending on the project, architectural issues translate functional specifications to (most of the time high-level) system design, or list additional specifications. Projects involving adaptivity usually refer to D3, “adaptivity reference framework”, in their section about architecture.

The presented projects also follow general tendencies with regard to software development. In particular, at least two projects emphasize that a component-based (or plug-in) architecture is envisioned to enhance the flexibility of the final demonstrator.

In conclusion, the presented projects are subject to a smooth migration from the first to the second prototype with varying points of attention. This shows from the fact that the functional specifications and architecture are based upon but more detailed than the requirements listed in deliverable D2.