AALBORG UNIVERSITY

Intelligent MultiMedia

Title	: inTelly.dk – an intelligent TV-guide
Project period	: September 2000 – June 2001
Project group	: IMM1074/2001
Group members	: Peter Michael Jensen Kenneth Kammersgård Lars Kromann
Supervisor	: Lars Bo Larsen
Copies	: 9 pcs.
Main Report	: 188 pages
Test Report	: 157 pages
Appendices & Supplements	: 129 pages

Abstract

In this project there has been developed an intelligent and personalised TV-guide that shall ease the everyday task of selecting what to watch in the television. The focus of the project has been on the User-Centred Design, the personalisation and the use of software agents.

The system was developed using the User-Centred Design method through-out the project, based on the principles of: Early and continual focus on users and user testing, iterative design and integrated design. Templates where made to make it easier to perform future tests and to give a higher consistency.

There has been developed a new method for categorising and personalising TV-programs. The artificial intelligence used in this method has been tested on Neural Networks and Bayesian Networks.

A proposal for a functional Agent framework has been made. Agents have been used to make it possible to distribute tasks between multiple computers. The agents have been developed using Java technology.

Java Servlets have been used for providing data, gathered by the agents, for the user at a web page. The data is presented in a browser using XML and XSL, which provides the ability and flexibility to separate data from layout.

There has been developed a complete usable website in this project. There has been reflected on the 5 different usability tests used.

Foreword

This report has been composed by the project group IMM1074/2001 at the Intelligent MultiMedia specialization at Aalborg University in the period from September 2000 to June 2001. The report is dedicated to the supervisor of the group, Mindpass A/S, the external examiners and fellow students who might be interested in the issues presented in this project.

This project is divided into three parts, where the first part is the Main report, which contains the overall description of the project and the decisions made. Test video recordings, report documents, etc. is located on a CD enclosed with the Main report. The second part is a Test report where all the user tests made in the project are presented. The appendices and supplements made for this project are located in the third part.

The software developed in this project is developed in collaboration with Mindpass A/S and is confidential, but can in certain situations be retrieved by contacting either of the three group members.

Literature references are like this [GFL, p.44-56] and a bibliography is placed at the end of the Main report. References to the Test report are in the form Test report – Test plan – Questionnaire and references to the appendices are in the form Appendix H. The figures are consecutive numerated.

Lars Kromann
sealclub@bigfoot.com

Peter Michael Jensen
sods@oncable.dk

Kenneth Kammersgård
kenneth@kammersgaard.net

Contents

INTRODUCTION

1. Introduction

The problem to be solved in this project is to create a system, which can make it easier for users of an Internet service to find the information they are looking for. This project is narrowed down to concentrate on TV-programs as the information source. It is the intension to create a personalised TV-guide that should make it easier for the user to find out what to watch in TV.

The main objectives in the project is to:

- Use the method of User-Centred Design in all phases of the development in order to reflect on and thoroughly evaluate the method.
- Investigate whether software agents are appropriate to be used in this kind of Internet application.
- Make a suggestion of a method to personalise the TV-guide.

The purpose of the developed system is to work as a personalised TV-guide, but the issues treated in the project could also be used in other areas, and not only on TV-data.

Different tools are used in this project. For keeping the schedule Microsoft Project 2000 is used, and the primary development tool used in this project is JBuilder 3.0 Professional from Borland Inc. Microsoft FrontPage 2000 is used to develop user interface mock-ups.

The name inTelly is a combination of the words intelligence and telly (everyday talk for television).

1.1 Structure of the Main Report

The project is developed using the User-Centred Design method and the structure of the Main report is reflecting this method. In this section there will be a description of the different parts in the Main report.

Introduction

The introduction starts with a general presentation of the project. A chapter will explain the method of User-Centred Design. This part also includes chapters concerning the theory about software agents and user profiles in general.

Pre-analysis

The pre-analysis starts by defining the system shortly. The following chapter will define the user group. Next there is a chapter concerning information gathering, where influential systems, new technologies, and interesting standards are presented.

Initial Design

In the initial design part, the users and their requirements are analysed. In the analysis of the users and their requirements two exploratory tests are used. In this section there will also be developed some functional requirements and testable goals that the final system should fulfil.

Iterative Development

This part describes the iterative development of the system. It contains three iterations where each version of the system is evaluated by a test. The first test is a heuristic evaluation of the first user interface mock-ups (version 0.1). Next there is made an assessment test, to test primary functions of the system (version 0.2). The last iteration implements the missing functionality and a heuristic evaluation is used (on version 0.3) in the evaluation of the system before the final version (1.0) is developed.

Final Design – Version 1.0

In this part all details about the final design of the system is described. This includes the user interface, the Agent framework together with the agents used, the artificial intelligence used for the personalisation, etc. The functional requirements and testable goals are tested and evaluated. The usability goals (part of the testable goals) are tested and evaluated by the use of a validation test.

Conclusion

This part will reflect and conclude on the already presented main objectives of the project.

2. User-Centred Design

The method used in this project is the User-Centred Design Method. It is used as a guideline for the complete development and design of the system. One of the main purposes of using the User-Centred Design method is to develop a system with a high level of usability.

The method used is called User-Centred Design and is a method that keeps focus on the users at all stages, in the development of the system. There are used different sources in the creation of the method. The primary sources are [JDG], [CL], [JN1], [LU], [JN2] and [EB].

There are four principles in User-Centred Design, which are presented by [JDG]. The four principles are:

- **Early and continual focus on users**. The users should be involved in all stages of the development.
- **Early and continual user testing**. To obtain a system with a high level of usability user testing at all stages of the development is necessary.
- **Iterative design**. The development of the system should be structured by making, testing and evaluate prototypes in an iterative process.
- **Integrated design**. All aspects of usability should be considered from the beginning of the development (help facilities, user manuals etc).

(For a further explanation see Appendix A).

The four principles are used in the different phases of the development process. They are used to make a procedure for designing User-Centred systems, which will be presented in this section. Egil Boisen [EB] has used the principles given by [JDG] and [JN1] to make a procedure to be used in all phases of the system development. The project group has used this procedure as a starting point in the making of a development procedure. This means that the project group has adjusted the procedure where necessary. The resulting procedure made by the project group has four design phases (see Figure 1).

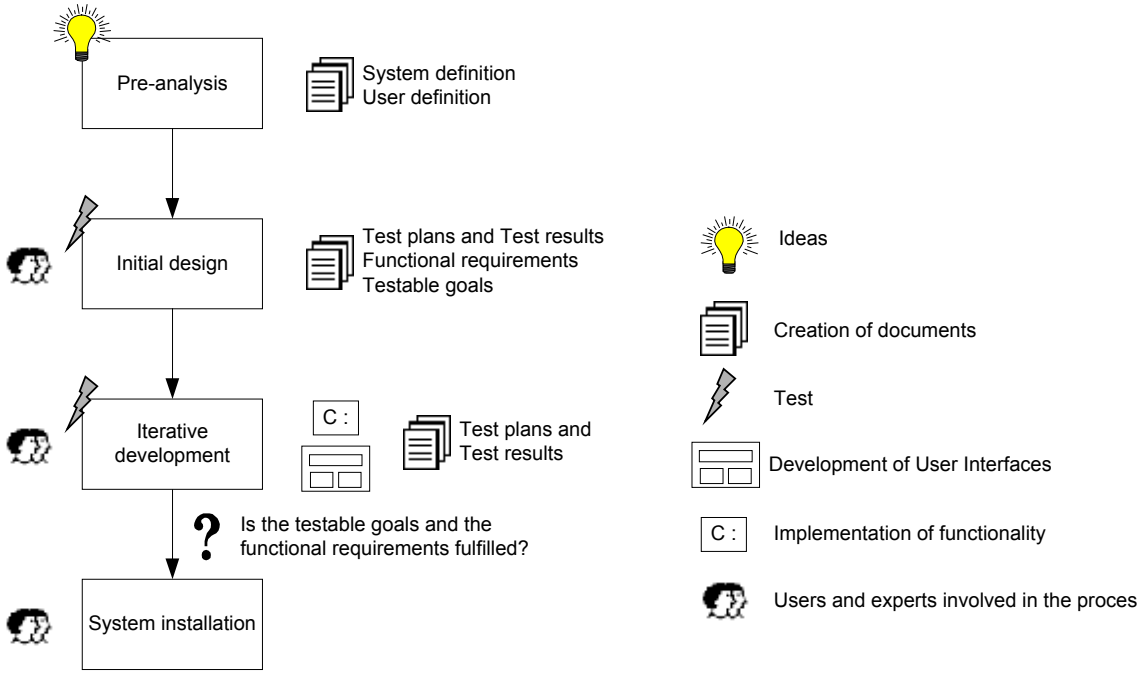

Figure 1 : The four main design phases.

The first phase (Pre-analysis) consists of the general considerations made before the project group talks to the users: defining user group and the system etc. The second phase (Initial design) includes a study of the user requirements, talking to the users etc. The next and third phase (Iterative development) is where the iterations are made by making prototypes, tests and evaluation of the tests repeated until the system is considered ready to be tested on the functional requirements and testable goals. If the system passes the final test the development enters the final phase (System installation), which handles the release of the system like the installation and the introduction to the users. If the system fails to pass the final test it should be considered if a new iteration is necessary. It should be noticed that this project will not enter the System installation phase, but the testable goals and functional requirements will be tested and evaluated on the final version of the system.

In the following sections the different analysis, design and test methods used in the four phases are listed.

2.1 Pre-analysis

In this phase, the system and the users are defined and information in general is gathered, and the concept of the system is described. The goal is to develop a clear, shared and communicable vision of the product [LU]. In this phase there are no contact with the users.

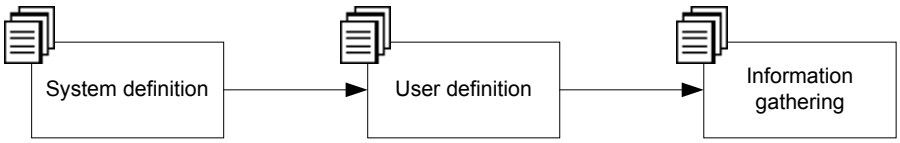

Figure 2 : Pre-analysis phase. This phase result in several documents specifying different parts of the system.

Contents

The contents of the pre-analysis phase are (see also Figure 2):

1. **System definition**
   - The purpose of the system. This will present the overall purpose of the system.
   - The contents of the system. A presentation of what the system will contain, without too many details.
2. **User definition**
   - A definition of the user group.
3. **Information gathering**
   - A presentation of the follow-on systems. Follow-on systems are existing systems that need to be taken into account, when designing the new system (e.g. an existing infrastructure that the new system must integrate with).
   - Considerations about influential systems (e.g. competing products).
   - A presentation of which new technologies that will be used in the system.
   - A presentation of the user circumstances (e.g. user terminology).
   - A short going through standards that will be used in the development of the system.
   - A presentation of possible other designers and consultants that could be used in the development of the system.

Results

The result of the pre-analysis will be a document describing the system and the users. This document will be used as a starting point in the Initial design phase that follows the pre-analysis. It should also be used to get some ideas of the direction of the project at this early stage and to make documentation of the overall project goals.

2.2 Initial Design

The goal of the initial design phase is to develop a comprehensive and systematic analysis of users and their requirements, through studying the users needs, expectations, tasks and work process, and determine the implications for the user interface of this information. [LU]

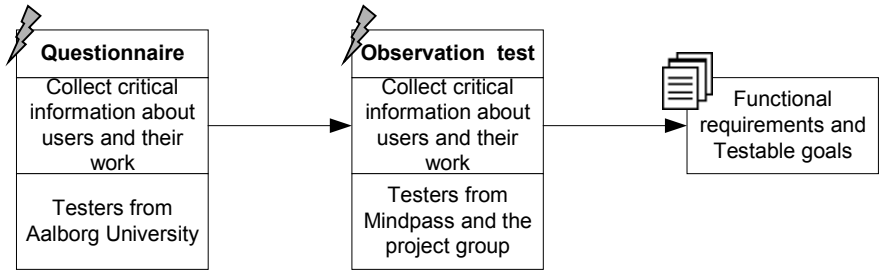

Figure 3 : Initial design phase. The two exploratory tests Questionnaire and Observation test are performed before the Functional requirements and Testable goals are developed.

Contents

The initial design phase are separated in following main parts (Figure 3):

- Questionnaire.
- Observation test.
- Development of functional requirements
- Development of testable goals.

To collect critical information about the users there will be made two exploratory tests; a Questionnaire and an Observation test. The purpose is to find out what the user likes and dislikes about existing systems, which problems and wishes does the user have etc. The Observation test is performed by observing users solving a specified problem using the possibilities that are available to them before the new system is developed, using competing products (e.g. find out which movies are on the television tonight). The result of the information collection will be a list of functional requirements and testable goals, which will be used to evaluate the final system.

Some methods suggested by [JDG] in Appendix A are left out for different reasons. E.g. there is no point in visiting the customer location, because this could be everywhere, where there is a PC connected to the Internet present, it could be at home or at work, etc.

Results

The results of the initial design phase consists of the following items:

- Functional requirements.
- Testable goals.

The results of the initial design phase are used as a starting point in the iterative development phase.

2.3 Iterative Development

In this phase the purpose is to develop a User-Centred product. The iterative development phase consist of several iterations, each iteration consist of three sub-phases starting with a development of a prototype (see Figure 4). The three sub-phases are presented in detail in the next section.

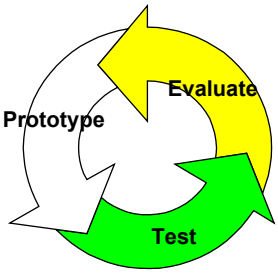

Figure 4 : Each iteration consists of three sub-phases, which starts with a prototype.

In Figure 5 there is a sketch of the different iterations in the iterative development phase.

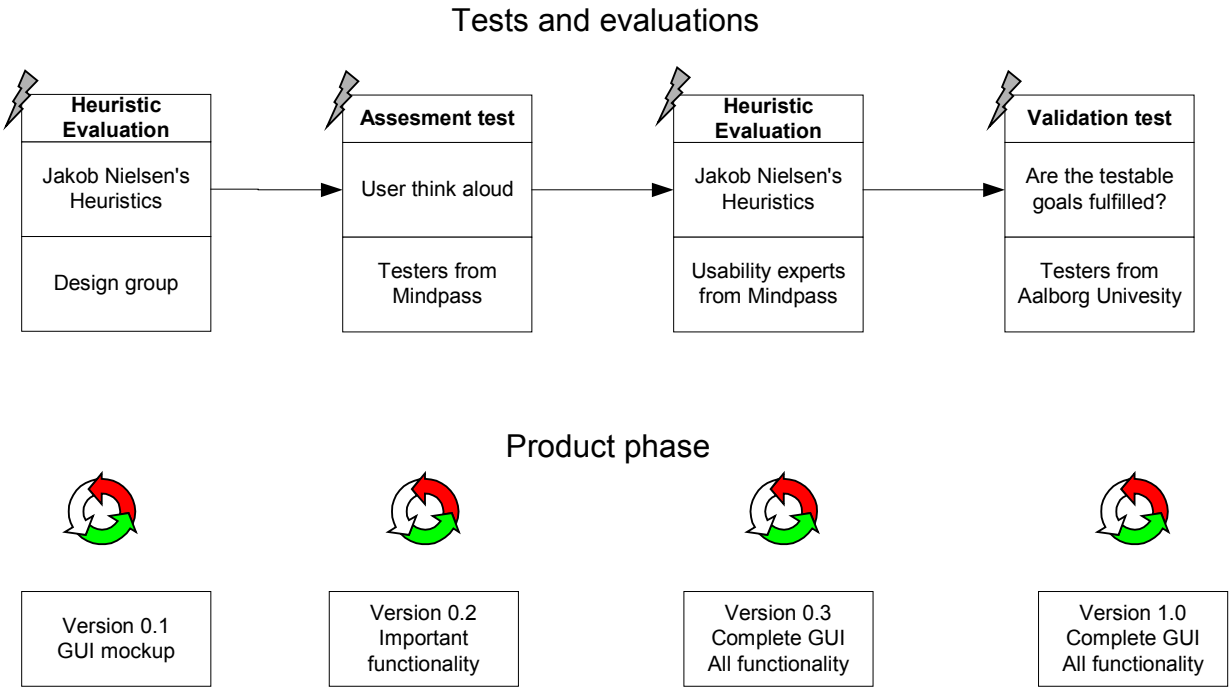

Figure 5 : Iterative development phase. Each version of the system consists of a prototype that is tested and evaluated.

Every box in the figure presents the test used at a given time in the iterative development phase, the test method and the participants performing the test. The tests used in the beginning

of the development phase will primary put focus on **why** issues in the test (e.g. why the user is performing a certain task). Later in the development phase the focus point will gradually change to focus on **how** issues in the test (e.g. how the test participants are performing a task). Furthermore the version and state of the system is displayed in the figure.

The use of the different tests and evaluations performed in the design phases will be reflected upon as part of the conclusion of the project (see chapter 31. Reflection). The development of the design phase has been made by the project group based on the sources listed in the beginning of this chapter.

The results of each iteration are:

- A prototype of the system.
- A Test plan document used in the test of the prototype.
- A Test result document of the test performed.

The test plans and test results are all to be found in their full length in the Test report.

The four iterations are considered by the project group to be an appropriate number of iterations. The four iterations make it possible to test the main parts of the system. The iterations will give an idea of the design of the user interfaces, test the basic functionality of the system, test the complete system where both interaction and functionality is tested and at last verify the functional requirements and testable goals to evaluate whether the system fulfils the developed demands.

In the following sections there will be a description of the sub-phases and the associated test methods.

Contents

The sub-phases in each iteration contains the following:

1. **Prototype**: When starting an iteration, the first to be done is to design and implement a prototype. The prototype is created by performing a brainstorm, using the test results from previous iterations if available, the functional requirements and testable goals to structure and implement the system. This could be paper and pencil mock-ups, graphical user interfaces with or without functionality etc. As the development advances the prototype will gradually be brought near to the final product ready for release. At the beginning of the iterative development phase there will typically be worked on more than one prototype at a time (will be described further below).
2. **Test**: The tests will change character throughout the development and will as already mentioned primary focus on **why** issues in the beginning of the development and

primary focus on **how** issues in the end of the development phase. A short presentation of the different tests is given below. The tests are also described in more details in Appendix C. Before the test is performed there should also be made some considerations about how the test results should be analysed and evaluated.

3. **Evaluate**: When the test is performed, the results from the test should be evaluated. This will be dependent of the test performed. There is a significant difference in the analysis and evaluation of qualitative data compared to quantitative data.

The sub-phases correspond to Gould's [CL] five sub-phases presented in his iterative design. The prototype phase corresponds to Gould's change and implementation phases, and test equals his testing and at last the evaluate phase are equal to Gould's feedback and evaluation phases.

In the following list there is a short description of the test to be performed in the iterative development phase. For further information about the different usability tests see also Appendix C.

- **Heuristic evaluation**: This evaluation is based on Jakob Nielsen's usability heuristics, which are used to evaluate the user interface. The design group evaluates the first preliminary design, based on the functional requirements and testable goals. The user interfaces presented to the users should be significant different in order to get the maximum benefit from the test. Later in the development, a heuristic evaluation is performed by usability experts, this is when the complete system with all functionality is implemented.
- **Assessment test**: When this test is to be performed the primary functionality and user interface should be implemented. The results from the heuristic evaluation performed on the preliminary user interface are used to make a user interface "draft". The primary functionality and user interface is tested, by letting users think aloud when using the implemented system.
- **Validation test**: In this state all the functionality is implemented. The design is now considered ready to be evaluated and therefore a validation test is performed by users that try to solve problems on their own. The test though is not only used to get some qualitative results from a group of users but it shall also test some well-defined quantitative performance criteria (the testable goals). Another test that could be useful is a try-to-destroy-it contest. This test type is primary performed in order to evaluate the design under stress and under unusual conditions. After the release date there could also be made tests when the users have become more familiar with the system in order to make adjustments to the product.

When the design has passed the validation test it is ready to enter the system installation phase (see the subsequent section).

Results

The result of the iterative development phase will be a User-Centred product, fulfilling the functional requirements and testable goals specified in the initial design phase. After this phase the design is ready to be installed and will be ready for use.

2.4 System Installation

The system installation phase will most importantly include the installation at the purchaser, a final test making sure that the system works "out-of-the-box" and an introduction of the system to the end users. The steps are illustrated in Figure 6.

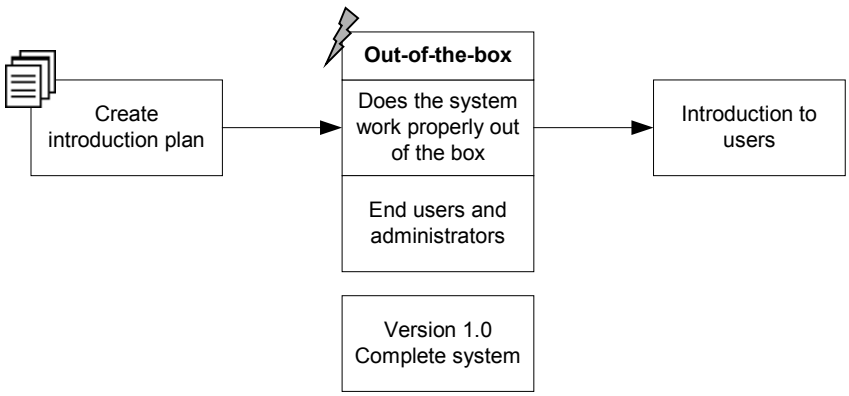

Figure 6 : System installation phase.

It should be noticed that this phase is not carried out in this project.

Contents

The detailed content of the system installation phase is:

1. Creation of an introduction plan [LU]. This plan is used to organize the complete installation programme. By making a plan of the things to be done in this phase will ensure that nothing is left out when the design team leaves the costumer locations.
2. Installation of the system in costumer's locations.
3. "Out of the box" test [LU]. This test is used as a final test that should ensure that everything is working as intended.
4. Introduction to the end users. In order to deliver a useful product the end users should be well introduced to the system and the possibilities given by the system. If the introduction is omitted there will be a risk that the system will not be used optimally. Some training material should accompany this introduction in order to ensure that future users also will be able to learn to draw the maximum benefit from the system.

5. Assure costumer acceptance. It is important to ensure that the costumer is satisfied with the system and that there are no last-minute misunderstandings, otherwise the system might fail to be a success, which will result in unsatisfied costumers and bad publicity.
6. Follow-up studies. These studies must ensure that the experiences with the system will result in adjustments in future releases. The follow-up studies could be made by monitoring the use of the system and collect data in order to be able to evaluate the performance in daily use.

Results

The resulting documentations of the system installation phase are:

- A release plan used to control every detail in the installation phase.
- A list of future goals to be fulfilled in the next version of the system according to the experiences achieved and the data collected after the release of the product.

2.5 Conclusion

In this section the User-Centred Design method was described in detail. The method is constructed by combining several ideas given by different usability experts like John D. Gould, Jakob Nielsen and Egil Boisen as well as the project group's own experience. The result is a method that can be used in projects where the user is put in focus from the early stages in the development to the installation of the system. The four main stages in this method are:

- Pre-analysis
- Initial design
- Iterative development
- System installation

In all of the above-mentioned stages, there are users or usability experts involved (primary the last three stages). In the initial design phase the users are interviewed and asked a lot of questions. In the iterative development phase users/usability experts are used to test each new prototype. In the system installation phase users are used to test if the system is working as intended.

3. Agents

This chapter will look upon the theories, which are associated with design of software systems that uses agents. First there will be a definition of the agent concept. After that there will be looked upon agent characteristics and also the different types of agents. For this chapter there has been used the following sources: [JB], [SF], [HSN], [SAM], [YL], [TF], [RF], [AC] and [DIC]

3.1 Definition of Agents

An agent can be defined as:

- One that acts or has the power or authority to act.
- One empowered to act for or represent another.
- A means by which something is done or caused.
- A force or substance that causes a change.

[DIC]

From this it is possible to point out the following key attributes:

- Agents perform tasks.
- Agents operate on behalf of someone or something.

The two central attributes mentioned above gives the following definition of a software agent: A computing entity that performs user delegated tasks autonomously. This means that an agent performs tasks on behalf of a user and these tasks could for example be retrieval of information or mail filtering [AC, p. 6].

3.2 Agent Characteristics

A software agent as described above possesses a minimal set of characteristics, which describes its abilities. These characteristics are as follows:

- **Delegation**: The agent executes a series of tasks on behalf of the user, which are explicit permitted by the user.
- **Intelligence**: The knowledge, reasoning and learning capabilities of the agent, used for conducting the appropriate autonomous operation.
- **Communicative/Social ability**: The interaction with other agents through a specific agent communication language like e.g. KQML. This characteristic could also include interaction with the user through a user interface.
- **Reactive/Responsiveness**: Observation of the surroundings in order to be able to act autonomously in timely fashion on external events and thereby affect the environment.

- **Autonomy**: An agent conducting tasks without direct intervention from the user but based on the observations made.

An agent's basic characteristics where just presented above. These are not the only attributes agents can have. Below there is a presentation of some additional characteristics. [SAM]

- **Goal-oriented/Proactiveness**: An agent accepts high-level requests indicating what a human wants and is responsible for deciding how and when to satisfy the request. This means having the capability to take the initiative to act in such a way that will enhance the fulfilment of its intended goals rather than acting simply in response to their environment.
- **Collaborative**: An agent does not blindly obey commands, but has the ability to modify requests, ask clarification questions, or even refuse to satisfy certain requests.
- **Flexible**: The agent's actions are not scripted; it is able to dynamically choose which actions to invoke, and in what sequence, in response to the state of its external environment.
- **Temporal continuity**: Agents are continuously running processes, not "one-shot" computations that map a single input to a single output, then terminate.
- **Character**: An agent has a well-defined, believable "personality" and emotional state.
- **Adaptive/Learning**: The agent automatically customizes itself to the preferences of its user based on previous experience. The agent also automatically adapts to changes in its environment.
- **Mobile**: An agent is able to transport itself from one machine to another and across different system architectures and platforms.

It is chosen that a typical agent will have to possess at least the first five characteristics mentioned above. The other characteristics mentioned are additional characteristics that may be added to the agent if they are necessary. Now knowing the possible agent characteristics it is possible to go on with describing the different types of agents it might be possible to create from this list of attributes.

3.3 Types of Agents

In this section it is attempted to give an overview of the different agents types. This will include presenting the main characteristics for each type of agent. In the list below there is presented an overview of the different types of agents.

- Collaborative agents
- Interface agents
- Mobile agents

- Information agents
- Reactive agents
- Hybrid agents

The different agent types will be described in the following sections.

Collaborative Agents

Collaborative agents emphasise autonomy and cooperation in order to perform tasks of their owners. They are able to negotiate with other agents reach mutually acceptable agreements. These agents may be able to learn but this is not a typically characteristics of their work. Their general characteristics are: autonomy, social ability, responsiveness and goal oriented. Collaborative agents may be used to solve problems that are too large for a single centralised agent or they can help with interconnection and interoperation of multiple existing systems.

Interface Agents

Interface agents emphasise autonomy and learning in order to perform tasks for their owners. This type of agent is similar to a personal assistant who is collaborating with the user. These agents are able to support and provide assistance to a user learning a particular application. Communication with other agents is typically limited to asking for advice and not as the collaborative agent getting into negotiations. Interface agents might be used in situations where boring and laborious tasks at the user interface has to be performed. These tasks could then be delegated to the interface agents.

Mobile Agents

Mobile agents emphasise mobility, autonomy and cooperation in order to perform tasks for their owners. They are capable of roaming wide area networks like the World Wide Web, where they interact with foreign hosts in order to gather information on behalf of its owner and when having performed the given tasks they return “home”. The benefits of a mobile agent is:

- Reduced communication
- Limited local resources
- Asynchronous computing
- Flexible distributed computing architecture

Information Agents

Information agents used on the Internet emphasise autonomy, learning and cooperation in order to perform tasks for their owners. Information agents emerged because of demands for a tool that could help manage the explosive growth of information, which we are currently

experiencing. The more specific tasks of these agents are to manage, manipulate or gather information coming from distributed sources.

Reactive Agents

Reactive agents are agents that do not possess an internal symbolic model of the environments in which they are operating. These types of agent will act promptly and properly to the changes that occur to its environment. A reactive agent is to be seen as a collection of autonomous modules where each module is responsible for a specific task. This means that the features to be recognized are foreseen by the designer of the agent. For each feature recognized, an appropriate action is programmed into the agent.

Hybrid Agents

A hybrid agent is as indicated a combination of other agents. It is a combination of two or more agent types where the strengths from each type have been used to create a new and optimised agent. The reason for having these agents is that the benefits increases from having this combination of types with in a single agent is greater than the gains obtained from using an agent based on only a single type.

3.4 Agent Communication

Knowledge Query and Manipulating Language (KQML) is one of the most widely used agent communication standards [YL]. There are also a great number of common known applications that already uses KQML as communication language. KQML has furthermore shown to fulfil the needs and demands for the agent communication language needed in this project. See Appendix D for details.

KQML focuses on an extensible set of KQML messages defining some speech acts. A KQML message is called a performative and should be seen as a message, which is intended to perform some action by virtue of being sent. KQML expresses an agent's beliefs, desires and intentions. The communication is done through speech acts, which makes it possible for agents to carry out a simple dialogue (opposite to single message parsing) given by some predefined performatives. There are given 42 reserved performatives (see [TF] for a description). These performatives are split into nine main groups, which are basic informative performatives, basic query performatives, networking performatives etc. For details about these groups see Appendix D.

The information exchange between agents can be divided into synchronously and asynchronously transactions. A synchronously transaction is where a client sends a query to a server and then waits for reply. The reply from the server can then contain one or several

answers. It is also possible that the server's answer is incomplete in the way that it returns a handle that allows the client to ask for the components of the reply one at the time.

Agents can also use asynchronously transactions where a client subscribes to a server's output. This will result in an undefined number of replies that are arriving in irregular intervals.

Broadcasting is another variation of information exchange. Here messages are sent out to a number of hosts instead of sending it to a specific host. The replies might arrive synchronously or asynchronously and after their arrival they will have to be collated and optionally associated with the query that they are replying to.

The communication between agents goes from simple point-to-point communication to more complex scenarios where a facilitator agent is used. A facilitator agent is an agent that may maintain a registry of service names, forward messages to named services or provide matchmaking services between information providers and clients.

The different types of agent communication will now be described. The first is the simple point-to-point communication where an agent is aware of the other agents and their capabilities. The agent will then just send a message to another agent and ask this one for help or whatever it might need. It is also possible to use a facilitator agent when there has to be established some communication between agents. The main task of a facilitator agent is to help other agents finding appropriate clients and servers that can perform/process the necessary performatives. This means that the facilitator e.g. can find an agent that has the possibility to process specific performatives. When the facilitator agent has found a suitable agent it will forward the given task and the agent will then process the performative and reply to either the original sender or the facilitator agent, which then will forward the result of the performative to the original sender.

Another possibility is that an agent asks the facilitator to recommend a suitable agent for a given performative. If the facilitator agent is aware of any agent that can process a given performative it will reply with the name of the suitable agent. This communication example can also be seen in Figure 7.

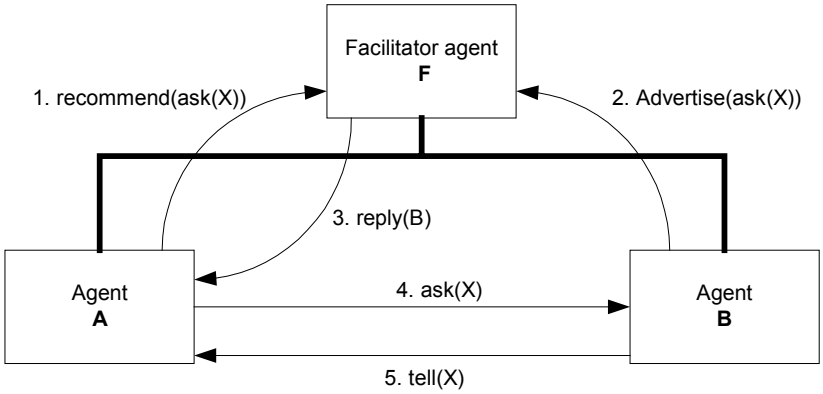

Figure 7 : This figure shows what happens when the facilitator agent receives a recommend performative.

The communication chosen in this project is the one showed on Figure 7 because it is believed to be the best suitable for the Agent framework because of its flexibility. This way of communicating makes it very easy to add extra agents if necessary, due to the fact that the other agents does not have to know about the new agents existence. The other types of communication explained in Appendix D, can easily be implemented as well.

3.5 Designing Agents

In this section there will be a presentation of the possible method that can be used in the design of an agent.

A successful development method, which has been used in traditional software development, is the object-oriented approach. Here an object is usually described as a holder of state information together with some behaviour using operations upon the state information. The object-oriented methods provide support for identifying objects, and allows for abstraction using object classes and inheritance by means of class hierarchies.

A traditional object-oriented approach is not directly applicable to agent systems, because normally agents are more complex than typical objects. This goes for both their internal structure and in the behaviour they show. Agents not only consist of methods and attributes, but also of mental states and they use concepts such as plans and goals. Normally objects are believed to be passive and activated by messages or method invocations. Agents instead can be proactive and decide for themselves how and when to respond to messages. As seen there are a few things that differ between objects and agents, but this is nothing compared to the number similarities that are between them. Therefore it is not out of the way to base a methodology on ideas from object-oriented methodologies and therefore the following will give an overview of a specific agent-oriented method.

When specifying an agent system there can be used the following approach: decompose the system into agents. These agents are then characterised by their purpose, responsibilities, services, required and maintained information and external interactions.

A system is best decomposed based on the key roles of the system. Identifying these roles and their relationships, should give a specification of an agent class hierarchy. By further analysing the responsibilities of each class one can identify the services provided and used by an agent, and thus its external interactions. If one considers the creation and duration of roles it is possible to determine the control relationships between classes.

Two models are used to hold this information

1. The Agent Model is used to describe the hierarchical relationship between the agent classes. It also holds information about the agent instances and their creation and duration.
2. The Interaction Model contains the responsibilities of each class, its services, interactions and control relationships between classes.

Note that roles can be seen as sets of responsibilities. Responsibilities on the other hand can be seen as sets of services. Services then, are the activities that cannot be decomposed further in a natural way.

It is now possible to create a stepwise procedure for creating these models and this can be seen as follows:

- First identify the roles of the application domain so that it is possible to identify the type of agent/agents that is to be used. Also identify the lifetime of each role. Then build up a hierarchy of agent classes.
- Study each role further and try to locate the related responsibilities together with the services that they consist of. Then decompose each agent class into the level of services.
- Now, for each service, identify the interactions necessary to provide the service. Also identify events and special conditions that should be noticed, as well as actions that should be performed. This is done to see if the “standard” agent type can be used or extra characteristics has to be added.
- It is now time to refine the agent hierarchy.

[NS]

4. User Profiles

In this chapter the user profiles and their influence on information retrieval systems will be presented. Furthermore there will be a description of the generation of user profiles and finally the grouping of user profiles into stereotypes will be presented. The user profiles are to be considered in order to personalise information. This chapter is mainly based on [BB].

In an information society like the one we have today it is necessary to reduce the amount of information which is presented to the user of a electronic information retrieval system so that the user will not get an information overload. This reduction of information can be done in several ways. One of them is only to present a specified number of results for the user without thinking of what is thrown away. Another way could be to filter the information before it is presented to the user and in that way remove irrelevant information.

To do such filtering the filter has to be fitted with some data that describes the users interests and in that way gives him/her the relevant information. The data describing the user interests can be seen as a profile of the user and this means that the user profile is a description of the user and its purpose is to give a more individual response when interacting with the information retrieval system.

4.1 Classification

In this section there is a presentation of the different types of user profiles and a short description of these.

Implicit vs. Explicit: An explicit user model is a model, which is created explicit by the user, where as the implicit created user profile is made from an analysis of the users behaviour by the information system.

Given vs. inferred: By a given user model is meant a model that is created during the design phase of the system and can not be changed where as the inferred is analysing the user during the use of the system and thereby the system creates a user profile.

Static vs. Dynamic: When the user model does not change during the use of the system it is said to be static, but when the user profile is updated during the use of the system it is classified as a dynamic model.

Canonical vs. individual: A canonical model is a model that represents several users, where as the individual model is a model of the individual user.

Long term vs. short term: A short term user model is made in the beginning of e.g. a search session and will be deleted when ending the session, where as the long term model will be used several times.

4.2 Generation and Modification

The collection of data for generation of user profiles can be done in different ways. There is the explicit way and the implicit way and this also goes for the modification of the user profile.

By an explicit generation of the user profile is meant that the user gives the system some input, which is seen as the core of the profile. This explicit generation of the profile can also be done in different ways, namely by use of a questionnaire or by using a number of keywords describing the users interests. When using a questionnaire it is possible to get some additional information about the user besides their interests like their social status, age etc. Another possibility is the use of keywords that represents the users information needs. This gives the advantage that the profile can get very specific if the user knows how to create such a profile. The disadvantage about this way of generating a profile is that it takes time and requires a lot of practice to make a good profile. Letting the user express his/her information needs via some relevant documents and then letting the information system sort out the keywords is also a way of creating this collection of keywords.

The creation of a user profile the implicit way means that the information system will analyse the user and create the profile from this analysis. This could be done by e.g. letting some software monitor the user actions. The system will typically use a statistical method for deducing the user profile.

The modification of a profile can also be done either implicit or explicit. In the case of implicit modification the system itself makes these modifications to the profile after having monitored the user for a while or maybe getting access to new relevant data about the user. The explicit made profile is changed either by letting the user modify the profile through the same interface, which was used for creating it or by ranking the information found by the system.

4.3 Stereotypes

When creating a user profile it is possible to group the users into user groups where all users have the same main interests. A group of users that all have a set of main interests is called a stereotype. To be able to use stereotypes it is necessary to define the stereotypes. Dividing the user group into stereotypes can e.g. by done by some expert within the area. The stereotype

will contain a set of interests for one or more of the users. When the definition of the stereotypes is finished the users of a system has to be assigned to one or more of the stereotypes. The allocation of one or more stereotypes to a user can be done in different ways. One way is to observe when a user chooses a sports program and then apply the stereotype “sports interested” to the user. The other way of assigning a stereotype to a user is by letting the system compare the user profile of a user with the different stereotypes and then let it choose the one that fits the user best.

PRE-ANALYSIS

5. Introduction

The main objective of this part is to clarify and describe the system to be designed, and to give some definitions of the users of the systems, etc. The language used in this section is kept in general terms so that it is easy understood for e.g. exterior readers. The primary reason to keep the language simple is that this section is describing the system before any further decisions about the details are taken. This will ensure that many considerations and decisions will be taken after the initial ideas of the system have been presented to the future users.

The main subjects in this part are:

- System definition.
    - The purpose of the system.
    - The contents of the system.
- User definition.
- Information gathering.
    - A presentation of the follow-on systems.
    - Considerations about influential systems (competing products).
    - New technologies.
    - A presentation of the user circumstances (user terminology, etc.).
    - A short going through standards that will be used in the development of the system.
    - A presentation of possible other designers and consultants used in the development phase.

6. System Definition

This section contains a description of the system purpose and the contents of the system.

In the latest years there has been an enormous growth in the amount of information placed on the Internet. This major increase in information has made it difficult for a user to find the relevant information the user is looking for. Therefore it would be convenient with a system that could find the information the user is interested in and remove the irrelevant information.

Such a system should have the purpose of filtering and personalising information given by different Internet information channels like: news, TV-guide, etc. To be able to suggest relevant information the system needs a profile of the user.

It has been chosen to put focus on a TV-guide. This choice has been made because this is a service that Mindpass A/S currently does not have in their portal toolbox and because the project group considers that the currently available Internet TV-guides could be significantly improved. The principles for the personalisation of the TV-guide are identical to those that might be used to personalise other information channels, e.g. newsgroups and news. Due to the similarities between different information channels it is decided to discard the other information channels and only concentrate on the TV-guide.

6.1 Purpose of the System

The main purpose of the system is to filter and personalise the information presented in a TV-guide. To be able to suggest TV-programs there must be made a user profile, which the system can use to find out user relevant information.

It has been chosen to find out whether the use of software agents is a good solution for this type of application.

The purpose of the system can be written in the following sections:

- From a given user profile, information gathered from the Internet is filtered and personalised and information that could be interesting to the user is sorted and presented to the user.
- There will be an examination of whether software agents can be used for the preceding purpose.
- There will be made some considerations and development of methods to personalise the TV-programs.

6.2 The Contents of the System

In the system there will be an evaluation of the TV-data, which will make it possible e.g. to sort TV-programs according to a user's needs.

The system will have to contain a user profile for each user in order to adapt the information to different users. There are several possibilities to construct and maintain a user profile. These considerations were described in chapter 4. User Profiles.

The system will also contain some sort of artificial intelligence. This intelligence will together with the user profile help to find the personalised TV-data for each user.

Basic search facilities shall also be available to the user of the system, if the user decides to search for something that is not present in the profile of the user.

At last the system has to contain help facilities that might help the user in understanding the system.

7. User Definition

This section will present the definition of the users of the system. In this project it has been decided to concentrate on the group of advanced users.

One of the reasons for concentrating on the advanced users is because this group of users are typically not afraid of trying something new. This means that advanced users are more willing to try out things like e.g. filling out a user profile because they know the Internet and the terms used. These advanced users also have the necessary understanding about the Internet to be able to give good and possible design ideas. It will also be the more experienced Internet users, which will have greatest benefit of such a personalised information system, because they are familiar with some of the problems with the Internet (information overload and the difficulties in finding the relevant information).

The users must meet the following constraints in order to be considered as an advanced user:

- The user must have an Internet connection at home. If the user does not have an Internet connection at home it will be more difficult for him or her to use the TV-guide service when necessary.
- The user must use a somewhat new Web browser. This will indicate that the user is using the Internet regularly because many homepages demands a new Web browser in order to display the pages correctly.
- The user must use the Internet more than twice a week. This means that the user must be in frequently contact with the Internet in order to be considered an advanced user.
- The user must have used the Internet in more than a year.
- The user must have used more than one of the different Internet information channels (e.g. newsgroups, search, chat etc). A user that is only using the Internet for one of the purposes is in this project considered to be a non-advanced user. The reason is that the user must have some idea of the many different information services available.

The system should be designed in such a way that it is possible for non-experienced users to use the system. Even when chosen not to do so it will not exclude the non-experienced users but it might take these users a bit longer time to set up the system.

It is chosen not to take the maintenance users into account in the development of the system (system administrators) because of time constraints.

8. Information Gathering

In this chapter there will be a presentation of the things that might influence on the design of the TV-guide. Considerations about existing systems and/or user terminologies will be described. There will also be presented new technologies and standards that might be used in the design.

8.1 Follow-on Systems

The follow-on systems include the existing services given by the Mindpass A/S portal, which are:

- MindLogin: authentication tool that monitors the access to restricted and/or personalised parts of the Mindpass A/S portal.
- MindRegistration: basic registration tool that enables the Mindpass A/S portal owners to create registration interfaces any way desired.
- MindProfile: component that displays the different personal information of a user account.
- FSQL: Special Fuzzy SQL database with advanced search functions.

Mindpass A/S's decision of separating their data layer from the layout layer gives the possibility of integrating with their technologies quite easy. The Mindpass A/S infrastructure will also be a follow-on system to consider when designing the information system (server structure, software available, operating systems etc.).

8.2 Influential Systems

This section will list some of the influential systems and competing products, which are currently available.

To get some inspiration and ideas about how a TV-guide could be designed there has been a search for competing products and there are currently the following competing products:

- TV-guides on paper (TV magazines, Newspapers, etc.)
- Text TV
- Internet TV-guides

In Appendix J the different competing products concerning TV-guides are described together with their pros and cons for the different products. Below there is a summary of the Internet TV-guides described in Appendix J:

- TV-channels are typically divided into language and categories (movies, sports, documentary, children etc.)
- There are typically a choice of time like the day of week, time of day and possible choices like “rest of the day”, “rest of the week”, “right now” etc.
- The sort facilities are:
    - Sort by time
    - Sort by channel
    - Sort by category
- Normally it is also possible to show/hide program details and also show/hide the Show view code.
- The possibility of creating an individual user profile contains the following term:
    - Expert/novice user
    - Channels of interest
    - Individual defined shortcuts to different settings like: channels, time etc.
    - The programs that these shortcuts point to can be emailed to the user each morning or e.g. 10 minutes before program start.
- On some of the TV-guides there are shortcuts to newsgroups concerning TV.

Only the Internet TV-guides are presented because these have all the same functionalities as the rest of the competing products.

8.3 New Technologies

The primary new technologies to be considered in this project are the Internet in general and the use of software agents. The Internet has become very popular within the last few years and will also become more popular and important within the next decade because more and more things are being made possible via the Internet. The use of agents has not yet played a major role on the Internet, but there is a tendency indicating that the use of agents on the Internet will increase in the future. These agents can e.g. be seen as small robots and they will have the possibility of being sent into the Internet e.g. searching for information.

8.4 User Circumstances

This section presents user terms that a designer ought to know before starting to develop the system.

The terminology of the users is difficult to define because the user-group is very large. This means that there cannot be listed a number of terms that should be defined and known before the development of the system. The number of technical terms should be kept at a minimum so that all within the user group are able to understand the language used.

Even if the user-group is based on advanced Internet users the designer should take into account that this user-group contains more sublevels of users so the terminology used in the system must be at a level that all users understand and not only for the experts. Therefore the language should be kept simple and relaxed, using well-known terms if possible.

8.5 Standards

This section will present some of the standards that are taken into account in making this project. The primary sources are the Internet, scientific papers and running Internet information systems.

Standards	Description
KQML	This is an abbreviation of Knowledge Query and Manipulating Language. KQML has become a de facto standard as an agent communication language. The content of the message is only a part of the communication between agents. When using KQML the agents wraps the contents of the message in a template containing the message type, the recipient, the language used in the message contents, the way the recipient should answer etc. For a more complete presentation of KQML see also Appendix D.
HTML	HTML stands for Hyper Text Mark-up Language. An HTML file is a text file containing small mark-up tags, which indicates to a Web browser how to display a page. Version 4.0 of HTML is used.
XML/XSL	XML stands for EXtensible Mark-up Language and is a mark-up language much like HTML. XML is designed to describe data and will in this project be used to contain e.g. the TV-data. XSL stand for eXtensible Stylesheet Language and is used to transform and format XML documents in order to generate HTML code that can be seen in a browser. Version MSXML 3.0 (XML version 1.0 and XSL version 1.1) is used.
Java	Java is an Object-Oriented programming language that will be used for all the server side software in this project. The main reason for using Java is that this language is platform independent and this means that the server side software can run on any platform that has a Java Virtual Machine running. Furthermore Java is well suited for developing agents. Java 1.3 is used.
Java Servlets	Servlets provide the possibility to develop Web Applications. Web Applications allow a website to be dynamic rather than static pages. The use of Servlets makes each session persistent, which is not possible in traditional scripting language like PHP and ASP. Version 2.3 is used.
MySQL	MySQL is an Open Source SQL database provided by MySQL AB. MySQL is a relational database management system and uses SQL to access the

	database. SQL (Structured Query Language) is a common standardized language used to access databases. The version of the MySQL is 3.23.33.

8.6 Other Designers and Consultants

This section lists the consultants and other designers that could be used in the development of the system. They will be used as sparring partners in discussions and when important decisions about the system are to be made.

The possible other designers and consultants used in the development of the system will be:

- The supervisor (Lars Bo Larsen)
- Employees at Mindpass A/S. Hardware and software experts, human factor experts, graphical designers etc. The Mindpass A/S employees are also used as test participants.
- Other students at the University of Aalborg primary used as test participants.

9. Conclusion

This section has presented the system and some of the considerations in generally terms.
The system purpose and the first draft of the system contents have been stated. The user group has been defined and is chosen to include advanced Internet users and not to include the system administrators maintaining the system after the implementation. The existing Mindpass A/S technology is the primary source of follow-on systems. There has also been a description of existing systems and sources of inspiration.

The result of this pre-analysis has resulted in focusing on the following:

- The primary focus in the project will be put on a personalised TV-guide.
- The TV-guide shall contain artificial intelligence, a user profile, basic search facilities and help facilities for the user.
- To solve the tasks in the system there will be used agents in order to determine whether these are well suited in this kind of system.

INITIAL DESIGN

10. Introduction

This section will present the Initial Design of the system. The main purpose of this part is to analyse the users and their requirements. The primary contents of the initial design phase are:

- Questionnaire
- Observation test.
- Functional requirements.
- Testable goals.

The critical information about the users is found by making two exploratory tests. The first test is a questionnaire asking the participants about relevant Internet sites containing TV-guides, news providers and personalisation in general. The second test is an observation test where the participants are doing some tasks on the Internet concerning TV-guides, news and personalisation.

The reason for also using news in the two exploratory tests is because this information channel is somewhat the same as a TV-guide. There are also considered to be more people reading news than checking the TV-program on the Internet, which therefore gives a better idea of how the users get information from these types of systems.

The two exploratory tests are each based on two documents: A Test plan and a Test result. The Test plan describes the way of conducting the test and the possible results whereas the Test result presents the main results. Both documents for each test are found in the Test report.

As a conclusion on the two exploratory tests there will be made a list of functional requirements and testable goals that will be used to give a more detailed specification of the system. The two tests should give an idea of how the Internet is used concerning news reading, TV-guides and personalisation.

The functional requirements and the testable goals are developed in order to state how the final system should perform.

The test participants are found by sending an email to potential advanced users primary found at Aalborg University. The questionnaire is made as a web page, where there are links to the main project homepage, which furthermore contains a project description, prototypes, tests, test results etc. The reason for using an Internet homepage is to get and hopefully keep in contact with the future users and interested persons in general. When the participants are aware of that this homepage will be updated with e.g. prototypes during the project and that

the visitors can and will be more than welcome to contribute with their opinion, it is considered that it will be possible to get into a valuable dialogue with the users.

The second test will, as already stated, be an observation test, where the users are to perform some given tasks on the Internet. The participants will be the project group and some employees at Mindpass A/S.

The test results of the questionnaire will primary be used as an inspiration to the project group, and to be used as arguments in the design. The observation test is used to more directly get a requirements list and valuable comments from the participants: in this case rather few people (group members and employees at Mindpass A/S).

11. Questionnaire

The questionnaire is used in order to retrieve information about the use of relevant Internet topics (TV-guides, news and personalisation) among advanced Internet users.

The questionnaire has been made as a web page and the participants have been recruited by sending out emails with a link to the questionnaire, which is a part of the project homepage. The main reasons for using a Web-based questionnaire is that this method makes it simple to get in contact with many participants. Furthermore the results are easily collected and the participants are free to decide where and when they want to fill out the questionnaire.

Before sending out the final questionnaire, there has been a reflection on the questions to make these as easy understandable as possible. During these consideration there has been used both guidelines taken from [KL] and also added commonsense items by the project group.

The questionnaire consists of four major parts:

1. Check whether participant is an advanced user or not (as already described).
2. Collect information about the participant's use of Internet news services.
3. Collect information about the participant's use of Internet TV-guides.
4. Collect information about the participant's use of Internet personalisation.

The user profile of the advanced users is given in the Pre-analysis. The first part of the questionnaire consists of questions that determine whether the participant is an advanced user or not. The answers from the advanced users are the primary used results (see following section).

The email has been sent out primarily to students at Aalborg University at the Institute of Electronics. The reason is that these participants are considered to be advanced users.

The evaluation measures are the answers to yes/no questions, the answers to multiple-choice questions and the comments.

The complete test plan (including the questionnaire) can be seen in the Test report – Test plan Questionnaire.

11.1 Evaluation

The questions 1 to 7 are used to define the type of user. In the final result there was 62 advanced users and 24 non-advanced users. This means that 72% of the participants where

classified as advanced users and 28% of the participants was not categorized as advanced users. This number is considered so high that it was decided to study the answers from the non-advanced users in order to see why they were not advanced users. The result was that they typically failed on only one of the parameters that determine whether they should be categorized as advanced users or not. The result was that the comments from the nearly advanced users were also considered in order to ensure that the project group did not miss any important information from the questionnaire. But it proved that the answers and comments from the non-advanced users were similar to the advanced users so the high percentage of non-advanced users is not considered a problem.

Question number 8 to number 14 deals with news on the Internet. A significant high number of the advanced users are using the Internet for news reading (69% - only news on TV gets a higher score in retrieving news). In question number 10 and 11 the participant were asked to write down their favourite news provider on the Internet and why this is the case. The reasons were many and the project group will check up on the different news providers and find out how and why they live up to the expectations given by the participants. The Internet news providers seem to fail to present the news well arranged (question number 14). This is a result that will be used in the system design.

Question number 15 to number 20 asks the participant about the use of TV-guides on the Internet. The most important result in the questions concerning the TV-guides is the reasons why the participants are not checking all TV-channels for interesting programs. The reason is typically that they do not bother to check more TV-channels for interesting programs. A personalised TV-guide should make it possible to look through and sort all relevant channels when presenting the TV-programs to the user and this is a task that will be considered important in this project. Again the participants suggest some Internet TV-guides that has been investigated further in order to get inspiration to the design.

A low percentage of the advanced users use Internet TV-guides. Only about 43% of the participants use an Internet TV-service to find interesting programs. But a personalised TV-guide that makes it easy for users to find out what they want to see could attract new users. If this TV-guide is accessible while watching TV, via the TV or a portable device, this might give a large number of new users.

Personalisation on the Internet is only tried out by about 40% of the advanced users (personalisation is question number 20 to number 24). This could indicate that this opportunity is not yet a common feature on the Internet. The participants that have tried personalisation seems more or less satisfied but there is a tendency indicating that the personalisation removes too much information and that it is difficult to enter the personal data. These two topics will therefore get extra attention in the development of the system.

The detailed result of the questionnaire can be seen in appendix Test report – Test result Questionnaire.

11.2 Conclusion

The outcome of the questionnaire is considered useful. The project group has received a number of pros and cons of existing Internet TV-guides. The results will as already mentioned be used in the system development in order to fulfil the demands of the advanced Internet users. Also the information collected about news reading and news providers will be taken into account in the design of a TV-guide. Further more the project group has received some ideas of what is good and bad about personalisation on Internet in general.

12. Observation Test

The objective of the observation test is to perceive how the users solve some predefined tasks given by the design team. The tasks to be solved can be seen in the Test report – Test plan Observation. The observation of the users working will make it possible to extract which tasks and subtasks there typically are in e.g. finding TV-programs on the Internet. During the observation it will be allowed to keep a dialogue with the user in order to clarify what he or she is thinking and get some general or more specific ideas from the user. This interaction with the user should give a good idea about where and why the user gets frustrated and which things the user likes or dislikes about the news or TV-guide sites.

The user profile of the test participants is again advanced users and the criteria used are the same as used in the questionnaire.

The following list will describe the outline of the observation

1. The first thing to happen is to ensure that the participant recruited is an advanced user. This means that the participant is asked some questions in order to determine whether he or she is an advanced user. The criteria can be seen in chapter 7. User Definition.
2. The method will be that the test monitor asks the participant the questions given in Test report – Test plan Observation one at a time.
3. When the test monitor has ensured that the participant has understood the exercise, the participant should start finding the news or TV-program he or she was asked to.
4. The participant will be asked to inform the test monitor about what he or she is doing during the exercise (thinking aloud). This will include good and/or bad things about the visited web pages etc.
5. The test monitor will write down relevant actions performed by the participant and also the utterances of the participant. This task will be rather demanding, but it will be allowed to interrupt the participant in order to get observations clarified and to get a further description of the comments uttered by the participant.
6. When the participant has completed the exercise by finding the given news or TV-program, the process will be repeated until the final exercise has been performed or some maximum time has expired.
7. The test participant should be debriefed, about his/her experience with using the different sites.

The test monitor must make it clear that the exercises are made in order to evaluate the performance of the Internet services and not the participant. This means that the participant should not be frustrated or stressed when the news or TV-programs are hard to find. The typical maximum time spend in solving the exercises is tested to be about 30 minutes. This

duration is considered appropriate, because the time is not too long so that the participant becomes impatient.

12.1 Test Result

In this section a discussion will sum up on the most important things about the news search, TV-guides and the creation of personalised user accounts.

The exercises based on news search gave the following issues to consider about when presenting data to a user:

- The things that are important for the user when reading news is mainly to get the interesting news and that these are presented in a well-arranged and structured way.
- When searching for news people either use a search engine to find news or they go directly to a site where they know it is possible to find interesting news.

The exercises based on the TV-guides gave the following issues to consider about when designing the TV-guide:

- One of the most important things about the existing TV-guides seems to be the amount of information, which is presented to the user. This should be kept at a minimum and in a well-arranged way. This also means not to display major program descriptions and things like that, but just the most necessary things like program name, channel and time.
- Many of the participants where also looking for a sort of notepad, where they could put the programs they want to see, so that they do not have to remember them when browsing for other interesting programs.
- The participants also requested the possibility of selecting a few channels, instead of one or all channels, and this without having to create a personalised account. They would also like to have a more flexible time selection, instead of the one where they only have the possibility selecting between four partitions into which the day have been split.
- A thing that is important is, when using the category filter and there is found no programs, this should be stated clearly to the user.

The exercises concerning personalisation on the Internet gave the following results:

- First of all when you want people to create a personalised user account it should be visible for them where to do this. All the way through the creation process there should also be clear exits and clearly stated where to go and what to do next. As well as it at any time should be possible to stop with the set-up of the user profile and then use current settings. When the user makes a mistake it should not be necessary to start all over again.

- The data from the user should be kept at a minimum. This means only ask them about necessary data like username and password. When asking for additional information it should be stated what this information is used for.
- The selection of the channels should be easy and fast and the channels should be grouped logically, like the language of the channel or the type programs they show. The test participants would like both the possibility of selecting whole channel groups and of handpicking channels. There are all also a demand for the possibility of adding missing channels in some way. The selection of the categories should as well as the channels be easy and fast overviewed and set-up and also grouped logically. There is also a request for keywords that are attached to each category, to make the filtering as good as possible.
- If there is a possibility for several profiles it should be possible to reuse things like the channel configuration from the earlier profiles.

The detailed results from the observation test can be seen in the Test report – Test result Observation.

12.2 Conclusion

The observation test can be considered as both useful and successful. The project group has received a number of good and bad things about the existing services. The result of this test will be used later on in the system development and hopefully help to fulfil the demands of the advanced users.

13. Functional Requirements

This section presents the functional requirements that must be fulfilled in the final TV-guide. The motivation and arguments for the functional requirements are based on the results of the Pre-analysis and the Initial design. The functional requirements will be a list of needed functions in the final design, which will include both user oriented functionality and system specific functionality (primary concerning the Agent framework).

In the following sections the functional requirements will be presented and discussed. This will include both requirements to the TV-guide and to the Agent framework.

13.1 Functional Requirements to the TV-guide

In the table below the functional requirements to the TV-guide is presented. The first column contains the functional requirement (bold and italic) and the corresponding arguments for the functional requirement. The rest of the columns will indicate the sources of the function by the use of the following abbreviations:

- **Q** Questionnaire
- **O** Observation test
- **P** Project group
- **C** Competing products
- **D** Other designers and consultants

Functional requirement	**Q**	**O**	**P**	**C**	**D**
1. There should be filters helping the user to get an easy overview of the many TV-programs available. The filters that make it possible to filter the TV-programs have proved to work well (primarily confirmed in the observation test and found by the project group trying out different Internet TV-guides). This solution has been chosen due to the fact that it is considered to be a logically and intuitive way of making it possible for the user to filter and order the data. The project group has considered a solution that will include the possibility of user-defined filters, which is considered important seen from a usability point of view.	Q	O	P	C	
2. It should be possible for the user to define the amount of information displayed for each program. This functionality should ensure that the TV-guide could be adjusted to the user's needs. One example of the functionality could be to make it possible for the users to define which columns they want present in the list of TV-programs. The project group is aware of that	Q	O	P		

a feature that more or less lets the users define their own user interface can give rise to usability problems. This will be taken into account when this functionality is to be considered in the development.					
3. There should be some notepad function where the user can place the programs he or she wants to see. The observation test showed that a notepad function would be a good feature. This feature was both wanted by the test participants and the project group. The user should have the possibility of placing interesting programs in the notepad. It is considered important to make the use of the notepad intuitive and easy to use. The observation test showed that this facility in one of the TV-guides tested gave rise to misunderstandings and confusion [TV].		O	P		
4. It should be possible to create/update/delete/view a user account. The user should have a possibility of creating and administrating personal preferences, such as categories and channels of interest. It has shown to be an important factor that the user profile is easy and quick to create and administrate.		O	P	C	
5. There should be some guidelines/wizard that helps the user in the creating the user account. The system should help and guide the user in the creation of the user profile. It has proved to be important for the user to know e.g. how many steps the user is missing in finishing the generation of the user profile and/or why certain information is needed.		O	P		
6. It should be possible for the users to add keywords. The observation test showed that the users would like to have the possibility of adding keywords both in general and to each category. This feature should make it possible for the system to help the user in finding interesting TV-programs. The keywords are to be considered as a supplement to the personalisation of the TV-guide.		O	P	C	
7. The user should be able to make a printout of the TV-program. A typical feature that is considered to be important is the printout facility. This should make it easy for the user to get a copy of the personal TV-program. The importance of this functionality is also based upon the fact that printouts of web pages often give rise to problems (truncation of lines, etc).			P	C	
8. It should be possible to port the TV-program to other devices (Palm, WAP). It has been found that a possibility of porting the TV-guide data to			P		D

different portable devices will be a feature that the users will ask for more explicitly in the future and it should therefore be included in this design.					
9. It should be possible to record selected programs. It is considered to be a good feature if the system is able to record TV-programs directly (e.g. by the use of a TV-tuner or Internet based TV-programs). An integration of the TV-guide and the recording feature is considered to optimise the use of both functions.			P		
10. The TV-guide should give the user the possibility of getting a reminder about selected TV-programs. By looking at the competing products it was found that a reminder facility would be appropriate. The users would get some information via. SMS/email etc. prior to important programs and/or e.g. receive an email with a personal TV-program each morning.			P	C	
11. It should be possible for the user to tell the system, which TV-programs he or she likes/dislikes and the system should use this information to prioritise the TV-programs in order to personalise the TV-guide. It is considered important to personalise the TV-guide. This means that the system should gradually be able to know the programs that user likes and dislikes. The user have to somehow tell the system about which programs he or she likes/dislikes, this information will then be used by the system in order to filter out non-interesting TV-program and only present the TV-programs that the user might like. It is considered important that the process of prioritising programs should be intuitive due to the fact that this feature is unusual. The testable goals for the works of the personalisation can be seen in 14. Testable Goals.	Q		P	C	D
12. The system should prioritise the TV-programs based at prioritising made by users with similar interests. As a supplement to the personalisation is has been to chosen to include a group filtering in the system. The group filtering should group the users and sort the TV-programs accordingly. One example: All users within a group want to see “Friends” expect one, who has not prioritised the program. The system might then put a high system priority on this program for the user who has not evaluated the program.			P		D
13. There should be a search feature making it possible to search in the TV-programs.		O	P		

A search feature should be present in order for the users to make a query for some given program or programs contents. The reasons for adding this feature is both that it was requested in the observation test and that it is a common website feature that the users would miss if not present.					
14. It should be possible for the user to select and deselect repeating programs (e.g. serials, news etc – called "Repeaters" throughout the report). The detection and presentation of the repeating programs is considered to be an important task due to the fact that about 70-90 percentages of the TV-programs are to be considered as Repeaters. This means that the user has the possibility of evaluating the Repeaters and letting the TV-guide find and sort the corresponding programs in the future.			P		

The function list presented above will be used in the design of the TV-guide. The functions will be tested and evaluated in the final version of the system.

13.2 Functional Requirements to the Agent Framework

The use of agents results in some additional functional requirements to the system. These requirements are not a natural part of the functional requirements to the TV-guide and are therefore presented in a separate list.

The table below presents the functional requirements to the Agent framework. It contains the functional requirement (bold and italic) and the corresponding arguments for the functional requirement

Functional requirement
1. The Agent framework should support the agent characteristics given in chapter 3. Agents (e.g. mobility, flexibility etc). The argument for the agents being mobile, flexible etc. is that these are the characteristics of agents in general. It is considered important to develop an Agent framework and agents that is in correspondence with the de facto standards in this area in order to truly evaluate whether the use of agents is a good solution in this project.
2. It should be possible to distribute the tasks and agents on a number of computers in order to achieve a load balance. The project group considered it to be an important factor that the agents is able to distribute the tasks on a number of computers. It is considered to be one of the most important features

of the agents developed in this project. This is a fact even though the scalability aspects have been omitted from this project (see also 14.2 Performance Goals).
3. It should be possible to manually remove and add agents during runtime. The feature will give the system some flexibility due to the fact that it will be possible to make runtime adjustments to the system (e.g. replace some agent with a new and maybe improved agent). This is considered to be an important feature both in the development of system and when the system e.g. has been released.
4. During runtime it should be possible to observe the status of the agents and the agent communication. This aspect should make error handling easier both in the development of the system and for the system administrators that might be responsible for the system after a possible release of the final product.
5. The communication between the agents should follow the KQML standard (See Appendix D). The arguments for using KQML can be seen in 3. Agents. It should only be mentioned here that KQML is a de facto standard in agent communication and that this language has been considered to fulfil the needs for agent communication in this project.
6. The task of developing new agents should be kept effortless in order to simplify the process of adding new facilities to the system. It is considered important that other developers than the project group should be able to add agents to the system in order to add functionality and change the system after a possible release.

All the functional requirements presented in the above two sections should work as a starting point be implemented in the inTelly system. Adjustments to the functional requirements can and will be made based on the test results during the iterative development. It will also be possible to make corrections to the lists if time constraints and/or other factors make it necessary.

The functionality will be tested and evaluated when the final system has been developed.

14. Testable Goals

In order to test and evaluate the usability and performance of the TV-guide there must be stated some testable goals. These goals will be used together with the functionality in order to test and evaluate the final version of the system. The testable goals consist of two types: Usability goals and Performance goals, which are presented in the following:

- **Usability goals:** This issue will present the requirements to the TV-guide in order to be a system that is useful in helping the users finding out what to see in TV. This will include measures concerning: errors, enjoyableness, effectiveness etc.
- **Performance goals:** These requirements will primary concern aspects, which are not included in any of the above requirements. E.g. "The system should be able to handle 10.000 simultaneous users" or "The system should be able to regenerate after a power loss".

The succeeding sections will present the details of the testable goals.

14.1 Usability Goals

As already indicated there are a number of measures that can be used to evaluate the usability of a system. The measures are presented before the description of the usability requirements in order to give an idea of the source of the usability requirements. The measures used in this project are the based on [JR] and [JN1]. The project group has combined the sources in order to create an extensive set of usability goals:

- Usefulness: Degree to which users can achieve their goals and which tasks can the user accomplish with the system?
- Effectiveness: How well the users can perform their tasks?
- Learning: How long will it take before a user reaches some defined level of confidence?
- Errors: What are the errors made, how often do they occur, how well does the system help the user recover from errors etc.
- Acceptability: How do the users subjectively rate the system?
- Enjoyableness: Is the system fun to use?

The detailed usability goals are presented in the table below. It should be noticed that nearly all the usability goals presented have a constraint saying that 70 percentages of the test participants most meet the criteria. This number is taken from [JR, p.45] and could naturally be discussed further. The arguments of [JR, p.45] is that he wants to perform a validation test that is "reasonably challenging" and at the same time perform a test that leaves "the design team some work to do before the product release", where the number is moved toward a 95 percent benchmark. The project group has decided to make a validation test that is based upon

the 70 percent benchmark, in order to evaluate the final version of the system as a version that is "almost" ready for release. It could be discussed whether there should be made two validation tests (70 percent and 95 percent benchmark respectively), but this has been omitted due to time constraints. Furthermore is has been considered less interesting to make two very similar tests and then perhaps be forced to leave out one of the earlier tests. The time constraints mentioned in the usability requirements are found by:

- Making similar tests on competing products. This will include both Internet TV-guides and traditional TV-guides in e.g. newspapers.
- Looking at the time taken in similar situations in the observation test.
- Discussing what would be a reasonable time constraint in the project group.

Measure	**Usability Goal**
Usefulness	*At least 70 percent of the first time users must be able to find out what to see in TV for one day without using the help facility of the system.* This task is considered to be the most important task for a user of a TV-guide to perform. Therefore it has been included in order to evaluate the usefulness of the system. Again the use of the help facility is used to measure the usefulness of this part of the TV-guide.
Usefulness	*At least 70 percent of the first time users must be able to create a user profile without using the help facility of the system.* It could be discussed whether this usability goal measures the usefulness of the system, considered that the user profile is an extra subtask added to the overall task of finding out what to see in TV. But the project group has found that the user profile is a necessary and a very important factor in the usefulness of the system due to the fact that it e.g. helps the user in the filtering of TV-programs. The use of the help facility is considered to indicate a lack in the design of the user profile.
Effectiveness	*The first time users should be able to find out what to see in TV for one day in maximum 3 minutes for 70 percent of the users.* As already pointed out this task is considered the most important task to be performed in the use of the TV-guide. It is therefore natural to measure and evaluate the effectiveness of the final system on this usability parameter. Again the 3 minutes have been found by trying out the task on other Internet TV-guides. The project group has performed the tasks and mean time of the results has been used as the demand for the inTelly system. It should be noticed that this constraint could show to be hard to fulfil, because the inTelly system is planned to contain more possibilities than the TV-guides evaluated to the find the mean time. But the project group has found that the extra features

	most not result in a less efficient use of the system.
Effectiveness	*The creation of a user profile should at maximum take 5 minutes for 70 percent of the users (first time users).* In order to evaluate the effectiveness of the system it is decided to include the generation of the user profile. The main argument is as already mentioned that the user profile is an important issue in getting the most out of the TV-guide. The 5 minutes have been found by measuring the mean time taken for generating the user profile on the competing products (performed by the project group). It could be argued whether the mean is a good measure for the inTelly system, but the user profiles in the competing products that was tested in the observation text has proved to be acceptable.
Learning	*The first time users should catch the primary concept of the new features in the system in a maximum of 30 minutes for 70 percent of the users. The new features could be functions like: the possibility of telling the system which programs the user likes and dislikes, etc.* This usability aspect is taken into account because one of the primary issues of a website is to present the concept to new users or they will typically never return to the website. The 30 minutes is the time that is reserved to each user in the final validation test. It could be questioned whether the 30 minutes is a long time or not, but the project group thinks that it is a suitable amount of time for the first time users to evaluate the concepts of the TV-guide. An argument for the relatively long time is that some of the features of the TV-guide are new and therefore not immediately recognisable for the users.
Errors	*The task of finding out what to see in TV for one day must maximum result in 1 error for 70 percent of the users.* One of the typical ways to measure the success of a website and other applications is to count and evaluate the number of errors that the users makes when performing typical tasks. In this situation the number of errors is counted and evaluated on the two primary tasks of the TV-guide: Finding out what to watch in TV and the generation of the user profile. The number of acceptable errors can be discussed, but the project group has regarded a maximum of 1 error as a satisfying result for the first time users.
Errors	*The user profile must be generated with a maximum of 1 error for 70 percent of the users (first time users).* See comments above.
Acceptability	*70 percent of the users should be neutral, agree or strongly agree upon the following statements on a Likert scale containing: strongly disagree, disagree, neutral, agree and strongly agree.* *1) The TV-guide is comfortable to use.*

	2) The TV-guide is intuitively to use. *3) The layout of the user interface is nice.* *4) The user interface is well arranged.* *5) It is easy to navigate in the TV-guide website.* *6) The TV-guide offers an effective way of finding out what to see in TV.* *7) The TV-guide helps me in remembering what to see in TV.* The statements presented cover a number of different aspects involving the user acceptability of the final system. The Likert scale is used because this scale makes it possible for the users to subjectively evaluate the system. It should be noticed that is it not possible to calculate an average score for each statement by transforming the scale to numbers (e.g. 1, 2, 3, 4, 5 or –2, -1, 0, 1, 2). The reason is that there is no correspondence between e.g. the term "strongly disagree" and the value "5" and it is not possible to say that there should be an equal interval between terms like "agree/neutral" and "strongly disagree/disagree". There is a possibility of numerically evaluate the results obtained from a Likert scale [Marketing Research, p.467-470], but they will not be used in this project. The evaluation of the results will be performed by calculating the percentage of users that are neutral, agrees or strongly agrees.
Enjoyableness	*70 percent of the users should be neutral, agree or strongly agree upon the following statement on a Likert scale containing: strongly disagree, disagree, neutral, agree and strongly agree:* *1) The TV-guide is fun to use.* See comments above.

14.2 Performance Goals

There are some of the system demands that do not immediately fall into the category of neither functional requirements nor usability goals. These additional goals are named "Performance goals" and they are:

Performance goal
1. The system priority must match the user priority with an accuracy of 90 percent. This performance goal is difficult to meet, but the project group has decided to keep this high number in order to indicate that there is a risk that an AI that once in a while misses to show the correct programs might be considered useless by the users of the system.
2. The TV-guide should be able to detect 90 percent of a week's repeating programs. As mentioned above there is a risk that users will choose another TV-guide if e.g. the inTelly system has misinformed the user and the user therefore misses some favourite Repeater.
3. The load time for one page in the inTelly website should be maximum 10 seconds at a

modem speed at 56kbit/s (one user).
This performance goal is considered important to fulfil because the users of the Internet in general easily becomes impatient The 10 seconds should be considered as a maximum and it would be nice if the final system results in a lower load time.

It could be discussed whether the above performance goals should be tested as usability goals (E.g. maximum load time). But it was decided to put these specifically measures in the performance goals due to the fact that it is possible to test these goals without necessarily involving users. Furthermore the performance goals will also be implicitly tested in the tests of the usability goals. It will appear in the usability measures (such as effectiveness, enjoyableness and acceptability) if e.g. the load time of the web pages is considered too long by the test participants.

It should be noticed that there is no performance goals considering scalability of the system. This means that this issue is omitted this project. The maximum load time is therefore tested with only one user. The scalability issue will be an important factor in a final product, but the project group has chosen to put focus elsewhere in the development of the system. The use of agents will make the scalability problem relatively easy to solve because the structure of the Agent framework makes it possible to distribute the load on different computers.

ITERATIVE DEVELOPMENT

15. Introduction

This part of the Main report will present the iterative development phase, which consists of three complete iteration cycles. It should be noticed that this section will not describe the details of the design – it will only concentrate on the user interfaces and the overall functionality implemented in the different versions of the system. The detailed presentation of the final system (including the Agent framework, personalisation etc.) will be handled in the Final Design. Each iteration will be presented by the use of the following sections:

- **Prototype**: In this section a sketch of the version will be presented. Only the most important aspects of the version will be presented. For a thoroughly description of the versions the reader is referred to Supplements M, N and O.
- **Test**: In this section the outline of the test performed on the system is presented.
- **Evaluation**: Finally an evaluation of the test is presented. This will include a short presentation of the test result. The evaluation will reflect upon the test. This means that if there are special comments to the test itself and/or to the circumstances about the test they will be described here.

It has been chosen to present the iterations chronologically in order to give an idea of how the principles of the User-Centred Design Method have been utilized in this project, which is considered to be an important part of the overall project. The project group is aware that this presentation of earlier versions of the system can give rise to confusion about the state of the final design. But by separating the early versions of the system and the final design the project group hope to be able to give a presentation of the development of the system in all stages of the project without unnecessarily confusing the reader.

As already mentioned there will be presented three iterations and corresponding system versions in this section. The first version will be named “0.1”, the second “0.2” and the third version of the system will be named “0.3”. In the presentation of the method (2. User-Centred Design) it can be seen which evaluation/test methods there are used in each of the iterations. A short repetition is given below:

- **Version 0.1**: At this early stage of the development it has been chosen to create several different user interfaces in order to evaluate their pros and cons. At this stage there will be no functionality implemented. It has been chosen to make a heuristic evaluation on the user interfaces where the test participants consist of the project group members. It has been considered acceptable to use the group members in this evaluation due to the fact that other usability experts will perform the heuristic evaluation on version 0.3 (See also the Test report). Version 0.1 is based on the Functional Requirements and the Testable Goals developed from the results of the exploratory tests performed in the Initial Design Phase, project group ideas and inspiration from other sources (e.g. competing products).

- **Version 0.2**: From the former heuristic evaluation there has been made a version of the system that contains some of the primary functionality in order to let external test participants perform an assessment test trying out the different parts of the system. At this stage of the development the system begins to take form, but it is still possible to make both major and minor corrections to the design. It is important that it is possible and relatively simple to make corrections to the design after this test due to the fact that it is the first time that new users tryout the system. The users will be employees from Mindpass A/S.
- **Version 0.3**: The result of the assessment test is used to create an almost complete version of the system. The test used at this stage is a heuristic evaluation performed by usability experts. All functionality will be implemented before the evaluation and it is considered important that the test participants have a complete version of the system to evaluate. It is important because this will be the last evaluation of the user interfaces and functions before the development of the final version of the system.

The three iterations will be described in the following chapters.

16. inTelly – Version 0.1

Version 0.1 of the inTelly system will be presented in this chapter. This version consists of mock-ups of the user interface. The user interface will be evaluated by the use of a heuristic evaluation performed by the project group. The main sources for the preliminary user interface are the considerations that are collected in the chapters 13. Functional Requirements and 14. Testable Goals.

The user interfaces will not be presented in detail in this chapter – only the most important aspects of them will be discussed here. For a complete presentation of the user interfaces in version 0.1 see Supplement M.

16.1 Prototype

The user interface consists of several different main parts:

- Program list
- Detailed description
- Dialogue agent
- User profile
- Help

Further more there will be some additional user interface elements:

- Menu bar and logo
- Messages

In the following sections there will be a short presentation of each of the above user interface contents.

Program List

There has been developed several suggestions to the presentation of the TV-programs. In this section the primary considerations about the program list will be presented.

Two of the primary issues in presenting the TV-programs are to minimize the length of the program list and make it well arranged. It has been found that filters that make it possible for the user to filter the TV-programs is one part of the solution. Further more the test participants in the observation test asked for some sort of notepad where interesting TV-programs could be placed. Finally the project group has decided to personalise the TV-guide in order to present and filter out TV-programs according to the user's needs. This will include that the user should be able to prioritise some TV-programs: telling the system, which programs the user likes and dislikes. It has been chosen to tryout the possibilities of integrating the

prioritising in the daily use of the TV-guide: e.g. the system could regard a program that the user has moved to the notepad as a program that the user would like to see etc. Another possibility could be to identify the programs that the user likes and dislikes by presenting some typical TV-programs in the user profile. The user would then have the possibility of prioritising the programs given the system some idea of the type of programs that the user likes and dislikes. This solution has been decided to omit from the design, because this method typically does not catch the dynamics of a user's interests (4. User Profiles): The user will typically only prioritising the TV-programs once (when creating the user profile) and then never return to this page. If the system on the other hand is informed about the user's needs every time he or she is using the TV-guide, then it is more likely that the system can prioritise the TV-programs according to the user's current needs. This also means that the user profile will be easier to fill out, which has showed to be an important issue (see also Test report – Test result Observation).

The above considerations have given rise to different solutions. In the preliminary user interface there are examples of both integrated and separated notepads and in one of the solutions a notepad is placed next to the TV-program listing (the examples can be seen in Supplement M). The user evaluation of the TV-programs was the in preliminary user interface typically placed next to the TV-program in the listing. Various indications of the different types of evaluations were developed using icons, radio buttons and colours. The suggested filters in this version has been chosen to be select boxes due to the fact that this solution has achieved good results in the Observation test. The filters are used to filter the TV-programs available by defining the following classes:

- Day of program (Monday, Tuesday etc.).
- Time of day (e.g. 18:00-24:00, etc.)
- Channel (DR1, TV2, User defined channels, etc.)
- Category (Film, Documentary, User defined categories, etc.)

There has been added a feature called "Advanced" to each of the roll down menus in the filter. This feature is meant to give the user the possibility of defining his or hers own filters. One example could be a user-defined time of day.

When the user has evaluated some programs the system should use this information to make a personalised program list for the user in the future. This could be done by the system hiding programs that are considered of no interest to the user. This means that there should be a clear indication in the program list telling the user that some of the programs are hidden. Furthermore it should be possible for the user to select to see the hidden TV-programs. One of the layouts for the program list in version 0.1 can be seen on Figure 8.

Figure 8 : Example of the program list in version 0.1.

As it can be seen the filters are placed in the left side of the page and some additional functions are placed below them (such as show hidden programs). The program list is placed in the centre of the user interface and there are two prioritising possible in this user interface (indicated by two icons). A notepad has been placed to the right of the program list and the user can immediately see the programs selected. If the user clicks the "√"-icon in the program list then the program is meant also to be showed in notepad and the "X" will hide the program from the list. If the "X" is clicked on a program in the notepad then the program is deleted from the notepad.

Detailed Description

There has been considered several ideas of presenting the detailed description of the TV-programs, which mainly consists of a text describing the TV-program. It is not a simple task due to the fact that there can be many TV-programs present in the program list, which can result in information overload if the detailed description is an integrated part of the program list and if shown for all TV-programs. Keeping that in mind it has been decided only to present one detailed description at a time. This description could be placed in a separate frame in the program list or it could have a separate page in the website. Both methods are tried out in version 0.1. There have also been made a user interface where it is possible for the user to select to see a small part of the detailed description (called short description) for each program in the program list (e.g. 100 characters for each description).

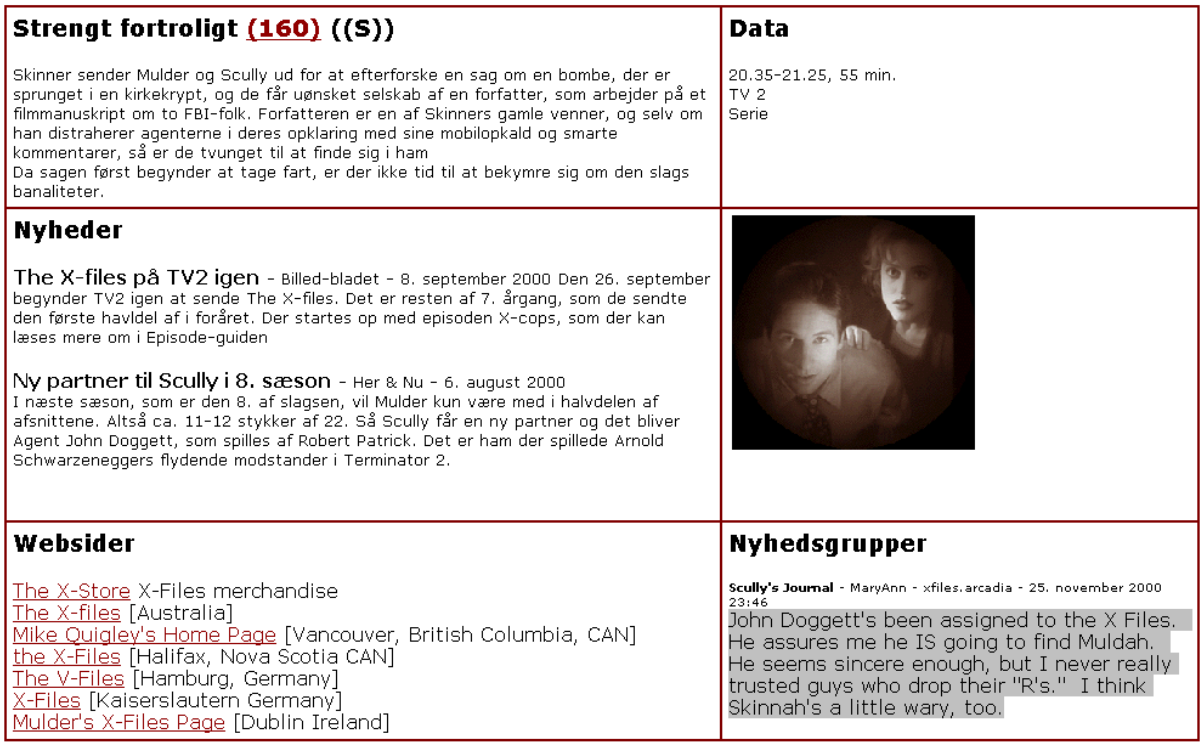

Figure 9 : Detailed description of a TV-program version 0.1 showed on a separate page.

In Figure 9 the outline for a detailed program description can be seen. The user interface consists of six boxes containing different information for an easy overview. Each box has as a heading except the picture box. The contents are:

- Detailed description text (Heading: ”Strengt fortroligt”).
- Program data such as time of day, channel, category etc. (Heading: “Data”).
- News related to the TV-program (Heading: ”Nyheder”).
- A picture corresponding to the TV-program.
- Links to related websites (Heading: “Websider”).
- Links to related newsgroups (Heading: “Nyhedsgrupper”).

Dialogue Agent

The dialogue agent was meant as an alternative suggestion to the more traditional TV-guides. The user could e.g. find out what to see in TV or change the user profile by answering a number of questions asked by an agent. The dialogue agent user interface is made as a suggested list of questions to be asked by the agent to clarify what the user wants and perform the corresponding action(s).

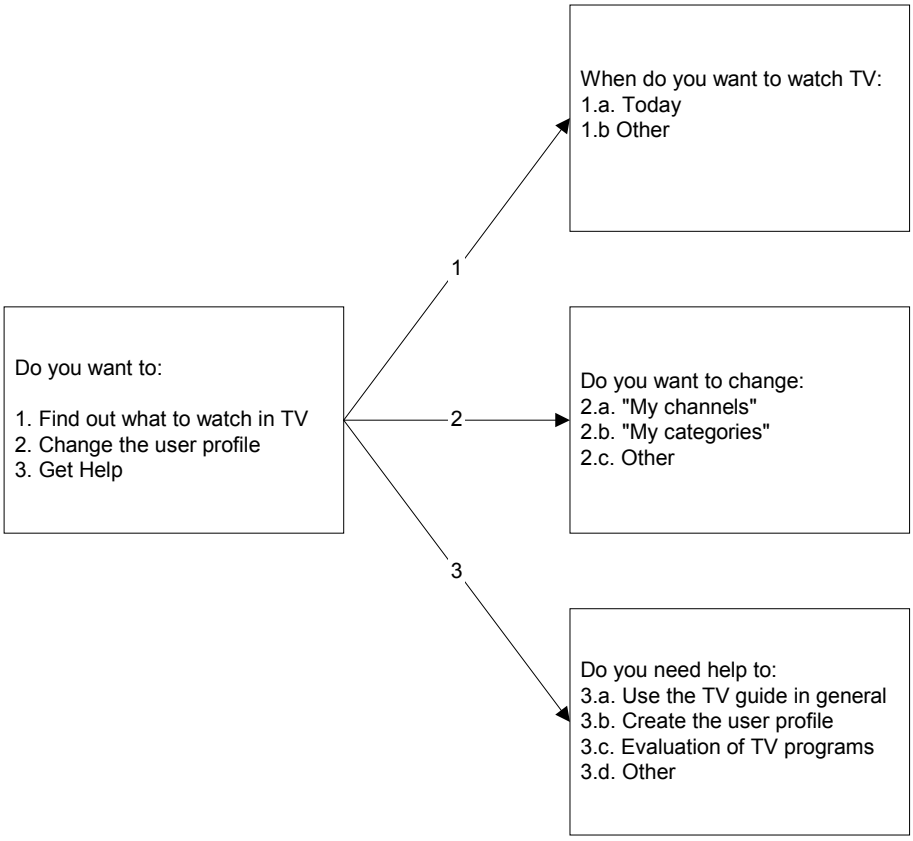

Figure 10 : Dialogue graph – example from version 0.1.

Figure 10 shows a draft of a simple dialogue with different possibilities. The figure is only considered to be an example of how the dialogue could be implemented. There are several options and possibilities to be considered in an implemented version, such as:

- A complete analysis of how the dialogue tree should be structured.
- Adaptive dialogue that gradually changes according to the user's needs.
- How the user should interact with the dialogue agent.
- How to present the dialogue to the user (e.g. graphical agent).

User Profile

In the preliminary user interface there is presented one solution to a user profile interface. The solution consists of six pages:

1. **Personal data**: In this page the user can enter data such as: name, email, etc. The amount of personal data will depend on the future facilities of the system: It would not be relevant to enter the email-address if the system never uses this information.
2. **Channels**: Here the user selects which channels he or she want to be shown when the user-defined channels are selected in the filters.
3. **Categories**: The user can select which categories he or she would like to watch. The information is used when the user selects the corresponding option in the filters.
4. **Repeaters**: This page makes it possible for the user to select which repeaters he or she is watching.

5. **Keywords**: If the user has special interests that he or she would like the system to look for in the TV-programs the corresponding keywords should be entered here.
6. **TV-guide set-up**: This page gives the user the possibility of defining which columns that should be present in the TV-program listing.

One of the pages in the user profile can be seen in Figure 11.

Figure 11 : Categories page of the user profile – the user can select individual categories. Version 0.1.

There are placed navigation buttons at the top and bottom of the page, where the user can perform the following: Go back to previous page in the user profile, Go forward to the next page in the user profile, Save and Reset the user profile.

Help

The help facilities of the system should, according to the User-Centred Design principles, be updated as the system development advances. In this preliminary user interface there is no functionality implemented and furthermore there are typically many different solutions to each part of the system. This means that it will be a huge task to develop help facilities to all the different solutions and possible implemented functionality. It has therefore been chosen to make a draft of the help system (which can be seen in Supplement M), which will be further specified in the next version of the system.

Menu Bar and Logo

It has been chosen to make a logo and menu bar at the top of every page in the inTelly website. This choice should serve mainly two issues:

1. The user can easily identify pages that belongs to the website when he or she knows the logo.

2. The menu bar ensures that the user can get to any of the main pages of the system at any state in the use of the inTelly website.

Both issues should ensure consistency and provide clear exits. The first layout of the menu bar and logo can be seen in Figure 12.

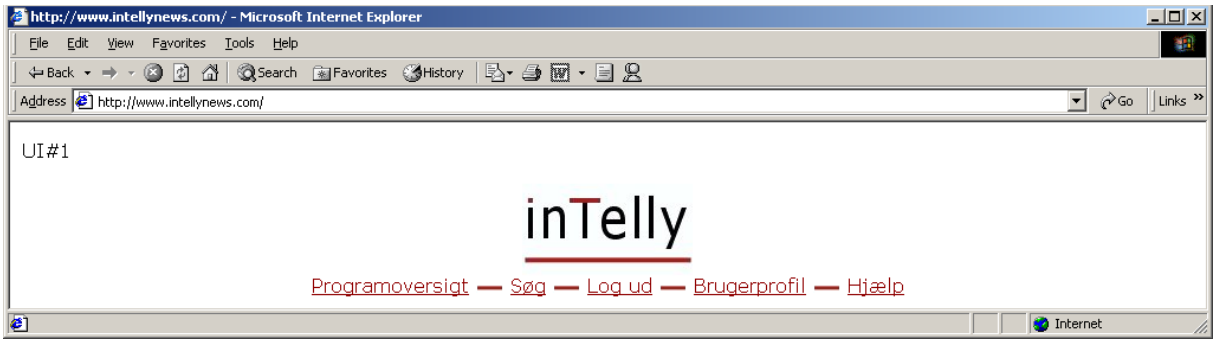

Figure 12 : Logo and menu bar version 0.1.

Messages

There has been made a draft of a message box in the preliminary user interface. This facility should present relevant information and get feedback from the user during the use of the system. The project group is aware that it should be possible for the user to ignore the messages and continue working, which means that the messages must not "steal focus" (the user does not have to press e.g. a button to continue).

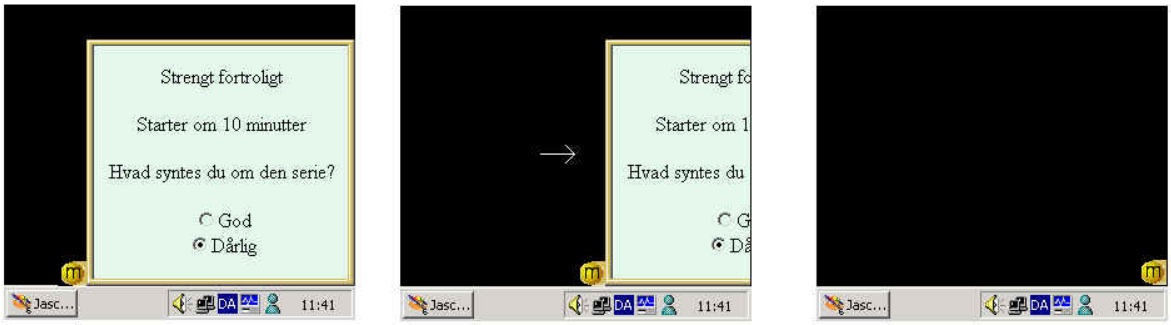

Figure 13 : Example of a message box (version 0.1).

In Figure 13 an example of a message box is presented. The idea is that the message disappears after a number of seconds and it will be possible to see the last message by clicking on the little "m".

16.2 Test

This heuristic evaluation performed on version 0.1 is based on the Test report - Test plan Heuristic evaluation I. The test plan describes how to perform the test and what outcome there might be expected.

Jacob Nielsen originally presented this heuristic evaluation method (see also Appendix B and C), which is used in this test. The method is used to determine usability problems in the user interface so that these can be taken into account in the development of the next prototype.

The test participants evaluate version 0.1 according to the usability heuristics given below (described in detail in Appendix B):

- Simple and natural dialogue
- Speak the users' language
- Minimize the users' memory load
- Consistency
- Feedback
- Clearly marked exits
- Shortcuts
- Good error messages
- Prevent errors
- Help and documentation

The test participants evaluate the user interfaces individually. The results from each test participant are presented as a written document. To ensure that the documents are somewhat easy to evaluate and to ensure that all the usability heuristics has been evaluated there has been made an evaluation scheme to be filled out by the test participants (for an example see Test report - Test plan Heuristic evaluation I).

A new scheme containing all the usability problems found by the test participants is made when the first schemes are filled-out. The problems found is then rated. The rating is a method that is based on a "Severity Rating" given by [JN3]. Sending out all the usability problems to the test participants, asking them to evaluate each usability problem according to the scale below, performs the rating of all the usability problems:

0 = I do not agree that this is a usability problem at all.

1 = Cosmetic problem only: need not to be fixed unless extra time is available on the project.

2 = Minor usability problem: fixing this should be given low priority.

3 = Major usability problem: important to fix, so should be given high priority.

4 = Usability catastrophe: imperative to fix this before product can be released.

When the ratings are returned the average is found for each usability problem and they are sorted accordingly. It is now possible to determine which problems that are considered most important to correct.

16.3 Evaluation

This section will present the test results of the heuristic evaluation and make a conclusion on this version of the system.

The results presented below will concentrate on the major problems, to draw out the most severe problems in the user interfaces and to focus the attention on these problems. Problems are considered as major problems if these are rated with an average of 3 or higher (will typically indicate that all test participants evaluates the problem as a major problem or higher). The test participants agree quite much on the rating of the usability problems, which indicates that the evaluation result is quite thrust worthy.

From the immediate results it is obvious that the user interface with most usability problems is the program list. When looking a bit deeper into these results there are discovered three major usability problems in the user interface, and these are considered to be:

- The presentation and interaction in connection with prioritising programs.
- The interactions with the notepad.
- The help concerning why programs are hidden and the help concerning prioritising of programs.

Prioritising of Programs

A major usability problem in the system is to find out how to prioritise the programs in an efficient and intuitive way. Several different degrees of evaluation for both the user and the system are also proposed.

Interaction with Notepad

The evaluation made it clear that there cannot only be a separate page with a notepad. There has to be an indication (at the front page) of which programs has been added to the notepad, which gives the following different ways of implementing the notepad:

- A combination of a frame with a notepad besides the program list and a separate page containing a notepad.
- A combination of a notepad that is integrated in the program list and a separate page containing a notepad.
- A notepad that is integrated in the program list.

Help for Hidden Programs and Prioritising of Programs

It is not intuitive that the system hides some programs from the user, therefore the system should provide help for the user concerning why some programs are hidden. This help should only be provided for the first time users, because when the user knows why some programs

are hidden, it seems logically. The evaluation of the programs should also be assisted by the system, because this kind of evaluation is also not known by the user (e.g. from other Internet TV-guides), and maybe difficult to understand. This should be tested with new users when the help and the functionality are implemented, to see if it is comprehendible.

16.4 Conclusion

The evaluation of the draft user interfaces made the group become aware of the problems in version 0.1 of the system and it also gave a good idea of what to be attentive to when developing these further.

This test gave the impression that there is the need for a supplement to the heuristics from Jacob Nielsen at this early state of the development. This supplement should concern new or missing functionalities, which the user might find necessary during the evaluation.

The next versions of the system will make use of the results obtained in this iteration.

17. inTelly – Version 0.2

In this chapter version 0.2 of the inTelly system will be presented. Contrary to version 0.1 this version will only consist of one user interface layout, but also the primary functionality will be implemented. The design of the user interface will be made using the same sources as version 0.1 plus the result of the heuristic evaluation performed on version 0.1. A fully presentation of the user interfaces in version 0.2 can be seen in Supplement N.

17.1 Prototype

The prototype will nearly contain the same main parts as presented in version 0.1. The changes are that the dialogue agent is omitted and that the messages are not implemented in this version. The dialogue agent is left out because the project group has considered that this method of using a TV-guide will not be appropriate. The main reason is that it is considered more difficult and complicated for the user to go through a dialogue in the daily use of the TV-guide. In special situations the dialogue agent could be helpful and the idea will therefore be kept in mind in the future versions of the system. A possible use for the dialogue agent could be in help situations and in creation of user profiles where the user typically is in a situation where he or she does not know what to do. The messages will not be implemented in this version, but will be considered in the next version of the system (0.3).

There has been implemented some functionality in version 0.2. The primary functionalities implemented are:

- Create user.
- Login.
- Most important part of the user profile (channels and categories).
- User prioritising of programs.
- Simple version of the priorities of the system based upon the user evaluations. This functionality is implemented to make it possible for the test participants to comment on the non-standard functions presented in the inTelly system. The system priorities will be based upon earlier user evaluations in order to put a high system priority on the programs that the user is considered to like and a low system priority on programs that the user is considered not to see.
- The integrated notepad is also implemented, this includes an automatic sorting of the program list according to the user prioritising: “Want to see” and “Maybe want to see” will be placed on the top of the listing and “Do not want to see” is placed in the bottom of the program listing (hidden programs). It is also possible for the user to sort the programs list according to the content of one column by clicking on the corresponding column header.

The following important functionality is not implemented in version 0.2, but will be in future versions:

- Repeaters (e.g. serials and news).
- Search.
- Detailed description is only implemented with a static page.
- Messages.
- Advanced filter option.

In the following sections there will be a presentation of the most important aspects of version 0.2. It should be noticed that some user interfaces are omitted from the presentation below if there is no or only minor adjustments to the design compared to version 0.1.

Program List

The program list can be seen in Figure 14. As it can be seen it has been selected to make an integrated notepad in this version. There were several solutions to the notepad problem, but it was decided to make it integrated in the program list for the following reasons:

- The integration of the notepad is considered by the project group to be somewhat immediately understandable by the users of the system. This consideration will be tested in the following assessment test.
- The use of more than one web page to present the TV-programs gives the risk of confusing the user.
- The integration of the notepad means that the process of evaluating programs and updating the notepad is performed in the same act minimizing the user's work.
- The user will typically want the same information about a program nevertheless the program is in the program list or the notepad. This means that it would be natural to make the looks of the notepad and the program list identical, which is the situation when the notepad is integrated in the program list.

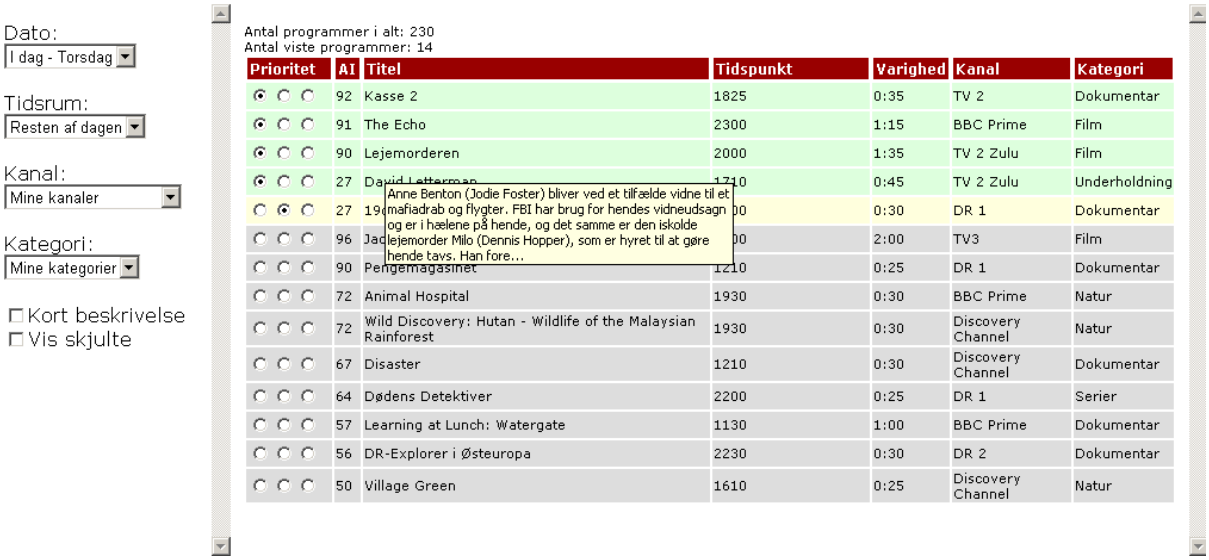

Figure 14 : Program list version 0.2.

The first column in the program list is named Priority (Danish: "Prioritet") and consists of three radio buttons for each program. The buttons are used both to evaluate the programs and to control the notepad. The indication of the user-evaluated programs is both indicated by a colour (green = "Want to see", yellow = "Maybe want to see" and red = "Do not want to see") and by a corresponding radio button. The three degrees of evaluations are chosen because the project group has considered them appropriate and they will naturally be reconsidered if the tests show any problems related to this decision. The second column named "AI" is a rating of the programs given by the system based upon earlier evaluations made by the user. The scale goes from negative to positive (0-100).

A typical example of what happens in the program list during the use is given in the following list:

- The user wants to see some program and want to place it in the notepad.
- The user clicks on the leftmost radio button ("Want to see").
- The program jumps to the top of the program list and the colour is changed from grey to green. A black dot will also mark the corresponding radio button.
- The user do not want to see some other program and want to remove it from the program list.
- The user clicks on the rightmost radio button ("Do not want to see").
- The program jumps to the bottom of the program list and the colour is changed from grey to red. A black dot will also mark the corresponding radio button. The programs is now a part of the "hidden programs" that only will be shown if the corresponding check box is selected (placed below the filter menu).

The situation is similar if the user prioritises the programs as "Maybe want to see" (middle radio button). In this case the program is also placed in top of the program list (notepad) under

the programs evaluated as “Want to see”. This situation is also illustrated in Figure 14 (there is one program in the list that has been evaluated as “Maybe want to see”).

There has only been used a mouse over description of the different priority possibilities (the three radio buttons) and no icons. The reason is that the use of icons might disturb the well-arranged program list and it has been considered that the user easily pick up the idea of the radio buttons – it is considered to be only a matter of having tried out the system once before the principle is intuitively. This type of decision will always be difficult and it will typically have to be a compromise between satisfying the first time user and the everyday user. In this case it has been tried to make a solution that both users will find appropriate.

The rest of the columns represent more traditional TV-data such as title, time of day, duration, etc. These have been chosen because they are considered important when a user wants to find out what to see in TV. As already mentioned it is possible to sort the program list after the contents of all the different columns by clicking on the header of the different columns.

The filters to the left of the program list are kept similar to the outline in version 0.1 and their usefulness will be tested in the assessment test. It should be noticed that the advanced features in the filters are not implemented in version 0.2.

Detailed Description

It has been chosen to use several of the ideas presented in version 0.1 in the design of version 0.2. First of all it is possible to get a short description shown as a part of the program list. Another possibility is to click on the title of a program and the user is sent to a separate page containing the complete description of one program. Additionally it has been chosen to try out a new method for showing the short description by the use of mouse-over (an example can be seen in Figure 14). This method is considered to give the user an easy way of getting an overview of the TV-programs in the list, without having to go to the detailed description for each program or having to select to see a short description of all the programs, which makes the program list longer and less well-arranged. It has been chosen not to implement a detailed description in a separate frame in the program list, because this is considered to take up too much space on the page and also making the program list more confusing to read.

User Profile

The user profile had several usability problems in version 0.1 (see Test report - Test result Heuristic evaluation I). It was difficult to navigate around in the creation of the user profile, because there was no navigation menu in the design. This navigation has been implemented in the user profile in version 0.2, by adding a separate frame containing a menu presenting the

different items in the user profile. The user profile page containing “My categories” is shown in Figure 15.

Figure 15 : User profile page example version 0.2. It should be noticed that the screen dump only contains a part of the total menu items that will be present in a final version.

Help

The help pages available have been updated according to the new version of the system. Further more there has been implemented a navigation menu similar to the one in the user profile described above. This should ensure that the users in the assessment test are able to get help if needed in an appropriate and logically way. The first page of the help pages in the system is shown in Figure 16.

Figure 16 : Help front-page version 0.2.

17.2 Test

This section presents the assessment test performed on the inTelly version 0.2. For a complete description of the test plan and test result the reader should see the Test report: Test plan - Assessment test and Test result - Assessment test.

The objective of the assessment test is to determine the strengths and weaknesses of version 0.2. This will include the user interfaces, implemented and non-implemented functionality. The non-implemented functions will be presented to the participants by the test monitors in order to get a feedback on these issues where appropriate. The test participant will be asked to perform some tasks on the inTelly system and think aloud during the use. Furthermore the test should focus on the "new" facilities presented in the inTelly system (integrated notepad, user priority and system priority (AI) etc.).

The method used will be to ask the participants to:

- Solve some given task on the system.
- State his/hers opinion about the features of the system (primary the new features presented in the inTelly system).
- Describe how some tasks could be performed by the use of intuition.
- Describe the overall experience with the system.

The tasks to be performed by the test participants are the following:

1. State your opinion about the inTelly start page (program list, where the user is not logged in, which means that the user evaluation and system priorities are not available).
2. Try out the system without logging in.
3. Try to make an account on the inTelly system.
4. Describe the different columns in the program list after the creation of the account. It should be noticed that this is the first time the test participant set eyes at the new facilities (such as user evaluation, integrated notepad, system priorities (AI) etc).
5. Try to evaluate some TV-programs in order to get a feeling about the notepad. After the evaluation the participant is asked to go look at tomorrows program in order to see the effect and works of the system priorities.
6. The user should have tried the different filters, sorting facilities and the detailed description in the test. If not he or she is explicitly asked to tryout the functions.

Five employees from Mindpass A/S will perform the test. It is possible for the test monitor to help the test participants if this is considered appropriate. It is also allowed for the test monitor to ask the participants to comment on the actions he or she is performing during the test in order to maximize the outcome of the test.

The evaluation measures will be the comments given by the test participants during the test.

17.3 Evaluation

In this section an extract of the most valuable results of the assessment test is presented. For a more thoroughly list of results see the Test report (Test result - Assessment test). In this document the immediate results containing all comments is presented.

The most interesting results are:

- The works of the prioritising, integrated notepad and the system priorities is not immediately intuitive. But it was observed that the users did catch the ideas with a minimum of information from the test monitors.
- There is missing some feedback to the user in a number of situations (like when a user has been created, when a user has prioritised a program, etc.)
- There is generally too much text presented to the user, which they typically did not bother to read (e.g. in the help interface and in the user profile interface).
- Some of the participants were missing a wizard that could guide them through e.g. the generation of a user profile.
- There was a demand for an advanced filter setting.
- The three degrees of prioritising TV-programs (“Want to see”, “Maybe want to see” and “Do not want to see”) was considered to be appropriate.

17.4 Conclusion

The project group has become aware of many of the weaknesses and strengths of the inTelly system version 0.2.

One of the major objectives in the test was to test how the first time users did interpret the prioritising, the integrated notepad and the system priorities (AI). In the test these aspects showed to give rise to problems, but it was, as already mentioned, easy for the test monitors to explain the principles and the participants did catch the ideas quickly. The project group therefore thinks that it will be possible to insert a short description in the message box, which has not been implemented at this stage of the development. The test pointed out that the messages should be kept simple and short in order for the users to bother to read them.

Another important and similar result of the test was that the participants missed some feedback in several situations. One solution to this problem could be to implement the already mentioned message box, in which feedback could be given to the user when needed.

The participants did not bother to read much text when they needed help or when creating a user profile for the first time. This indicates that the user interface in both situations could need a work over and the explaining text should be structured differently and unnecessary text should be removed.

A wizard facility should also be considered in the next version of the system. The wizard was primary missing in the generation of the user profile.

The advanced filter options were missing according to some of the participants. The project group in both version 0.1 and 0.2 considered this feature, but it has not been implemented in any of the versions. The result indicates that this feature should be implemented in the next version of the inTelly system.

Finally a comment about the test itself: There were during the test several valuable discussions with the test participants about the system concerning both implemented and non-implemented functionality. This was not mentioned in the test plan and therefore could give rise to a question about whether this was a good idea. But it was considered that the discussions gave the project group some good inputs to improve the design in future versions. The conclusion is that the possibility of having a discussion should have been mentioned in the test plan in order to plan for it, because the typical duration of each test was increased form the planned 30 minutes to about 45 minutes for each participant.

18. inTelly – Version 0.3

This chapter will present the inTelly version 0.3. This version will be the first version where all functionality is implemented. At this stage of the development the design should at some level be at a final state. There will be performed a heuristic evaluation on this user interface, but this time it will be performed by usability experts from Mindpass A/S. The final design (version 1.0) will be based upon the results of the heuristic evaluation.

18.1 Prototype

As already mentioned this prototype will contain all the functionality that has been planned. This means that there are the following major additions compared to version 0.2:

- The complete user profile has been implemented.
- The Repeater list has been implemented.
- The search functionality has been implemented.
- Detailed description of the TV-programs has been implemented.
- There are added messages giving the user feedback and the messages are also used as a wizard in the creation of a user profile.
- The advanced filter option has been implemented.

In the following sections there will be a presentation of the most important user interfaces in version 0.3.

Program List

The program list in version 0.3 is somewhat similar to version 0.2. Therefore it has been chosen to show a screen dump of the program list that points out the main differences between the two versions (see Figure 17). In the figure the short description (Danish: “Kort beskrivelse”) has been selected and it can be seen that there has been added some additional information to the program list in this situation. It has been chosen to insert a graph that represents the start time and duration of each program. This feature was requested in the assessment test performed on version 0.2 (see also Test result - Assessment test). The principle of the time graph is explained in a pop-up message that appears on a mouse-over event. It should be noticed that if the user deselects the short description the program list would be almost identical to the program list in version 0.2 (see Figure 14).

Figure 17 : Program list – version 0.3. The short description has been selected in the screen shot. If the user deselects the short description the user interface is more or less identical to the program list in version 0.2 (Figure 14).

Detailed Description

The detailed description has a completely new user interface. The main reason is that is has been decided not to include the different information channels (related news, newsgroups, etc.) in this project due to time constraints. Furthermore it has been made possible for the user to also evaluate the TV-programs in this user interface, because it has been considered that user typically will decide whether he or she wants to see the program or not when looking at the detailed description. An example of the detailed description can be seen in Figure 18.

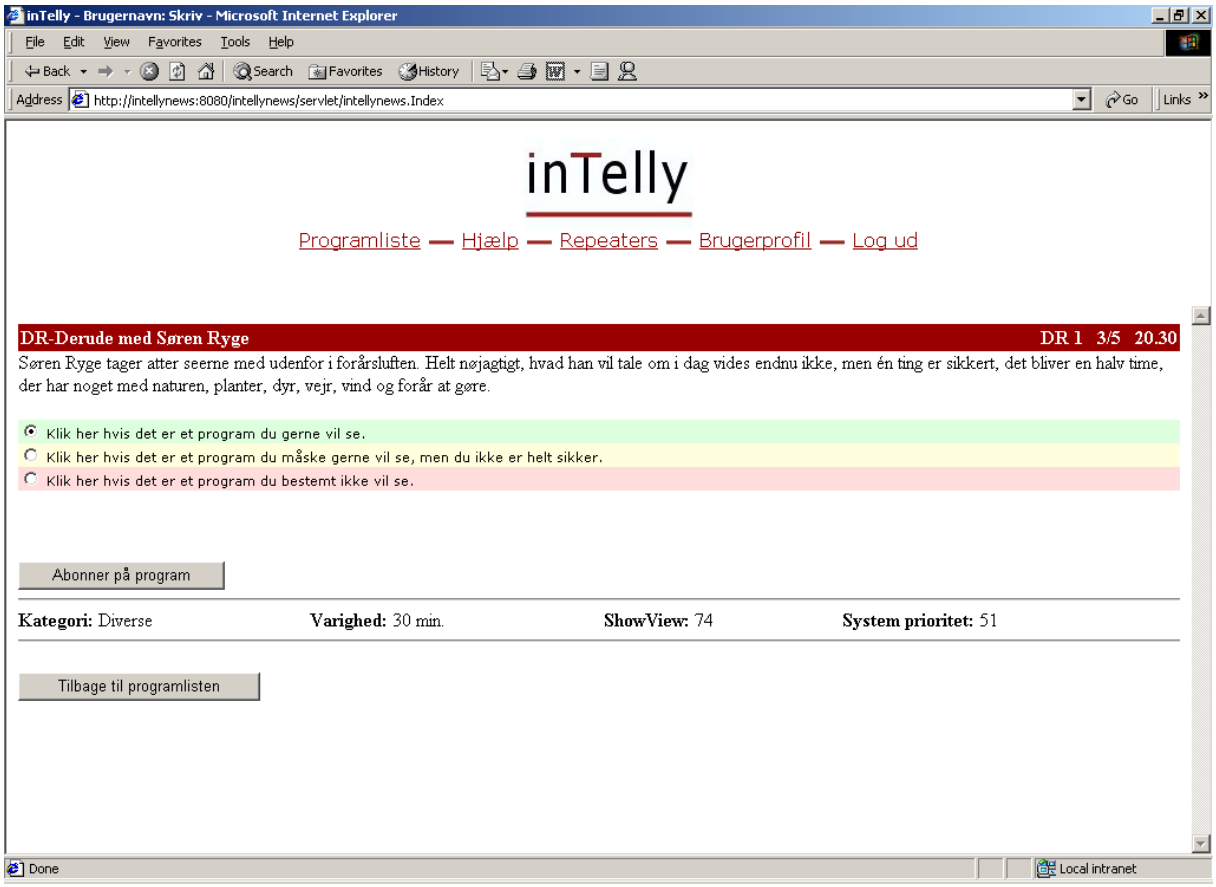

Figure 18 : Detailed description version 0.3.

There is added a possibility of subscribing on a program so that is will automatically be put in the notepad every time it is sent. This facility is only available if the program is detected as a Repeater by the system.

Help

The first page in the inTelly help is showed in Figure 19. There has been removed some text from the pages in the help facility. The contents have been updated in correspondence with the changes made on the system. There has been added a help-navigation picture, which shows a small image of the program list. This feature is made to help the user finding the relevant help page: The user can click on the part of the image that he or she might require help to. E.g. if the user needs help to the message box then he or she should click on the message box on the image, etc.

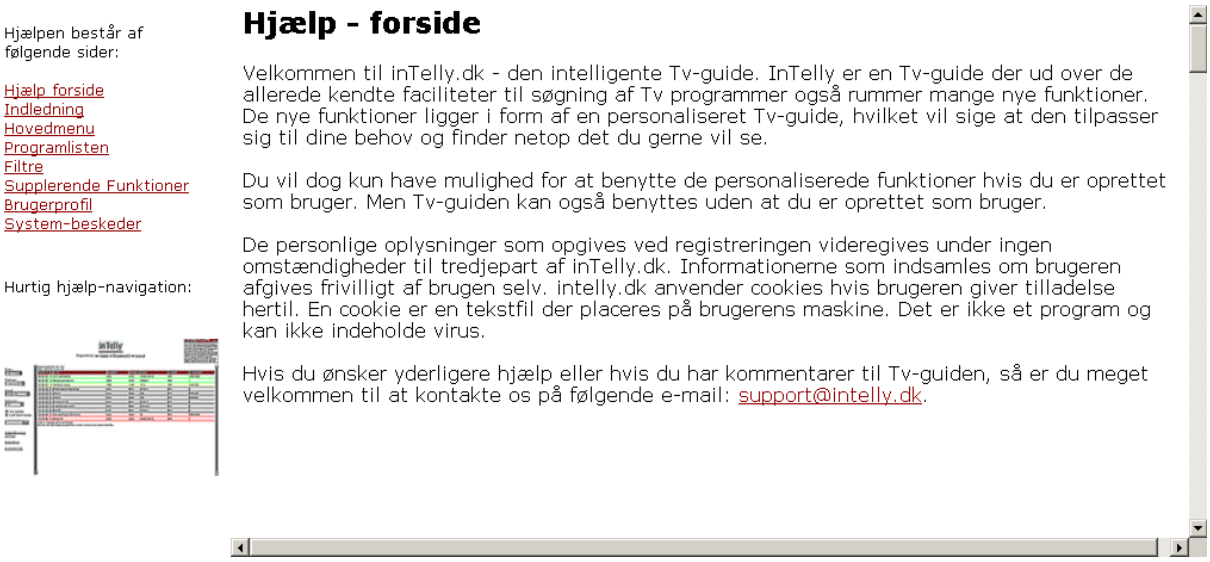

Figure 19 : The help front page version 0.3.

It should be noticed that the user still has the possibility of choosing the appropriate help page by selecting the corresponding menu item.

Repeaters

The Repeaters list is somewhat similar to the program list. An example of the Repeater list can be seen in Figure 20. If the user selects to see a Repeater the system will automatically put the program into the user's notepad every time the program is detected by the system. The principle is somewhat similar to the original program list, but the effect of prioritising a Repeater is different compared to prioritise a program in the Program list. In the Repeater list the programs are not sorted automatically according to the user priorities (are sorted when the user clicks a button). In the Program list the programs are sorted automatically.

Figure 20 : An example of a Repeater list in version 0.3.

It has been chosen not to include the possibility of getting a detailed description of each Repeater. This has been omitted due to the fact that this description will change over time.

Messages

The message box is used to give feedback to the user and to guide the user through the creation of a user. The message box is only present on the user interface when needed and when it appears it does not "steal focus", which means that the user can completely ignore the message box. When e.g. a feedback is needed the message box smoothly moves in at the top right corner of the user interface (next to the logo and main menu). The message box is placed here because this part of the user interface is unused and it is considered important that the box does not hide any information on the original user interface. After 10 seconds the messages box disappears. The amount of time is believed appropriate in order for the user to notify the message box and read the text in the box. An example of the message box can be seen in Figure 21, where the message indicates both the current step and the remaining steps in the creation of a user.

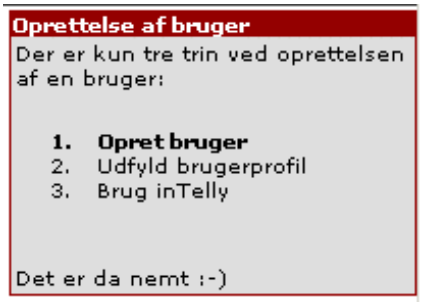

Figure 21 : Message box example in version 0.3.

Advanced Filter

The advanced filter has been planned implemented since the first version of the system. The advanced filter consists of checkboxes as in the user profile. The advanced filter can be selected by choosing an item named "Advanced" in the filter menu next to the program list. If the advanced time option has been chosen. The user has the possibility of defining a personal timeslot, which e.g. could be a typical timeslot that he or she is watching TV.

18.2 Test

The test plan is based upon the Test report – Test plan Heuristic Evaluation II.

Due to the fact that there has already been performed one heuristic evaluation and there are many similarities between this heuristic evaluation and the first heuristic evaluation. The reader should see Test plan Heuristic Evaluation II for a presentation of the test plan. In this section the differences will only be presented.

Usability experts from Mindpass A/S perform this second heuristic evaluation (not involved in the development of the TV-guide). This means that the participants are considered to be objective in the evaluation. This would probably not be the situation if the project group at this late state should perform the evaluation, because the project group would be too involved in the design of the different user interfaces and would therefore tend to overlook usability problems.

A minor difference from the first heuristic evaluation is that it is possible for the participants to suggest new or missing functionality during the evaluation.

18.3 Evaluation

This section will first present the result of the heuristic evaluation and second present the conclusion on the result and the test itself.

The complete result of the second heuristic evaluation can be seen in Test result – Heuristic Evaluation II this will include a list of usability problems ordered by the average of the severity rating given by the test participants.

The major usability problems discovered by the test participants are:

- The navigation on the website gave rise to problems. This was primary concerning the browser's back button, which did not always bring the user back to the last visited page and the navigation in the user profile was inadequately.
- The help facility was not meeting the user's needs and there is still too much text in the help pages.
- The user interface that presents the Repeaters detected by the system was not working as the similar program list presented in the normal program list. This seemed disturbing to the participants because the two lists are almost identical and therefore it was expected that the worked in the same way.

The complete list of usability problems will be considered when designing the final version of the system. In the Test result – Heuristic Evaluation II there is a table of actions concerning each usability problem. The most important actions and corrections will also be presented in the Final Design.

18.4 Conclusion

The heuristic evaluation has again proved its value. The list of usability problems contained about 60 problems that are considered relevant to correct or adjust in the final version of the

system. The advantage of using external experts is that they are considered to be more objective and will probably look at the user interfaces in a different angle.

The design of the final version of the system will use the results obtained in this evaluation and naturally keep the previous results achieved in the former tests in mind.

FINAL DESIGN – VERSION 1.0

19. Introduction

This part of the Main report describes the final system, both the User interface and the design and implementation of the final system (version 1.0). The final User interface is a result of the iterative development phase. The design and implementation of the system has been going on in parallel with developing the User interface.

The User interface of the system is described in the second chapter of this part of the Main report. By reading this, it should be possible for the reader to understand how the system works, without reading the description of the Iterative development phase in the preceding part. Although there will be made lots of references to the arguments on why it is made the way it is, these arguments are placed in the chapter Iterative development.

The design and implementation of the system is divided in two areas, the inTelly system and tools used by the system. The tools are big parts of the system, which are used by the inTelly system to provide some functionality. The two tools developed are:

- Agent framework
- Categorisation

The Agent framework provides the functionality required to make the agents of the system. The reason for separating the description of the agents and the description of the Agent framework is that the agents is specific for this project whereas the Agent framework can be used without any changes as a Java library in other projects. The Categorisation tool is used by the Categorisation Priority Agent (see 24.4 Categorisation Priority Agent) to calculate the categorisation priority that is a part of the AI-value presented in the Program list. Three agents calculate the AI-values:

- Categorisation Priority Agent
- Group Priority Agent
- Keyword Priority Agent

The functional requirements and the performance goals are tested in the test sections for each module. The usability goals for the User interface are evaluated in a validation test after the description of the system.

20. inTelly System

This chapter will provide an overview of the design and implementation of the system. The system is divided into different modules, and the interfaces between these modules are then described. Another major aspect of the system is the personalisation of the TV-programs this is described further in this chapter. Each of the modules in the system is described after the description of the User interface in the following chapter.

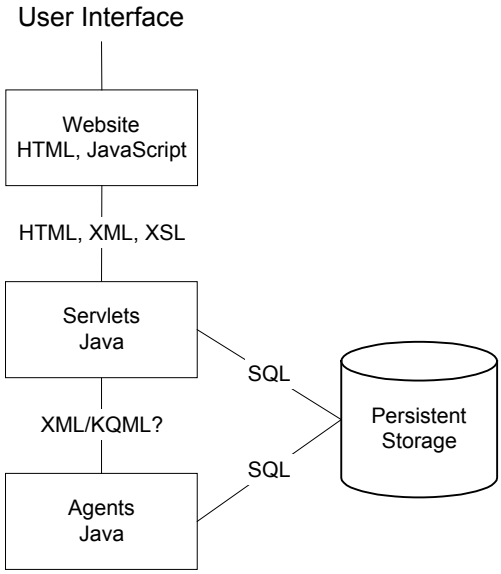

Figure 22 : The main modules of the inTelly system.

The technical aspect of the pages shown in the User interface is dealt with in the module Website. The Website is present on the users computer (client-side), and the remaining modules are present on the application provider's computers (server-side). The Servlets produce the pages for the Website, based on the data provided by the agents. The agents produce data based on data provided from an information channel on the Internet and data produced by the user. These data are stored in the Persistent Storage.

It is chosen to use browsers instead of applications because the users are getting quite accustomed in using browsers and because then there is no need to install any application on the users computer. Servlets are used to get directly into Java without using any scripting languages.

20.1 Interfaces

The interfaces in the system are defined by the interfaces between the (see Figure 22):

- Website and Servlets
- Servlets and Persistent Storage
- Servlets and Agents

- Agents and Persistent Storage

Each of these interfaces will be described in the following sections.

Website and Servlets

The interface between the Website and the Servlets consists of the user entering data and requesting pages, when the user has requested a page, the Servlets sends pages of the Website back to the browser.

The communication between the Website and the Servlets is based on HTTP. When the user requests a page in the browser, a HTTP request is sent to the web server, and the web server redirects the request to the Servlets. The request consists of an URL requesting a page and maybe some data as part of the URL.

The Servlets receives information from the client in different ways:

- URL query
- Form variables
- Cookies

URL queries has the form:
http://intelly.dk?data1=1&data2=hello

The URL is parsed in the Java Servlets, and the content of each of the variables can be fetched.

One of the most common ways of sending information from the client to the server is using forms. Forms consists of several input elements like text fields, check boxes, radio buttons, etc. Then the form is submitted to an address and the value of the input elements can then be fetched in the Java Servlets.

Another way of sending data from the client to the server is using cookies, these cookies are stored on the client computer, and are then available next time the user reaches the same page. Cookies are usually used to login a user automatically, but can be used to store other information.

In the Servlets there are different functions available from the Website:

Function	Description
Create user	Creates a user with a unique id, a username and a password.

Delete user	Delete the user and all information in the system regarding this user.
Save filters	Saves the filter settings of the filter menu in the database. This is done each time the user changes the filters.
Change program priority	Changes the user priority of a program in the database.
Change repeater priority	Changes the user priority of a Repeater in the database.
Remove message	Removes a message, so that it will not be shown to the user again. E.g. the wizard presented when creating a user should only be showed once.

These functions are used without getting a response from the Servlets and back to the browser, except Create user and Delete user, which is redirected to the inTelly.dk front page.

Data from the Servlets to the Website (browser) is send as XML, and is presented in the browser with XSL (see Appendix E for details about XML and XSL). The data sent to the browser is XML-strings that contain data for each page. This will be described further in chapter 22. Website.

The advantage of using XML and XSL instead of just sending a HTML page is among other things that when separating the data and layout in two files, the layout file only has to be sent once. Apart from that the data in both the XML and XSL file can be changed dynamically at the client side. This actually means that there is no need to fetch all the data again from the server, because they can be changed at the client side. If data should be stored for future use, they are both changed on the client side and also sent to the Servlets. This way the communication between the client and server is kept at a minimum. E.g. when the user wants to sort the program list by another column, there is no need to send all the program data again, the XSL file is dynamically updated on the client side, and the browser is repainted without sending any data.

Servlets and Persistent Storage

The data that is sent to the browser (Website) is fetched from the Persistent Storage and data received from the Website are also stored in the persistent storage.

There is made a static class which is used by all the Servlets which require database access, the database class provides the following primary functions:

Function	**Description**
Get program list	Sends the program list to the Website as XML for one user.
Check login	Checks if the username corresponds with the password
Get user profile	Sends the user profile for a user to the Website as XML.

Insert program priority	Inserts a user priority for one program in the database.

To make the transfer of data to the client as quick as possible, some of the database functions send the data directly to the browser, instead of back to the requesting Servlet. The reason for sending the data from the database class is the volume of e.g. the Program list, which can consist of 3000 programs with maybe 250 characters each. Tests have shown that it takes a long time for Java to manage this in one string object, but if they are split up and sent for each program the transmission is much faster.

When inserting a user priority for a program, the data is also sent to the agents. This is explained further in the next section.

Servlets and Agents

The main interface between the Agents and the Servlets is actually using the Persistent Storage just explained in the preceding section. But when inserting a priority for a program, this priority is also sent directly to the agents (the Servlet Agent). A client-server pair with a TCP/IP-connection establishes the communication between the Servlets and the Agents. The Servlet client sends commands and data to the Servlet Agent.

The following data can be sent from the Servlet client:

Function	Description
Save program priority	Sends the user priority of a program to the agents. This will eventually make the agents make new AI-values for the user.

The commands and data exchanged follow the XML-standard (see Appendix E).

Agents and Persistent Storage

To save the data in the system a database is used.

The following primary data-queries is used to fetch and save data:

Function	Description
Get priorities	Gets all the users priorities of the different programs.
Save programs	Saves the programs fetched by the Ripper Agent.
Insert categorisation priority	Inserts a categorisation priority for a user for one program.
Insert group priority	Inserts a group priority for a user for one program.
Insert keyword priority	Inserts a keyword priority for a user for one program.

Different agents use these functions. When data is fetched from the database, it is converted to XML and then sent to the different agents that need them.

20.2 Personalisation of TV-programs

One of the main issues in this project is personalisation of information. In this project the information that should be personalised is the TV-programs. The programs can be personalised by highlighting certain programs, removing other programs with filters and sorting the programs according to the individual users interests. To make this personalisation there is used user priorities and AI-values, the user makes the user priorities explicitly and the AI-values are made automatically by the inTelly system.

The filtering of the programs is done with the filters in the left side of the Program list and by inserting a threshold in the AI-values. This threshold decides if a program should be hidden or not. The programs that have been priorities negatively by the user are also hidden. The hidden programs can be made visible by checking the Show hidden (Danish: "Vis skjulte") select box.

The programs the user has prioritised positively are highlighted in the program list. The priorities explicitly chosen by the user overrules the AI-values automatically calculated by the system. Another way for the user to prioritise programs is by using automatic prioritising, this will automatically prioritise the selected Repeaters (see the section 21.5 Automatic Prioritising - Repeaters).

In the performance goals it is stated that the AI-values must match the user priority with an accuracy of 90 percent (see section 14.2 Performance Goals). The overall objective for the system intelligence is to perform this personalisation.

The features in the individual user profile to be used in filtering and sorting of the programs are:

- Channels
- Categories
- Keywords
- User priorities

These features should be used to make the AI-values and to remove programs. The channels and categories are used in the filters shown at the graphical user interface, the user controls these filters and can be used to remove programs. The channels and categories could be used to increase the AI-values for the chosen channels, but this is not implemented.

The keywords and the user priorities are used to make some personalisation agents that contribute to calculate the AI-values. These agents are:

- Categorisation Priority Agent

- Group Priority Agent
- Keyword Priority Agent

The categorisation priorities are calculated according to the users earlier priorities of the programs in the TV-guide. The agent that calculates the categorisation priorities is described in section 24.4 Categorisation Priority Agent. The categorisation method used by the Categorisation Priority Agent is presented in chapter 27. Categorisation.

Another use of the user priorities is to divide the users in groups according to their earlier priorities. When two users are considered in the same group, the new priorities made by one of the users will adjust the AI-values for the same programs for the other user. This contribution to the AI-values is explained further in the section 24.5 Group Priority Agent.

The keywords are used to increase the AI-values for the programs that contain the keywords in their title or description (see section 24.6 Keyword Priority Agent for details).

The number of programs removed from the program list according to the different personalisation methods just presented in this section will be presented in the different sections in the final design where the different methods are explained further. And the correspondence between the user priority and the AI-value will be evaluated in the conclusion.

21. User Interface

The user interface presented in this chapter is the final user interface, and it is designed with grounds in the results from the different tests made in this project, the functional requirements and testable goals, the analysis of the competing products and the ideas of the project group. The focus will be put on the user interfaces for logged in users, because these user interfaces are considered most important.

There are some main elements on the user interface that appear on several or all the different user interfaces these are listed below:

- Menu bar and Logo
- Messages
- Tool tip

The user interface itself is divided into several pages, which are listed below:

- Program List
- Automatic Prioritising - Repeaters
- User Profile
- Help
- Create User
- Login

First there will be described some general issues concerning the user interface and then the main elements that appear on more than one of the website's page will be presented in the following sections. After that the single pages in the user interface will be described.

The design of the user interface is optimised to a resolution of 1024*768, because 61% of the users that filled out the questionnaire use this resolution (see Appendix L). Before the product is released, it should be considered to adjust the user interface to other resolutions as well. Primary because a user interface designed for 1024*768 can be more or less useless for a person using a resolution of 800*600 (14%), on e.g. a laptop computer. Furthermore 90% of the advanced users were using Internet Explorer 5.0 or newer as browser (see also Test report - Test Result - Questionnaire), because of this, the design is optimised for this browser. But again it should be considered before release to optimise the user interface for other browsers, to support the users that are not using Internet Explorer.

21.1 Menu Bar and Logo

There is a menu bar and a logo present on each page. The logo indicates who has made the system (see Figure 23). This logo also makes it easy for the users to see that they are on the

right website, when they changes page within the inTelly website. Furthermore a consistent menu is presented on each page. There are two different menus available, one when the user is logged in (see Figure 23) and another when the user is logged out (see Figure 24).

Figure 23 : User interface of Menu bar and logo. It is placed in a frame at the top of the screen. This is the menu bar when the user is logged in.

The obvious difference between the two menus is that when the user is logged out, the user is able to go to the login page, but this is not be possible when the user is logged in, instead it is possible to log out. When the user is logged out the create user option is also available. And the Automatic prioritising and the User profile is only available for logged in users.

Figure 24 : User interface of Menu bar and logo when the user is logged out.

The menu is available on all pages in the website, which avoids that the user ever feels trapped, because it is always possible to go to e.g. the Help ("Hjælp") or to some of the other pages. The menu makes it easier to navigate, because the menu indicates which page is currently being displayed (the text). For more arguments see chapter 16. inTelly – Version 0.1.

21.2 Messages

The system is able to present messages to the user in a general and consistent manner, e.g. when the user for the first time tries to prioritise a TV-program. A message slides in at the top right corner of the screen, explaining to the user what has happed with the program he or she just prioritised. The message disappears after a few seconds.

The demands for the messages beyond those mentioned in chapter 16. inTelly – Version 0.1 are that the user shall be able to read previous messages. It is important to be able to see the previous message if the user miss reading the message for some reason. A better solution might be to implement a history function, which lets the user navigate through all previous messages. This has not been implemented, primary because the need for this feature is considered to be infrequent.

An additional functionality given by the messages and also a wanted feature by the users (see functional requirement 5) is that they are also used as a wizard when the user is creating a new user in the inTelly system. This wizard is a list of things to be done when creating a user. The wizard assists the user, in a way so that the user can foresee what is needed to create a user.

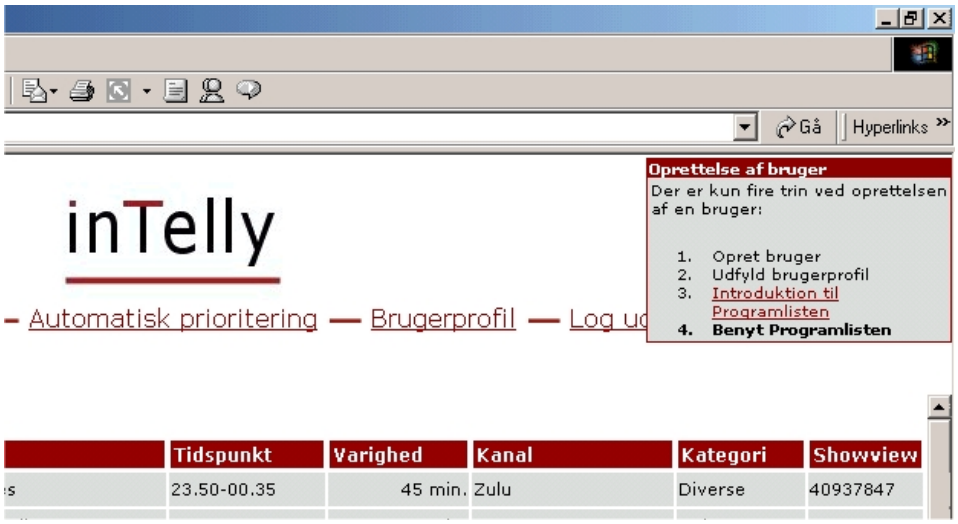

Figure 25 : The user interface where a message is present in the top right corner.

The last message is available at all times, by pressing a small 'b' in the corner of the screen. The 'b' is hidden behind the message showed in Figure 25 but it can be seen at Figure 27.

Another relevant use of messages could be to use them for presenting a message that tells the users that a program he or she wants to see begins in e.g. 10 minutes or that a Repeater that the user usually wants to see unexpectedly is missing (see Figure 13). This has although not been implemented in the final version.

It is chosen not to use an interface agent to present messages to the user, because of the dislike ness of these agents. This was uttered by some of the test participants in the Assessment test, when the idea was discussed.

21.3 Tool Tips

To assist the user in understanding the different parts on the user interface, pop-up descriptions are used. The pop-up descriptions where asked for by the test participants of the heuristic evaluation I (see Test report – Test result – Heuristic evaluation I). These tool tips provides a context sensitive help, which is considered to be a useful feature. This is primary based on the Assessment test where the participants did use the facility in several occasions.

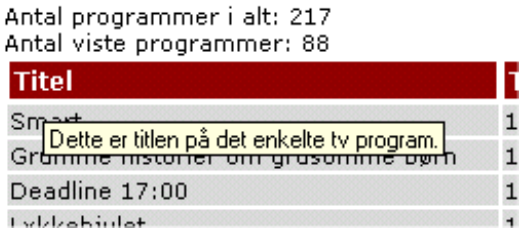

Figure 26 : An example of a pop-up description. This pop-up contains a short presentation of the title column ("Titel") in the program list.

21.4 Program List

This is the main page of the website. The page presents the TV-programs available for the user. When the user enters the inTelly website, a table containing information of the different TV-programs is presented. It is possible for the user to filter and sort this table (functional requirement 1). Furthermore this page assists the user in remembering which programs he or she has selected to see (functional requirement 3).

The Program list page consists of different components:

- Table of programs
- Prioritising of programs
- Notepad
- Filters
- Description of a program
- Miscellaneous

One or more of these components are present at the user interface at different times, depending on the user's actions. The user interface will also be different depending on whether the user is logged in or is logged out. The following will present an overview of how all these components are connected to each other. After this there will be a description of each of the components.

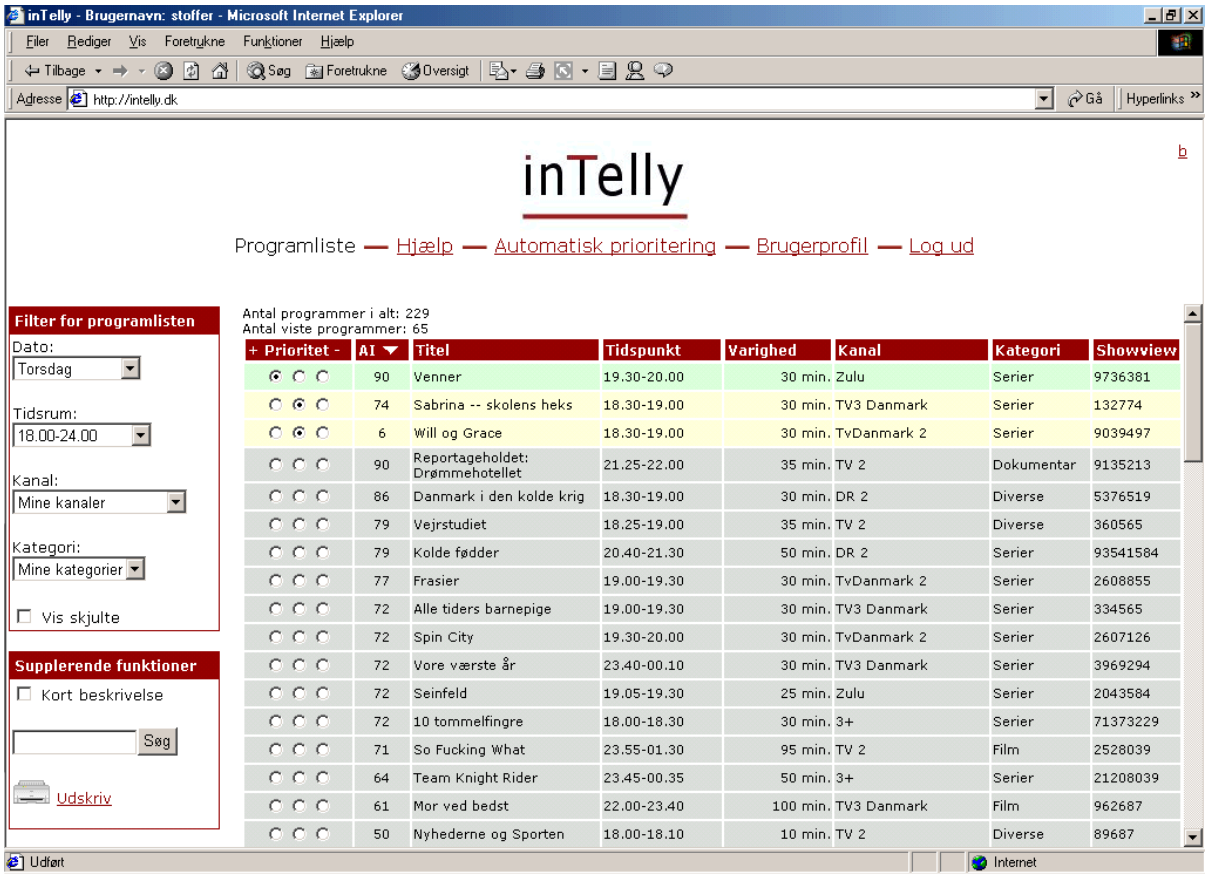

Figure 27 : User interface of the program list.

In Figure 27 the user interface of the program list can be seen. The only part missing is the detailed description of a program. The placement of the parts will be described in the following sections.

Table of Programs

The Table of programs (see Figure 28) contains the data about the different programs in a table, which is presented to the user.

Figure 28 : User interface of Table of programs.

The user is able to sort the table by each column ascending and descending by clicking the column head. This makes it easier for the user to overview the TV-programs.

The different columns contain the following data about each program:

- Priority.
- AI.
- Title of the program.
- Start time.
- Duration.
- Channel.
- Category.
- Show view code.

The priority is an evaluation decided by the user. The possibilities are (from left to right): "Want to see", "Maybe want to see" and "Do not want to see". The priority is part of the Prioritising of programs, but this is presented within the Table of programs, and that is why it is presented here as a column in the Table of programs. This priority is described in detail in the section describing the Prioritising of programs. The show view code is used to program VCR's.

AI is a priority decided by the inTelly system (see 20.2 Personalisation of TV-programs). The number ranges from 0 and 100, where 0 indicates that the system thinks the user does not want to see the program and 100 indicates the system finds that the user wants to see the program.


Antal programmer i alt: 240
Antal viste programmer: 139

+ Prioritet -	AI	Titel	Tidspunkt	Varighed	Kanal	Kategori	Showview
◉ ○ ○	50	Masken	21.00-23.25	145 min.	3+	Film	46701915
		Når den undseelige Stanley Ipkiss tager sin mystiske grønne maske på og forvandler sig til The Mask...	21.00-23.25				
◉ ○ ○	50	Fodbold: UEFA Champions League	20.00-23.00	180 min.	TV3 Danmark	Sport	85232489
		Igen er der lagt op til en spændende kamp, når Bayern München tørner sammen med Raul (bill.) og han...	20.00-23.00				
◉ ○ ○	50	Camille Claudel -- en kvinde	20.00-22.45	165 min.	Zulu	Film	69116199
		Den talentfulde 17-årige Camille kommer til Paris, hvor hun bliver den store billedhugger Auguste R...	20.00-22.45				
○ ◉ ○	50	Danmark i den kolde krig	16.30-17.00	30 min.	DR 2	Diverse	5389083
		Amerikanerne kommer...	16.30-17.00				

Figure 29 : This figure shows the short description and graphical format of the time of day and duration. This is showed when the user selects to see the short description of the programs.

The Table of programs (see Figure 29) is the main part in the Program list page; the filter, prioritising of programs and the miscellaneous component, controls the content of the list.

Furthermore the content is dependent on the intelligence of the system (if the user is logged in). The user can as already mentioned control, which programs is showed, by the prioritising of programs. The prioritising of the programs is an integrated part of the Table of programs and will be described in the following section. The choice of integrating it into the Table of programs originates from the first heuristic evaluation (see Test report – Test result – Heuristic evaluation I).

Below there is listed some suggestions to additional functionality, which has not been implemented:

- The user could define and adjust the columns in width.
- User defined sequence of the columns.
- It could be possible to add and remove columns.
- User defined format of the date, duration and time of day.

This functionality could be part of the user profile, but an argument against it, is that designers should be careful not to make too many possibilities for the user to configure the user interface, it will also make it harder for the user to fill out the user profile.

Prioritising of Programs

There are two different kinds of evaluation of TV-programs in the system, one performed by the user and one performed by the system. The evaluation performed by the system is needed to filter out programs that are considered of no interest to the user. The demand for this functionality originates from the functional requirements.

For the system to be able to make an evaluation about which programs the user would like to see, it has to get some feedback from the user concerning the different programs the user likes and dislikes. The feedback is the prioritising done by the user; this evaluation has three degrees, ranging from negative to positive. These three levels seem to be acceptable and earlier tests with more evaluation levels also showed that it was not acceptable to use the higher number of evaluation levels (see 16. inTelly – Version 0.1). It is decided to make this prioritising in combination with the user adding programs to the notepad (“Want to see” and “Maybe want to see”) and removing programs from the Program list (“Do not want to see”). See Figure 30 for an example.

Figure 30 : User interface of Prioritising component. There is three degrees of user evaluation. When the user prioritises a program, the row change colour and the program is moved. The two colours green and yellow indicates the notepad that is always present at the top of the program list.

As already mentioned it has been chosen to use three levels of user prioritising:

- Want to see
- Maybe want to see
- Do not want to see

The use of three levels has shown to perform well in the Assessment test, where all three degrees were used, although some participants used primary the negative priorities and others primary the positive priorities. Apart from the prioritising performed by the user, the system also makes an evaluation of each program based on the user priorities. This evaluation also has different degrees (AI-values), which in the inTelly system is presented by a value in percent, ranging from 0-100 (0 corresponds to the most negative system priority).

When the user has evaluated a program, it is visual indicated on the user interface by colouring the rows in the Program list. The colour is dependent on how the user prioritised the program. The program is also moved accordingly to the evaluation, that is, the program list is always sorted by the user priority and therefore moved to the top or bottom according to the evaluation. If the degree “Do not want to see” is chosen, the program is moved to the bottom of the program list, and it should be noticed that this part of the list is often hidden, unless the user explicitly has chosen otherwise by checking the checkbox “Show hidden” (“Vis skjulte”) in the filter.

This prioritising of the programs by the user is an implicit creation of the dynamic part of the user profile. This implicit creation does not affect the user profile pages in the website, but the data stored for each user. This user profile is a dynamic user profile that is altered while using the system on a long-term basis. It is an inferred user profile individually made for each user (see the chapter 4. User Profiles).

The prioritising of the programs by both the user and the system, can be described by the following table:

	Colour	User	System		AI
Visible	Green	X		Want to see	0-100
	Yellow	X		Maybe want to see	0-100
	Grey		X	Positive system evaluation	50-100
Hidden	Dark grey		X	Negative system evaluation	0-49
	Red	X		Do not want to see	0-100

The programs are hidden when the user priorities a program as "Do not want to see" and/or the system priority is less than 50. It should be noticed that the user priorities overrules the system priority. This means that a system priority of 100 is hidden and coloured red if the user priorities the programs as "Do not want to see", etc.

The colour grey represents the systems idea of what the user "Want to see" and "Maybe want to see" and dark grey represents the systems idea of what the user "Do not want to see". Programs with the colour grey are visible at all times whereas the dark grey programs will only be visible when the Show hidden checkbox is checked.

Notepad

An important task for the inTelly system is to assist the user in remembering which programs to see. The need for this functionality was primarily remarked in the Observation test (functional requirement 3). The test showed that the participants had difficulties in remembering, which programs they would like to see after they have run through the complete list of TV-programs.

It is chosen to make the notepad an integrated part of the program list based on the results obtained in the iterative development (see chapter 16. inTelly – Version 0.1).

When a program is evaluated it is given a specific colour and moved either to the top or the bottom of the Table of programs. This sorting shall help the user in remembering what to see but it might also be a source for confusion. When a user evaluates a program it will be moved instantly and the user may wonder where the program has gone and will not immediately see the connection between the evaluation and the notepad. This is though tried explained in the cartoon made to describe the program list (see section 21.9 Help) and by the help of

messages. The problem will probably also have vanished when the user has become familiar with system.

Filters

The filters makes it possible for the user to filter the data presented in the program list, this is done with four select boxes, based on experience from competing products (see Appendix J) and the exploratory test. The four select boxes filter the programs in different ways, by:

- Date
- Time of day
- Channel
- Category

When the user selects a value in the filter (see Figure 27), the Table of programs is updated accordingly. Checkboxes could be used instead of select boxes, but this would take up much more space on the screen, and the Table of programs already takes up a lot of space. An advantage of using check boxes, could be that the user was able to select more, e.g. categories at one time, to provide this useful functionality there is made both advanced filters which uses checkboxes in a pop-up window and My channels and My categories which uses checkboxes in the User profile. Besides the select boxes mentioned above the filter menu also contains a check box named show hidden (Danish: "Vis skjulte"). This checkbox makes it possible to view the programs that are hidden by the system.

The different filters can have these values:

Date	Time	Channel	Category
Today	Rest of the day	My channels	My categories
Wednesday	0.00-6.00	All channels	All categories
Thursday	6.00-12.00	All Danish channels	Movies
…	12.00-18.00	All English channels	…
Monday	18.00-24.00	All Nordic channels	Documentary
	All day	All German channels	Advanced
	Right now	Advanced	
	Advanced	DR	
		DR 2	
		…	

The different values of the filters are made from the competing products analysis (see Appendix J), and the Advanced possibility is requested in the Observation test (see Test report – Test result – Observation test).

The two values My channels and My categories are defined by the user in the User profile (see section 21.8 User Profile), and only available when a user is logged in. A typical example of a users defined filter removes approximately 90 percent of the programs.

Apart from the filters presented visible in the user interface, there are some advanced filters available. These filters are used if Advanced is chosen in one of the select boxes. This advanced filter is presented in a pop-up window.

Advanced date:	Could be made with check boxes with the same items as the select box. Including a check box with Last week. This function is not implemented.
Advanced time:	Is made with four text fields where the user can enter the Start and Stop time for the TV-programs he or she wants to see.
Advanced channel:	The same items as in the user profile (see also the description of the User profile page in section 21.8 User Profile).
Advanced category:	The same items as present in the user profile.

A problem with the advanced filters could be that they are too difficult to understand and use, and there would be presented too much information to the user. But it should not be a problem because the advanced filters are hidden from the user interface, unless the user explicitly asks for them, by choosing advanced. The advanced filters are considered only to be used by the expert users.

Description of a Program

It is possible for the user to get a detailed description of the different TV-programs. This description should help the user in selecting which programs to watch. The description is presented in different ways:

- As an integrated part of the Table of programs (short description).
- As pop-up description in the Table of programs (short description).
- As a separate page where only the detailed description is presented.

Different amount of detail is presented in the three different forms. The shortest is the integrated part and the pop-up description. These show approximately 100 characters of text describing the program (see Figure 29). The integrated part also contains a graphical indication of when a programs starts and ends. The integrated part shows the information for all the programs in the Table of programs. It is possible to turn this short description on and off from the miscellaneous component. The pop-up description pops up when holding the mouse pointer at the title of the program. This feature is assisted by a more detailed

description of each program in a separate page when clicking the title of the program (see Figure 31).

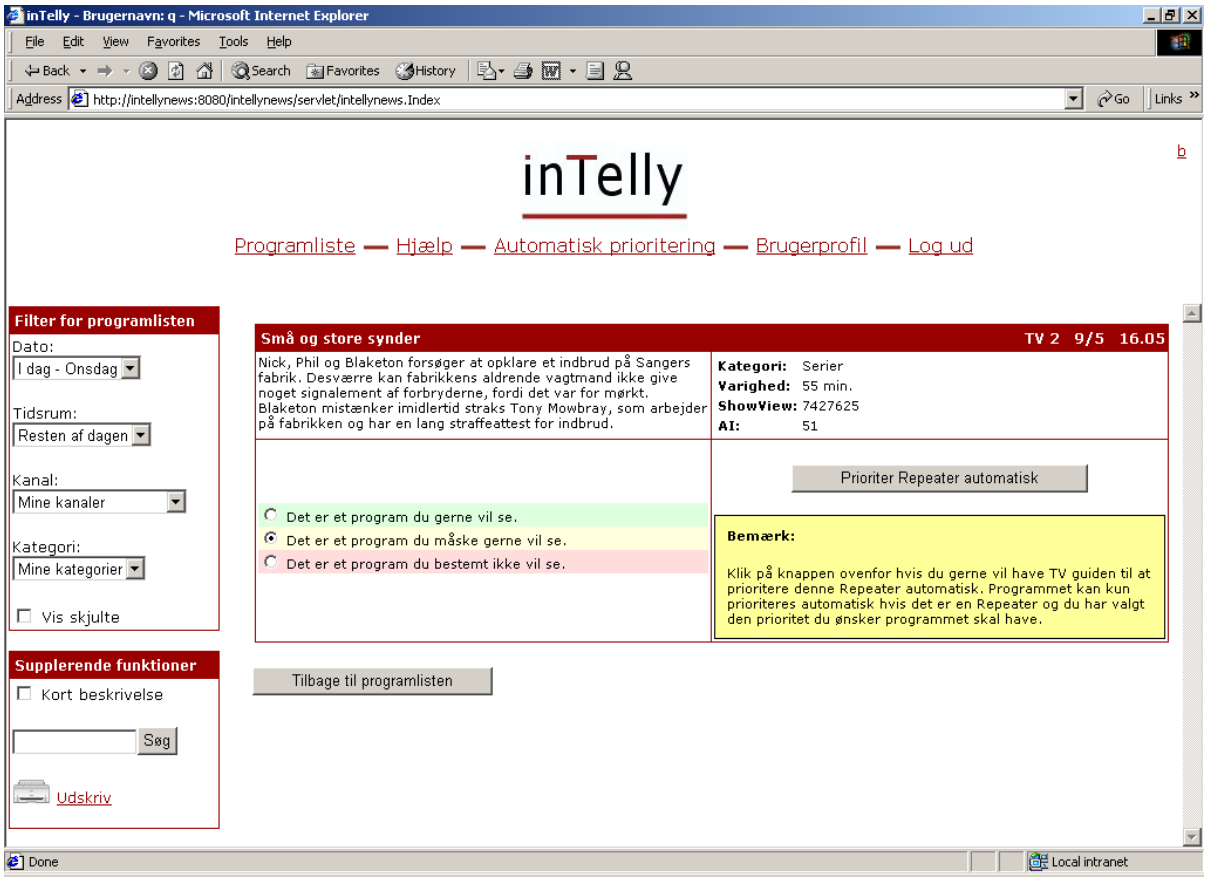

Figure 31 : User interface of Detailed description for one program.

The data shown at the detailed description is:

- Description of program.
- Data, column values from the Table of programs.
- Indication if this is a Repeater.

It is decided, based on the Heuristic evaluation (see Test report – Test result – Heuristic evaluation), that a short description should be available both as a pop-up description and with the possibility of showing all the short descriptions in the Table of programs. Furthermore there is a separate page available with all information concerning this one TV-program.

Miscellaneous

Miscellaneous contains the different elements, which does not belong to any other component.

The elements of miscellaneous are short description, search and print and can be seen in Figure 31 (Danish: “Supplerende funktioner”). The short description is implemented as a

check box, which makes it possible to show a 100 character long description of all programs when checked. This description also contains a graphical indication of when a programs starts and ends. The second element is the search facility. The search can be performed within all available TV-program data from today and seven days ahead (see Supplement P for a screen dump of the Search user interface).

The last element is the print functionality, which makes a printable version of the program list. In the implemented version the system only prints out the HTML of the program list but with the use of XML it could easily be made a PDF-document instead, which is supposed to be printed. The use of XML also makes it easy to transfer the information to other portables devices.

21.5 Automatic Prioritising - Repeaters

The automatic prioritising interface is where the user gets the repeating TV-programs presented. This interface is very similar to the program list and this is good because the user is then familiar with the interface and its functionality. The consistency of the program list and the repeater interface can although confuse people.

The user has the same possibilities of filtering and sorting as on the program list and of course the ability of prioritising a program is also here. There is no short or detailed description available because this description changes over time. There is a column that indicates the weekdays on which the repeater is broadcasted.

When a repeater is prioritised almost the same happens as when a program is prioritised on the program list. The only difference is that the automatic prioritising will prioritise this repeating program automatically from now on. An example could be; If a user likes to watch The Simpsons every night at 19.00 hours on 3+ and he/she prioritises this as “Want to see” in the automatic prioritising. The system will then mark this repeater as “Want to see” every day when it finds The Simpsons at 19.00 hours at the channel 3+.

When prioritising repeaters it can be chosen to update or not to update the interface each time a repeater has been prioritised. The advantage of not updating the Repeaters list is that it is possible to prioritise the Repeaters faster.

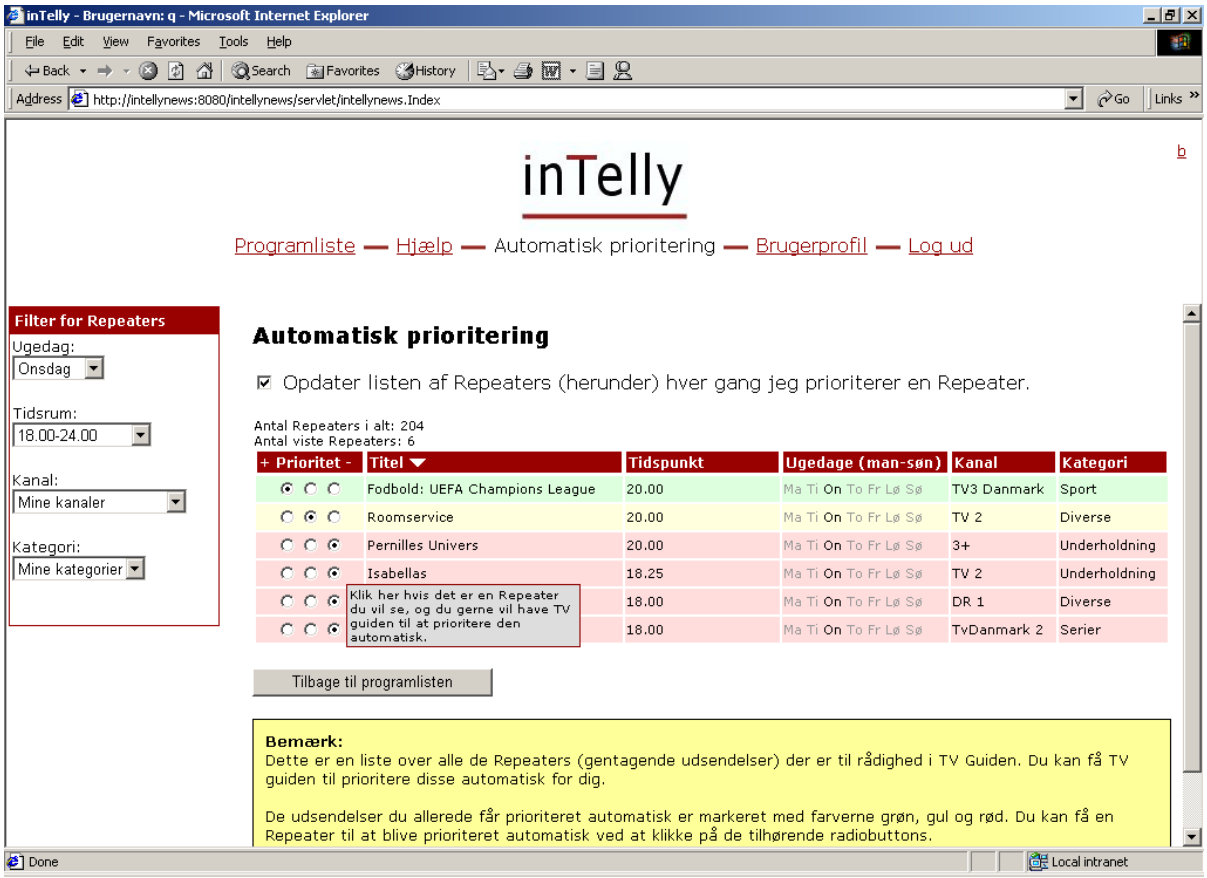

Figure 32 : User interface of Repeaters. Tool tips describes each type of priority that can be selected for a Repeater. The list of Repeaters is consistent with the Program list, in the way Repeaters are prioritised and presented. One of the differences is the column showing which days a Repeater is showed.

An additional functionality that could be provided is a search function in the Repeaters, but this functionality is not implemented.

The Repeaters can be used to both highlight and remove programs, 70-90 percentages is categorised as Repeaters. This makes it possible to remove a lot of the programs in the Program list.

21.6 Create User

This user interface gets the necessary data from the user in order to create him or her as an inTelly user. There are text fields where the user can enter a username and a password. If the user makes mistakes (e.g. enters an already used username or does not enter the password correctly) the system makes an appropriate response. There is presented no text on this page because the Assessment test showed that the user was not interested in reading it.

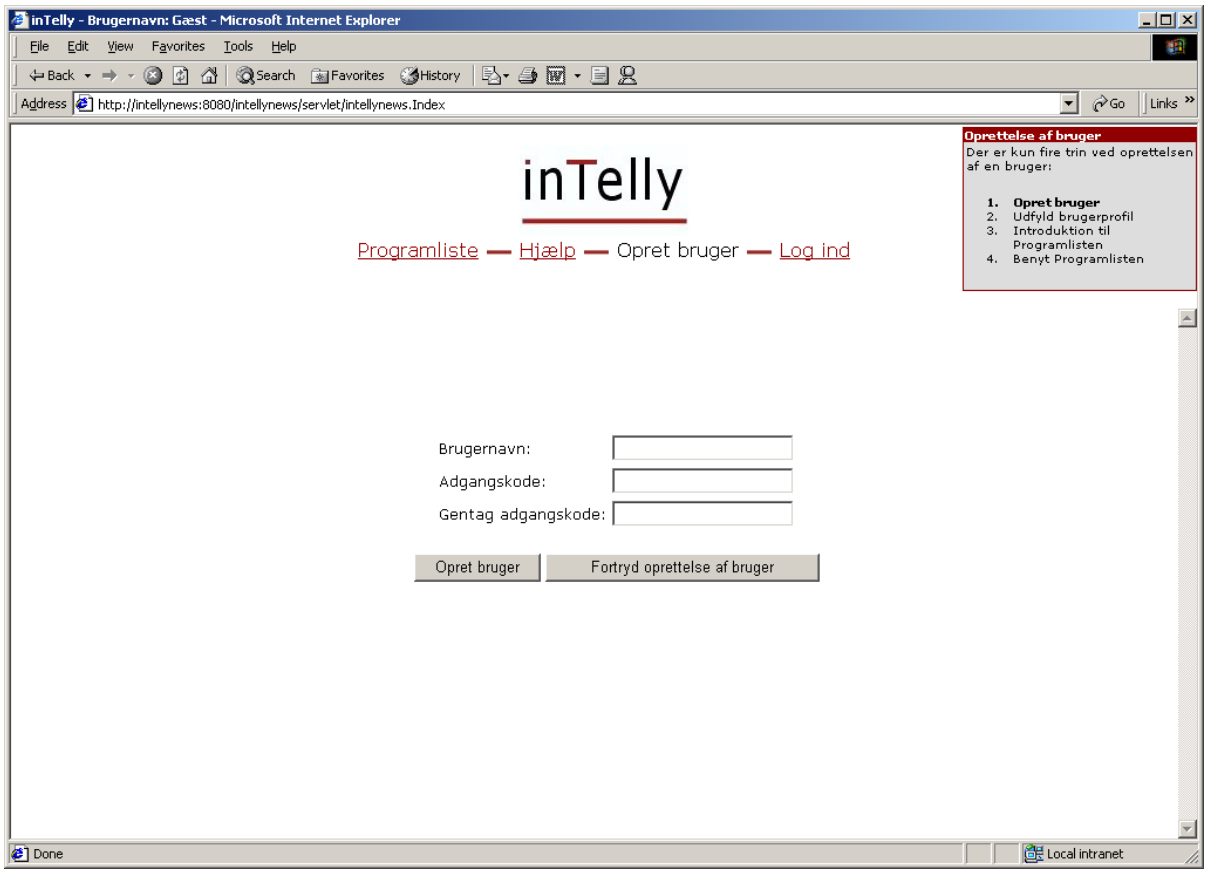

Figure 33 : User interface of create user.

To assist the user in creating a new user, a little wizard is presented as messages in the upper right corner; this wizard will guide the user in:

- Creating a user
- Filling out the user profile
- Introduce the user to the Program list
- Tell the user to use the TV-guide, because the creation is done

It has been proven necessary (see Test report – Test result – Heuristic evaluation) to show the user how to operate and understand the Program list and the concept of the TV-guide, as part of the creation of the user. To make a quick presentation, without too much text, which the users tend to ignore, a cartoon is created (see Figure 39). This cartoon shows how a user should prioritise some programs and how this affects the Program list. When the user is created and arrived at the user profile front-page the last two options in the wizard will be active links.

21.7 Login

The Login user interface makes the user able to login to the system (see Supplement P – Figure 83 to see a screen dump). When the user wants to login, he/she writes his/hers username and password, then presses the login button. When the user next time go to the

inTelly website, the user is automatically logged in via cookies, unless the user at the last visit explicitly logged out, then the user has to login manually again. The functionality, automatic login, is requested by several users in the Observation test (see Test report - Test result – Observation test for details), it was actually a reason for not using an entire TV-guide, even though the TV-guide is better than others.

As on the create user page all text has also been removed from the login page due to the user dislike ness of reading any instructions.

There could also be a checkbox where the user could choose that the system should use automatic login for the user, but this functionality is not implemented.

21.8 User Profile

In order to make a personalised TV-guide, the system has to get some information from and about the user. As mentioned in the Prioritising of programs section, feedback from the user is gathered about which programs the user likes and dislikes, this is dynamically gathered data for the user profile (see chapter 4. User Profiles), in addition to this information, static information is gathered explicitly from the user, e.g. which channels are available, which categories of programs does the user like, etc. The user explicitly gives these static data. This information is then used to optimise the intelligence that tries to figure out which programs to present to the user.

To gather these data, there are several sub pages in the user profile. The reason for having several pages is to provide an overview of the entire user profile with a menu. To guide the user through the user profile, navigation buttons is placed on each page, pointing to the next page (see Figure 34) and to the preceding page if possible. This helps the user in filling out the entire user profile. And he/she is still able to see how far he/she is, by looking at the menu, which indicates the current page at which he or she is. This navigation has been developed through the tests of the version 0.1, 0.2 and 0.3 in the iterative development.

Several of the pages in the user profile contain a small yellow box with the word important or note as a title. This box contains either important information like e.g. the user profile is saved automatically or a note that e.g. tells how inTelly.dk deals with the user data.

The user profile is as already mentioned saved automatically. This is done because several usability problems where found in chapter 18. inTelly – Version 0.3 concerning saving the user profile. Letting the user profile be saved automatically solves this problem.

Figure 34 : User interface of User profile. This is the front page of the user profile pages that contains some general data, e.g. the time and date for the last time the user profile was saved.

The first page contains some general data for the user. The username is presented; this makes the user able to see that it is the right user profile he or she is changing. When the user is just created, he/she is sent to the user profile. The user can here explicitly see that a user account has been created. In the title bar of the browser it is also possible to see the username of the logged in user. Furthermore the time and date for the last time the user profile was saved is showed. This way the user can assure that his/hers changes has been saved.

The second page contains information about which channels the user wants to watch programs at. The channels, as shown on Figure 35, are sorted by language, then each section is provided with a heading and a checkbox for selecting or deselecting all checkboxes in this section. Each channel is provided with a checkbox, for selecting this specific channel.

Figure 35 : User interface of User profile containing a possibility for selected/deselecting the different TV-channels.

The categories page is made the same way as the channels page (see Supplement P for a screen dump).

The next page contains keywords specified by the user (see Figure 36). These keywords are used by the intelligence in the system to find programs that might be interesting for the user. This option of having keywords originates from the observation test and the competing products (see Test report – Test result – Observation test and Appendix J).

Additionally the keywords could be associated with the categories of programs in the system. This means that the user could enter keywords to each program category. This is not implemented.

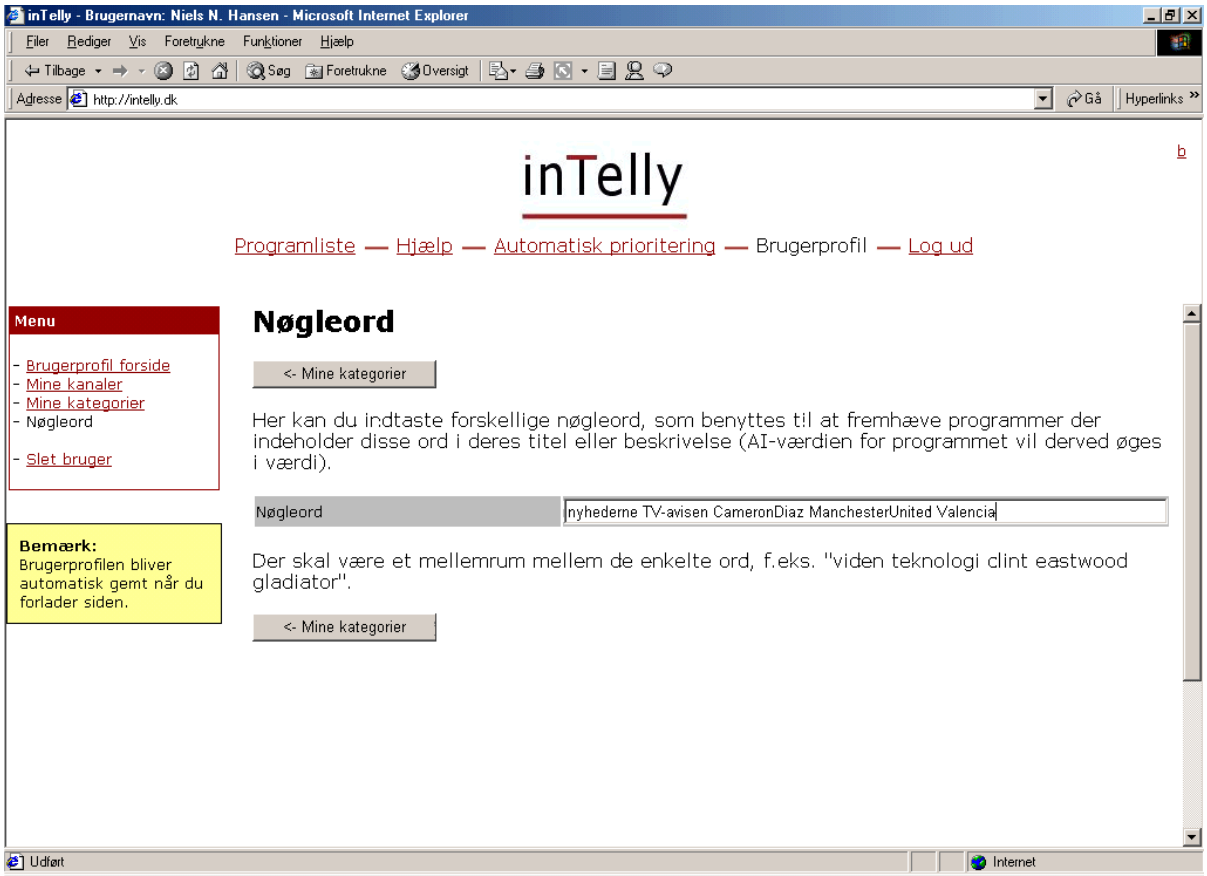

Figure 36 : User interface of User profile. This user interface gives the user a possibility to enter keywords that should be taken into account when the system finds programs to the user.

The last page in the user profile is delete user (see Figure 37). This page makes it possible for the user to delete him/her self totally from the inTelly system. The call for this function comes from both the project group and the observation test (functional requirement 4).

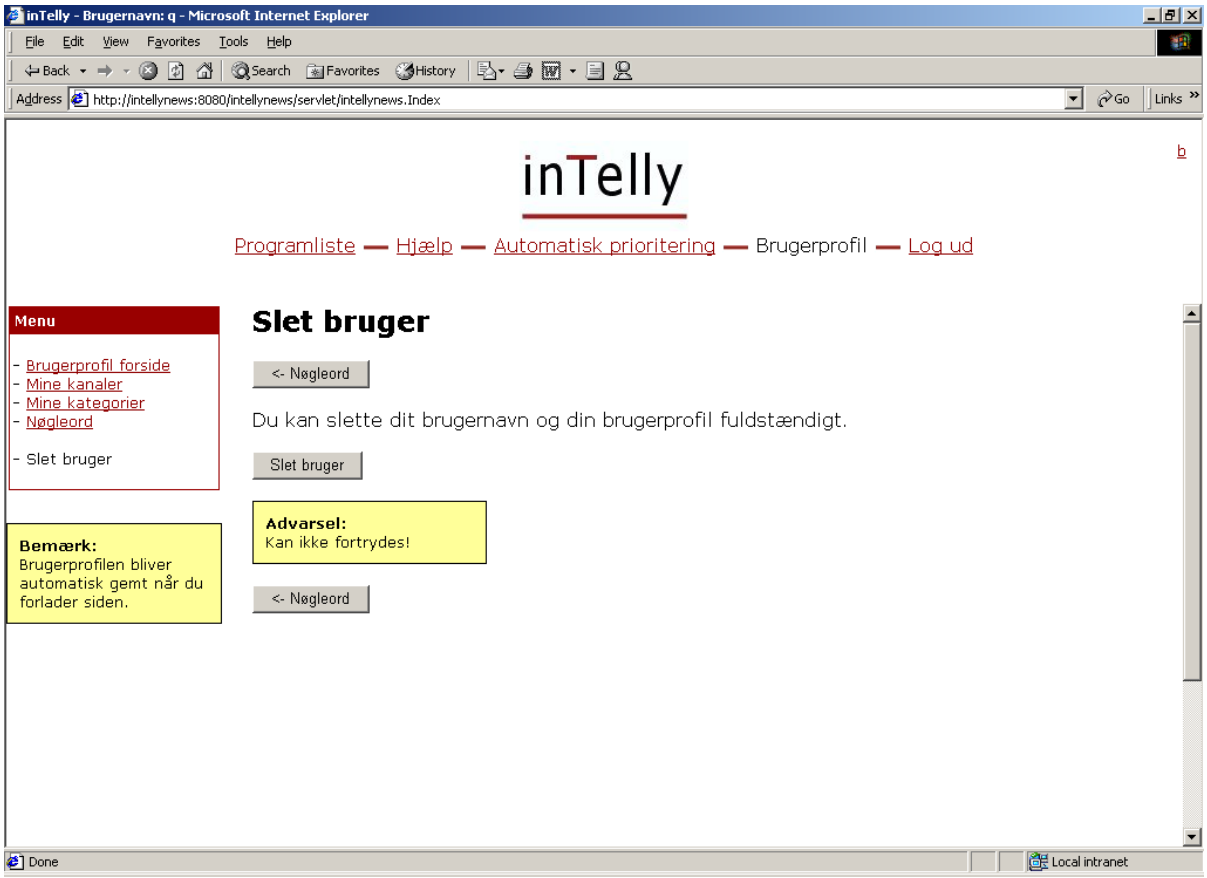

Figure 37 : User interface of delete user in the User profile.

This page is separated from the other pages in the menu in order to keep the user from accidentally pressing this link. The navigation buttons in the user profile, which lead the user through the creation of the user profile, does not lead the user to the delete user interface.

21.9 Help

The user interface for the help in the system is made much similar to the User profile interface, that is, with a menu to navigate between the different help pages (see Figure 38). This menu contains a link with help for each of the pages that are accessible from the top menu bar and also to the main parts of the program list. There is also the possibility of using a fast help (Danish: “Hurtig hjælp”) picture to select the necessary help.

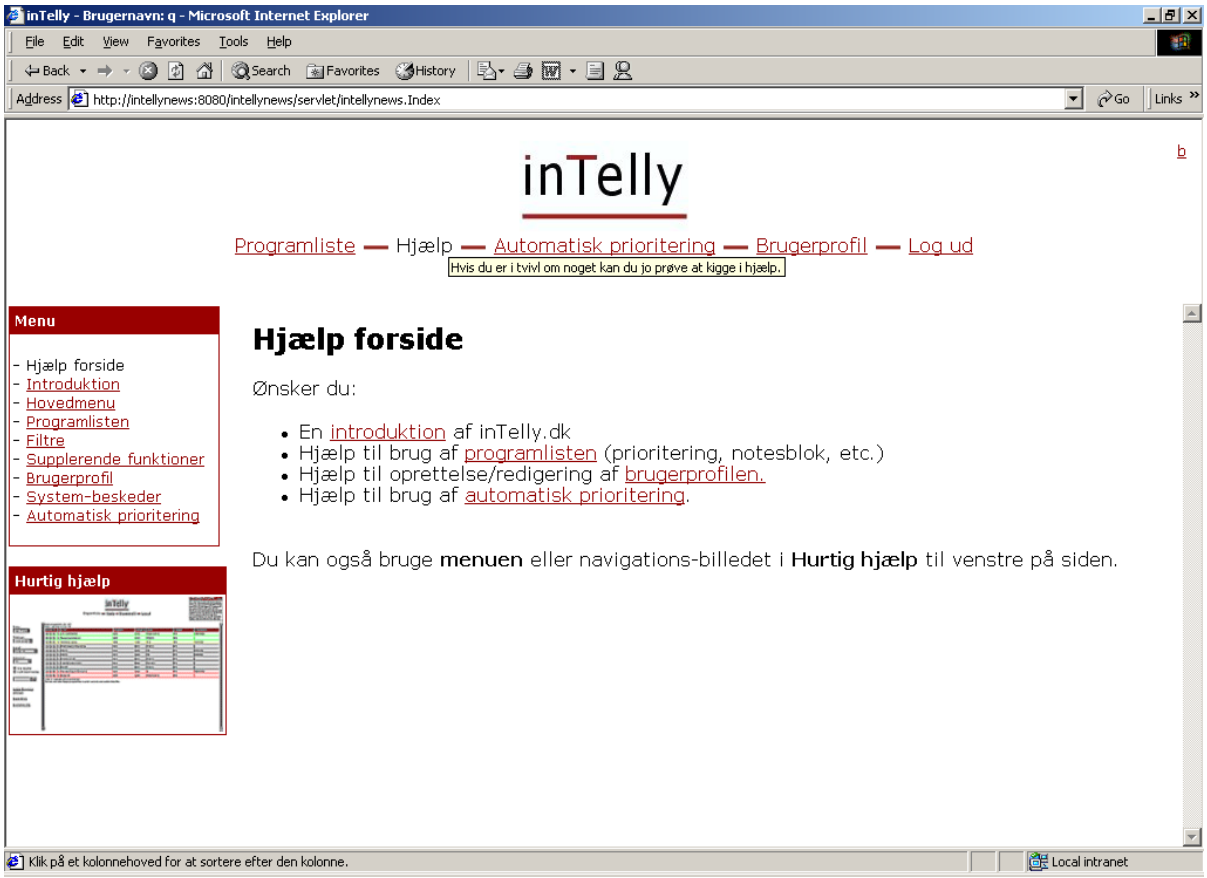

Figure 38 : User interface of Help front page.

Each page in the help system contains descriptions and graphical explanations of the inTelly system (see Figure 39 for an example).

There are several pages in the help user interface but there will only be presented the front page and the help for the program list because they are representative for all the pages.

The help text describing the system is tried held in short and precise descriptions and this is done because the Assessment test revealed that the users only want read very few lines of text or they will not use it at all.

It is tried to make a cartoon where there are used a lot of screen dumps together with short and precise descriptions. This is done to show the user how to operate and understand the Program list and it gives a quick presentation without too much text.

Figure 39 : User interface of Help.

This cartoon shall give a fast overview of the possibilities in the system. The use of a high number of screen dumps minimizes the text. The cartoon was created because of some problems with understanding the program list (see 18. inTelly – Version 0.3).

21.10 Search

When the user performs a search, a page much like the Program list is shown. The user is able to prioritise the programs within the search result; this means he/she can add them to his/hers notepad.

When the user wants to search for some specific e.g. X-files or John Wayne, he/she writes the search string into the text field and presses the search (Danish: “søg”) button. If any results are found they are presented like the programs in the programs list (see Figure 40 for example).

Figure 40 : User interface of Search.

An additional functionality that could be added is having filters on the search results as well, but this functionality has not been implemented and this might also make it difficult for the user to distinguish between the search page and the program list.

21.11 Conclusion

The final user interface design is made based on the different usability tests made throughout the project. The user interface, which is the most critical to design properly, was revealed by the first heuristic evaluation to be the prioritising of programs and how to implement a notepad with the selected programs.

The proposed suggestion to the user interface has minimised the number of usability problems, but it still contains some, which if tried corrected will result in new usability problems.

The final user interface has changed a lot since the first mock-ups that where presented in chapter 16. inTelly – Version 0.1. It is not only the look of the interfaces that have changed, but also the implemented functionality. There has also been added additionally functionality.

This final result of the user interface also appears to be much more user friendly than the first mock-ups. This therefore shows that the user testing during the design phase has been a very rewarding.

In the test of version 0.3 it has been proven necessary to show the user how to operate and understand the Program list. To make a quick presentation, without too much text a cartoon is created. This cartoon shall give a fast overview of the possibilities in the system. The use of a high number of screen dumps shall minimize the text and will make the cartoon very easy to use.

22. Website

The technical aspect of the user interface is described in this chapter. The Website is the pages shown to the user in the browser on the client side. The Website also contains some functionality, e.g. filtering the programs shown to the user and communicating with the Servlets (web server). The user requests pages to the browser, these are made as HTML pages sometimes including client-side JavaScript to provide some functionality. The HTML pages are made with XML-files transformed with XSL-files. The data is then separated from the layout of the different pages. The JavaScript is used because ordinary HTML is static pages, and if the user e.g. wants to sort the data in a different way, a JavaScript function is called which changes the XSL-file on the client-side (the XSL-file controls how the data is sorted), it would be waist of bandwidth to send the same data again.

This chapter will be divided into the following sections:

- Frames
- XML and XSL
- Control JavaScript
- Messages

The first section will describe how the Website is divided into frames, and the second section will explain how and why each frame uses both an XML-file and an XSL-file. The last two chapters will describe some of the JavaScript functions used to make the Website dynamic on the client-side.

22.1 Frames

The pages in the inTelly User interface are divided in three frames.

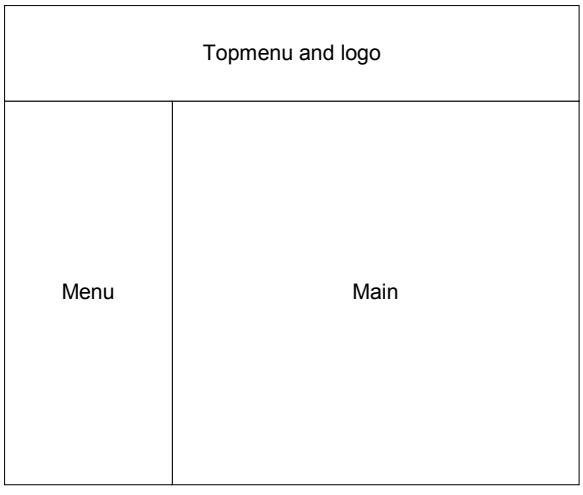

Figure 41 : The pages in the browser are divided in three frames.

There is made a frame in the top of the browser to make a consistent menu bar (Top menu) and logo (see 21.1 Menu Bar and Logo). The main part of the browser is used for a narrow Menu frame in the left side and a Main frame in the right side. The content of the Menu frame and the Main frame changes according to the page the user is visiting.

In addition to the three visible frames, there are two frames that are invisible:

- Control frame
- Command frame

The Control frame is used to control the entire Website, it is actually in this frame all the data is loaded. All the XML and XSL documents are loaded in this frame and stored in JavaScript variables. This way it is possible to store e.g. a XSL file, and if the user in the same session returns to a page the XSL is already loaded. The Command frame is used to send commands to the Servlets without updating the page that issued the command, this way it is possible to send a command without reloading the entire page and without flickering in browser.

By having all JavaScript functions in one frame, it is not possible for the user to open each frame in a new window this will not work. But that is not considered normal use of the TV-guide and it will be considered as acceptable to leave out this functionality. Another disadvantage of using one file for all the Java Scripts is that the browser has to load all the JavaScript for all the web pages.

22.2 XML and XSL

In each of the frames presented there is used XML and XSL to generate the HTML/JavaScript.

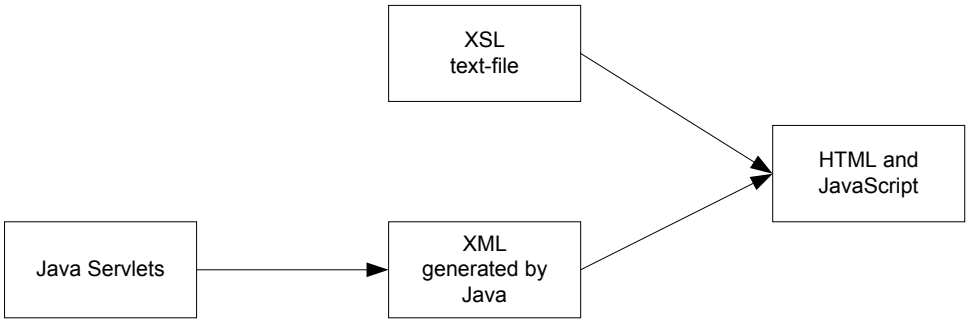

Figure 42 : The pages in the Website consist of HTML and JavaScript. This is generated from a XML-document and a XSL-document, the XML-document is generated in the Servlets.

To each page in the Website there is generated an XML-file in the Java Servlets, this file is never saved to disk, and it is created dynamically when needed. To each XML-document an XSL-document is needed to transform the XML-document to nicely formatted HTML, these

XSL-files are static text files fetched from disk. This form of response (XML and XSL) to the client's browser separates the layout of the data and the data it selves, which is very useful when maintaining the web pages and when working in teams on the development of a website.

Example

The Program list is used as an example of how the XML-document are designed and how the XSL-documents are used to transform the XML-documents.

The data of the different programs are available in the browser as an XML-document in the following format (see Appendix E for a description of XML):

```
<Program list>

   <sortInformation>
      <column>starttime</column>
      <direction>ascending</direction>
   </sortInformation>

   <program>
      <programId>23342</programId>
      <title>James Bond</title>
      <startDate>20010518</startDate>
      <startTime>1230</startTime>
      <dateCaption>Lørdag 20.30-22.45</dateCaption>
      <duration>135</duration>
      <userPriority>90</userPriority>
   </program>

   <program>
      …
   </program>
   …
   <program>
      …
   </program>
</Program list>
```

The XML-tag sortInformation contains information about which column the table of programs should be sorted according to and if it should be sorted descending or ascending. In this example the table should be sorted according to the starttime column. The starttime is how many minutes past midnight the current TV-program starts. And the dateCaption tag is the format of the starttime that is showed in the browser. The other tags should be self-explaining. For each TV-program there is a program tag that contains the data about the program.

The data in the XML-file is presented as an HTML table, with a table head, which makes it possible for the user to sort the table (see 0.
Program List), the table is sorted by the XSL command:

```
<xsl:sort select="starttime" order="ascending"/>
```

This makes the XSL-file sort the Program list ascending according to the starttime. If the user clicks on one of the columns in the table head; the sort function in the Control JavaScript is called (see next section).

Two special constructed pages in the Website is the Help page and the User profile page, they are not constructed quite like the other pages with an XML-document and an XSL-document. The Help pages are ordinary HTML-documents, which are fetched from disk on the web server, this is done because the Help pages are completely static. The User profile pages are constructed in a different way, these pages are dynamic because the content change according to the user, so these are made as XML-documents like the rest of the Website. The special thing about the User profile pages is that it is actually only one page, even though it is visually presented as several pages; this is constructed with the help of JavaScript. It is done because then the User profile only has to be saved and fetched once, when the user leaves and enters the User profile pages, and there only has to be constructed one Java Servlet to make the User profile.

22.3 Control JavaScript

All the functions called in the Website, e.g. the sort function, are placed in one JavaScript-file named Control. The Control JavaScript is used to control the entire Website. All the XML-documents and XSL-documents are stored in this JavaScript; this makes it easier to initialise the Website when the needed documents have arrived. The Website is initialised once when loaded, where all the appropriate variables are set and the requested page is shown.

The main functions of the Website is:

Initialise	Is used to initialise the appropriate variables once. And show the requested page. Is called when the needed data is available.
ChangePage	Change the page either in the Menu frame or Main frame, or both. This also changes the marked page in the Top menu.
RemoveMessage	Some of the messages shown to the user is only showed once, e.g. the introduction to the Priorities in the Program list. This function is then used to remove the message from the current users message list.

For each page in the Website there is some common functions:

GetPageXsl	Fetches the XSL-document for the requested page. This function is only called once for each page, then the XSL-document is saved in a JavaScript variable.
GetPageXml	Fetches the XML-document for the requested page. See explanation below this table.
UpdatePage	Refreshes the page in the browser by transforming the XML-document with the XSL-document.
InitializePage	Is used to initialise the appropriate variables when the XML for the requested page is fetched with GetPageXml.

The GetPageXml function is called when the data have to be updated from the server. Some of the XML-documents do not have to be fetched more than once, e.g. the menu in help does not change if it is visited two times. An example of when it is necessary to update the XML-document could be when the user changes a priority for a Repeater at the automatic prioritising page. Changing this priority will change all the priorities for all the TV-programs with the same title, channel and time of day (see 21.5 Automatic Prioritising - Repeaters for details). When returning to the Program list from Automatic prioritising page the priorities for some of the programs the current day could have changed priorities, as a result of changing the priority of the Repeater. Then the XML-document for the Program list has to be updated.

Additionally there are extra functions for some pages

Sort	Is used to sort the tables of programs showed on different pages in the Website.
ChangeProgramPriority	Changes the priority for one program for the current user. The priority is both changed in the XML-document on the client-side and the priority is also sent to the Servlets.
ChangeRepeaterPriority	Same as above just with a Repeater instead of a Program.
SaveFilters	Sends the filter values in the filter menu to the Servlets. This is called when the user changes the filter; to avoid the user having to set the menus each time the user reaches the Program list.
SaveSortInformation	Same as above just with information about how to sort the Program list instead of filters.
ChangeFilter	Is called from the Program list menu and changes the content of the Program list.
ShowDescription	Is called from the filter menu and makes the Program list either show the short description of the programs or hide the descriptions.

22.4 Messages

There are different messages presented to the user while he or she uses the system. These messages are constructed by the use of a JavaScript library [OL].

They are constructed using layers in HTML, making it possible to make the message float over the existing HTML in the browser. This is used to make the message carefully slide at the upper right corner of the browser.

22.5 Installation

The current browsers Internet Explorer 5.5 and Netscape Communicator 6.0 and earlier do not per default support Microsoft MSXML 3.0 that is used in this Website. There are made some installation files to make the browsers support XML and XSL, the procedure for installing these files is quite short, but very annoying:

- Download and install MSXML 3.0
- Download the file xmlinst.exe
- Run a dedicated bat-file

The files and some more details can be found at:
http://inTelly.dk/install

The browsers will in future version support MSXML 3.0, which follows the XML and XSL version proposed by W3C.

This installation has not been part of the design of the User interface and Integrated design in this project because it is thought, that it will not cause any problems, when the browsers do support it in future versions. The system should be adapted to different and also earlier browsers; this can be done server-side with XML and XSL. But this is not considered further in this project.

22.6 Test

Several of the functional requirements for the system are fulfilled in the Website. This is tested when the Servlets are up and running and TV-program data are available. Then the different functionalities required in the Website are tested, by using the inTelly system in a browser.

There is provided filters for the user in the filter menu for e.g. the Program list this is functional requirement 1.

It is possible for the user to turn the short description on and off for each program in the Program list, this is partly functional requirement 2.

XSL makes it possible to sort the Program list according to user priorities, when the Program list is always sorted after this column, a kind of notepad appears in the top of the Program list. The notepad functionality is requested in functional requirement 3. In close connection with the notepad, functional requirement 11 says that it should be possible for the user to prioritise the different TV-programs in the TV-guide; this is done when selecting programs for the notepad.

A combination of the Website and the Java Servlets (explained in the next section) gives the possibility of creating, updating, deleting and view the users profile, this is requested in functional requirement 4.

Functional requirement 5 requests a wizard to guide the user through the creation of a user profile, this is done in JavaScript that shows different messages, at different steps in the creation of a user.

The files needed to be loaded in the browser to fetch the front page of the inTelly Website:

Size (kb)	Name
43	Control.js
37	ProgramList.xsl
37	Overlib.js
7	Message.js
20	ProgramList_Filtermenu.xsl
4	Topmenu.xsl
Typ. 150	ProgramList.xml
7	Programlist_Filtermenu.xml
1	Topmenu.xml
4	Logo.jpg
1	Printer.gif

The amount of data needed for the front page is 161 kbyte plus some data for the program list. With a 56 kbit modem this should take approx. 20-40 seconds, which is way too slow. But it should be noticed that when the user has fetched the data once it is not necessary to fetch the data again except a few program data. This means that moving around the Website is much faster when the data is first loaded. With a 256 kbit connection the load time is approx. 4-8 seconds, which is acceptable.

Shortening function and variable names in JavaScript could reduce the amount of data that should be loaded by the browser, and maybe some of the functions could be gathered in fewer functions. The OverLib library can be shortened because not all of the functionality is needed.

22.7 Conclusion

The Website fulfils the functional requirements 1,2,3,4,5,10,11,13 and 14 that are proposed in the Initial design phase.

It has been very successful to use XML and XSL to split data from layout. This is a very important issue when constructing websites in teams, because the layout and the handling of data are often performed by two different kinds of employees (graphic designers and software designers).

JavaScript has been used to change the XML- and XSL-documents dynamically at the client-side. This has been quite successful because this avoids the need for communicating with the server each time the user changes something at the Website.

It is possible to make the system available on other devices, such as WAP-phones, Palms, etc. without changing much code, because of the use of XML/XSL.

23. Servlets

The Servlets of the system provide the pages to the Website. The Servlets generates XML-documents with data from the database and sends them to the browser. The Servlets also provides a link between the Agents and the user's browser. This layer could be developed in PHP, ASP or some other server-side scripting language. It was decided to use Java Servlets, because the rest of the system is written in Java. This give some advantages, e.g. reuse of code and the power of Java are also stronger than the scripting languages (see Appendix F for a short introduction to Java Servlets).

Java Servlets are very well suited for producing XML-documents compared to other scripting languages, which is often part of the static HTML page. Other scripting languages are well suited when most of a page is static HTML, because the dynamic data can be put in directly the HTML page. But with XML-documents the whole document is dynamic data, so Servlets are very well suited. If a lot of HTML is needed Java Server Pages could be used instead of Java Servlets.

There is a Java Servlet for each page in the Website. The web server responds to the HTTP-request with a combination of an XML-file and a XSL-file (the advantages of this respond is explained in the preceding chapter 22. Website).

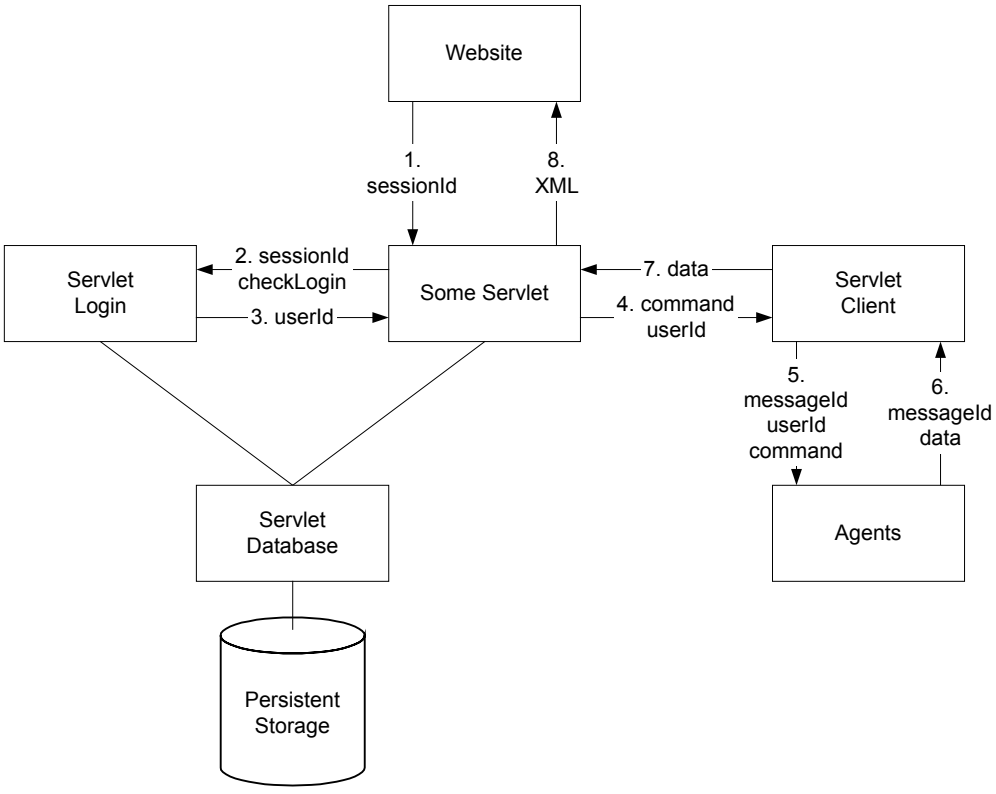

Figure 43 : Overview of the Servlets and the communication with the Website and the Agents.

When the Website (browser) sends a HTTP-request to the web server to get a Servlet that is part of the inTelly system, the sessionId is checked to see if the user is logged in (see Figure 43 1. and 2.). If the user is not logged in, the relevant data is sent back to the Website. If the user is logged in, then if it is necessary it is possible to communicate with the Agents, by sending some command and wait for an answer (4. – 7.). When the Servlet has sent a command it receives a message id for the current message. And if there is meant to be a response (some commands do not result in a response) then the Servlet wait for the response, otherwise the Servlet just sends the requested data back to the Website (8.).

There is made a static Database class that is used by several of the Servlets, including Login when it checks if the user is logged in. The other Servlets fetches data from the Database class when needed (explained further in section 23.2 Database).

The Servlets can handle requests from multiple browsers at the same time, these requests will be handled in parallel, and each of the requests starts a new Java thread.

23.1 Login

When a page is requested in the Website, the functions in the Login class are used to check if the user is logged in. There are different ways to check if the user is logged in.

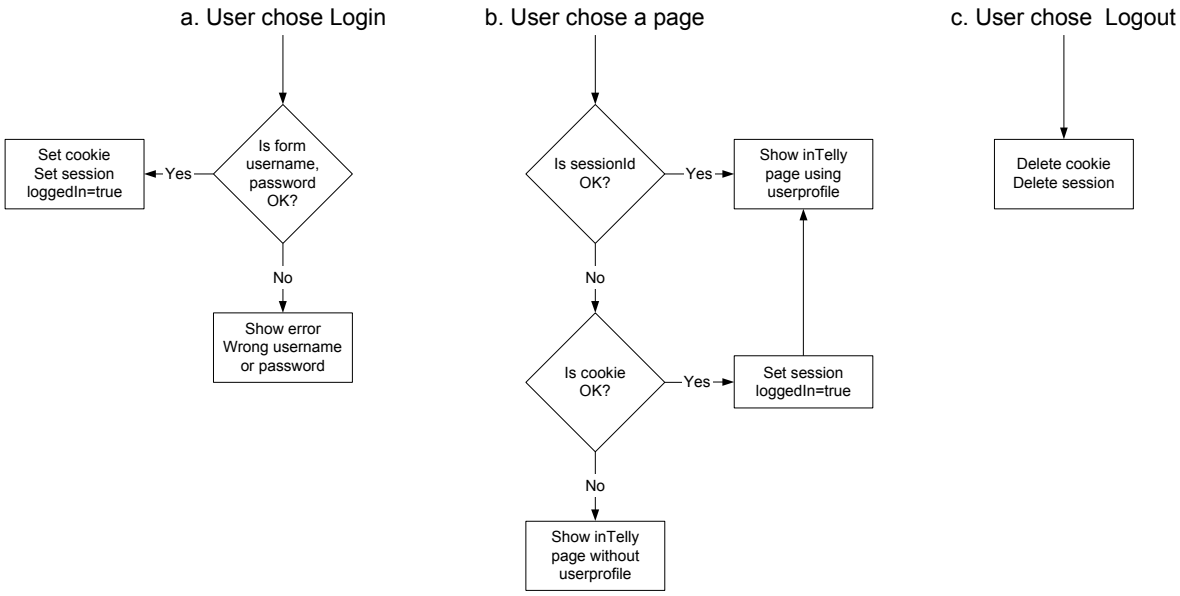

Figure 44 : Overview of Login and Logout. The user can login from the login page, but login is also checked when the user chose another page.

The user can be logged in via:

- Form variables
- Session variables
- Cookies

When the user has written his or hers username in the HTML-form at the login page and presses the login button (see 21.7 Login), the HTML-form is submitted and the form variables are fetched in the Login Servlet (see Figure 44 a.). If the username and password matches the ones stored in the database, the session variable and the cookie are set. And the user is redirected to the front page.

When the user requests any page, e.g. the front page, the first thing that happens is that the session id is checked to see if the session already exists (see Figure 44 b.). If the session does not exist, the cookie is checked for username and password. If the username and password does not exist or don't match in the database, the user is considered logged out, and the Servlet returns a page corresponding to a logged out user. If the username and password matches the ones stored in the database, the session variable loggedIn is set to true and a page corresponding to the user logged in is sent as a response. Then the next time the user request a Servlet in the same session the user is logged in via the session variable, which is quicker than making a database look-up.

If the user presses the Log out link in the top menu (see 21.1 Menu Bar and Logo), the Logout Servlet is called (see Figure 44 c.) and the cookie and session variable is reset, and the user is redirected to the front page.

23.2 Database

The Database class provide access for the Servlets to the database; this makes an indirect connection between the Agents and the Servlets. The Servlets gets the data for e.g. the Program list from the Database. The Database class provide functions for the Servlets to update, insert, delete and retrieve data from the database.

Some of the functions in the database class retrieves data from the database and returns the result in a string to the Servlet, which then sends it to the browser. Other functions, primary functions which returns a large amount of data, sends the data directly to the browser instead of sending it back to the Servlet, this way is much quicker, but not so nice programming.

23.3 Client

The Client class is used by the different Servlets to send commands to the Agents. When commands are sent to the Agents each message gets a unique id, so that the Servlets can recognize the response from the Agents.

The only function, which sends commands to the Agents in the inTelly system, is the function, which saves priorities for programs. Then the Agents can calculate a new AI-value if needed.

23.4 Test

The functional requirements for the system are not directly demands to the Servlets, but the demands for the Servlets can be derived from the functional requirements. The Servlets should be able to:

a. Respond to a HTTP-request with an XML-file.
b. Respond to a HTTP-request by executing a command.
c. Check if the user is logged in with form variables, session variables and cookies.
d. Insert, get, update and delete data in the Persistent storage.
e. Send a command to the Agents.

The Servlets are tested to check that they fulfil these demands. This is tested with the use of the finished Website and Agents.

Requesting a page in the browser starts the test:
http://intelly.dk

This returns a Program list page where the user is not logged in (demand a). There is presented a lot of TV-programs that are fetched in the database (partly demand d). A new user is created (partly demand d). The Program list is now shown with the user logged in (partly demand c). The help page is requested and thereafter the Program list again; this shows that the session variables is used to login (partly demand c). Then the browser is shut down (the session variables are deleted), and started again. The Program list is presented with the user logged in, this shows that the cookies are working (partly demand c). Some of the TV-programs are prioritised, and we wait 5 minutes. Then the browser is refreshed and the AI-values have changed, this shows that commands where sent to the Agents (demand e).

To check all of demand d, a user is created, the user profile is updated and the user is deleted again. And it is checked to see if it is possible to log in as this user.

23.5 Conclusion

The combination of Servlets and XML-documents is very successful. It has proved to be a good solution of making websites with dynamically changing data, and to separate the data from the layout.

There was not explicitly set any demands for the Servlets in the functional requirements (see 13. Functional Requirements), but there could be derived several demands for the Servlets from the functional requirements. The Servlets proved to fulfil all the demands made for them.

24. Agents

The Agents in the inTelly system are used to fetch TV-program data from an Internet information channel, and to prioritise these programs according to each users profile. This prioritising results in the AI-value shown in the Program list (see Figure 27). Apart from this the Agents are used for recognizing the Repeaters among the TV-programs, and update the user priorities accordingly. This is shortly the purpose of the Agents, which will be described further in this chapter.

The Agents in the system use the Agent framework to provide the basic functionality for the Agents, e.g. start agents, move agents, communication between agents, the Agent super class, etc. The understanding of these functionalities is important when reading this chapter. They are described further in the chapter 26. Agent Framework.

All the Agents are inherited from the Agent super class in the Agent framework, which means that they all have the same structure and also the same underlying technology. The Agent super class is to be seen as a reactive agent, which is an agent that is triggered by events observed in the surroundings (see chapter 3. Agents for details). When using this type of agent, the agents are actually not very far from ordinary object-oriented programming, this means that the agent technology could be explored and used much more than it is done in this project.

When a role in the application domain is located and the responsibilities and services associated with this role are found, there is basis for creating an agent. If an agent consists of several services or responsibilities, it can be considered to split this agent up in several agents with fewer services or responsibilities. An argument for splitting up an agent is if the agent consumes much computer power, because of a lot of calculations. An argument for not splitting up an agent could be that the services use the same data. The method for creating agents is described in the chapter 3. Agents.

By finding the different roles of the application domain, the Agents in the inTelly system is found and they are illustrated in Figure 45.

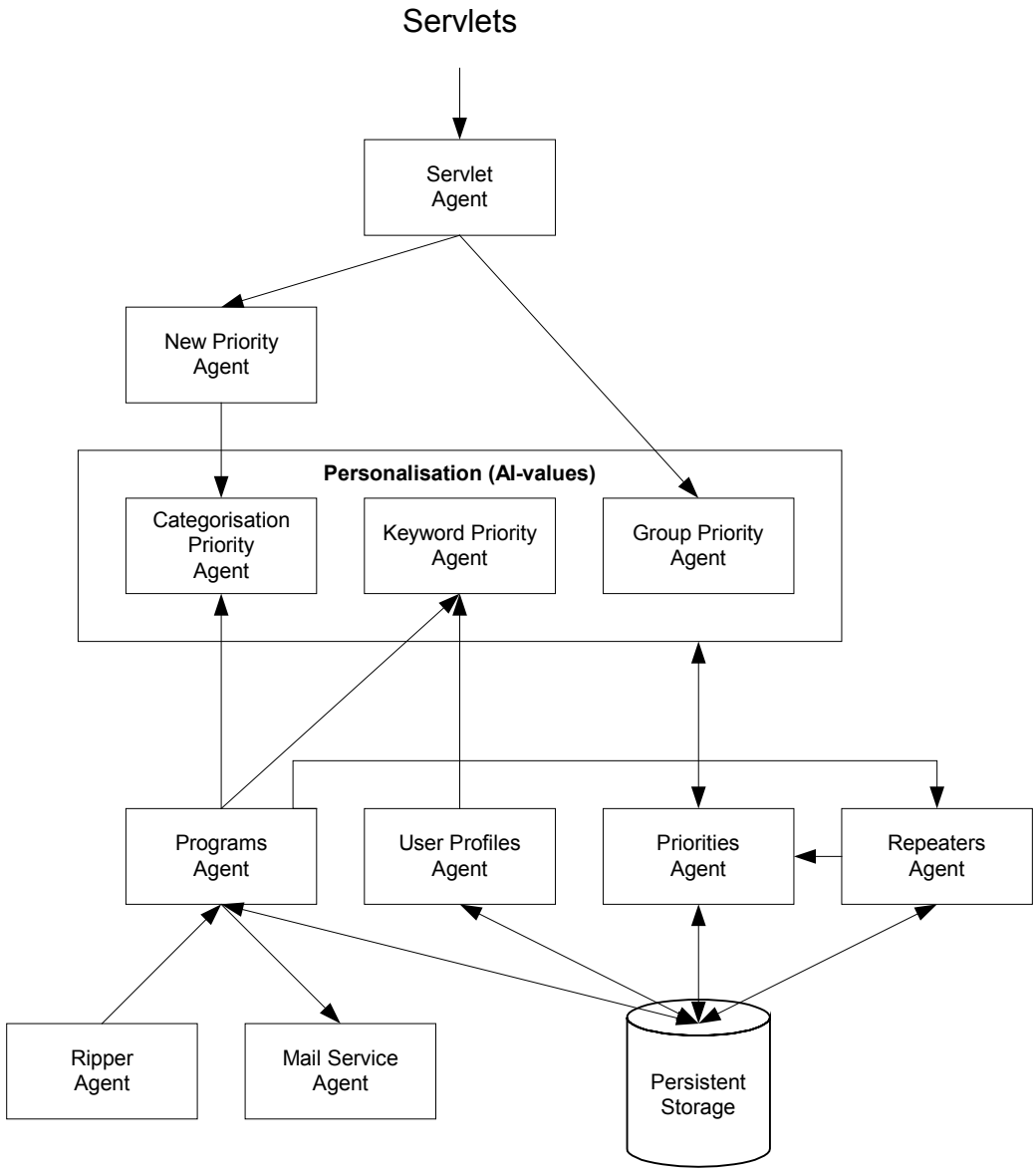

Figure 45 : The Agents in the inTelly system.

The purpose of the Servlet agent is to receive and answer requests from the Servlets; the only command that is actually sent from the Servlets is when the user has prioritised a program. When this happens the priority is sent to the New priority agent and the Group priority agent. The Group priority agent then updates the group priorities of the users that are in the same group and the New priority agent decides if the Categorisation agent should be signalled to start prioritising. The Personalisation agents calculate the AI-values for each user, these are calculated either when the user has prioritised some programs as just explained or when the Ripper agent has fetched new programs. When the Ripper agent has fetched programs, the programs are sent to the Programs agent and the Programs agent saves the data in the database and tells the appropriate agents that new programs has arrived. This is shortly a description of the primary agents.

The possible distribution of the agents in the inTelly system (see chapter 26 Agent Framework for details on distributing agents on several computers) should take into consideration which agents have a high level of communication and which agents require much computing power. The Personalisation agents, the Repeater agent and the Ripper agent requires a lot of computing power when new programs are fetched, so distributing them on several computers would be a good idea. The Servlet agent and the New priority agent should be placed on the same computer, maybe even the same computer as the web server to make the communication between the Servlets and the Servlet agent fast as well.

Each of the agents will be presented in more detail in the following sections. Each section will start with describing the purpose and responsibilities of the agent, then the services provided by the agent are described and the subscribing agents are listed. Then a list of the services the current agent is subscribing on is presented. And if necessary the needed information and external communication by the agent is described.

Most of the agent communication in the inTelly system is based on advertise-subscribe communication (see chapter 26. Agent Framework). An agent advertise a service, e.g. providing TV-programs, and other agents subscribe to this service and will e.g. receive the TV-programs every time the advertising agent fetches new TV-programs. When the system is started it is initialised by the doInitialise method in all the agents. A lot of the advertising and subscribing is done in the doInitialise method.

24.1 Events

In the agents there are only two events that cause actions, this is when the Ripper Agent fetches new programs at night and when the New Priority Agent receives user priorities.

These two events will be presented by the use of sequence diagrams, which shows the communication that happens when the events have occurred.

Program Fetched Event

When new programs are fetched from the TV-programs provider, the programs are sent to the Programs agent that saves the programs in the Persistent Storage (see Figure 46).

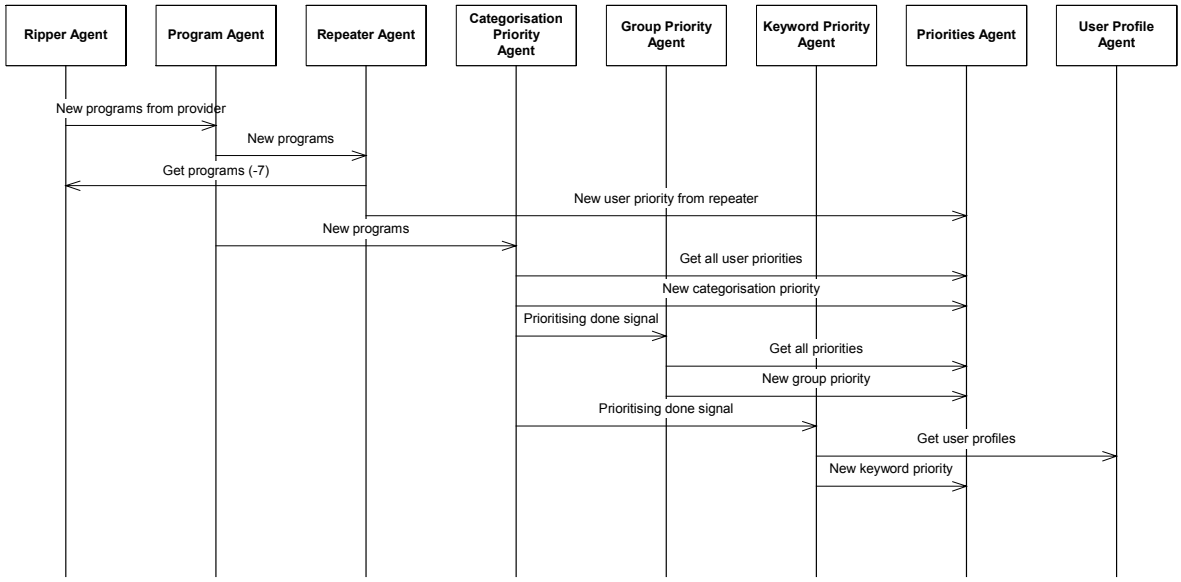

Figure 46 : Communication between agents when new programs have been fetched.

The Programs agent then sends the programs to the Repeaters agent and the Categorisation priority agent. The Repeater Agent starts recognizing Repeaters in the received programs and updates the user priorities accordingly. The Categorisation priority agent calculates priorities according to the earlier prioritised programs by the users, by categorizing the descriptions in the different programs (see the chapter 27. Categorisation for details).

When the Categorisation priority agent is done calculating the priorities it signal the other two Personalisation agents. The Keyword priority agent fetches the user profile from the User profile agent and looks for the user defined keywords in the description and title of the programs and updates the keyword priorities accordingly. When the Group priority agent receives the signal it starts grouping users according to earlier user priorities. Then the future programs evaluated by grouped users will be prioritised similar in the group priority.

New User Priority Event

When a user has prioritised a program and the specified amount of time has past, the Categorisation priority agent and the Group priority agent starts calculating new AI-values according to the new user priorities (see Figure 47).

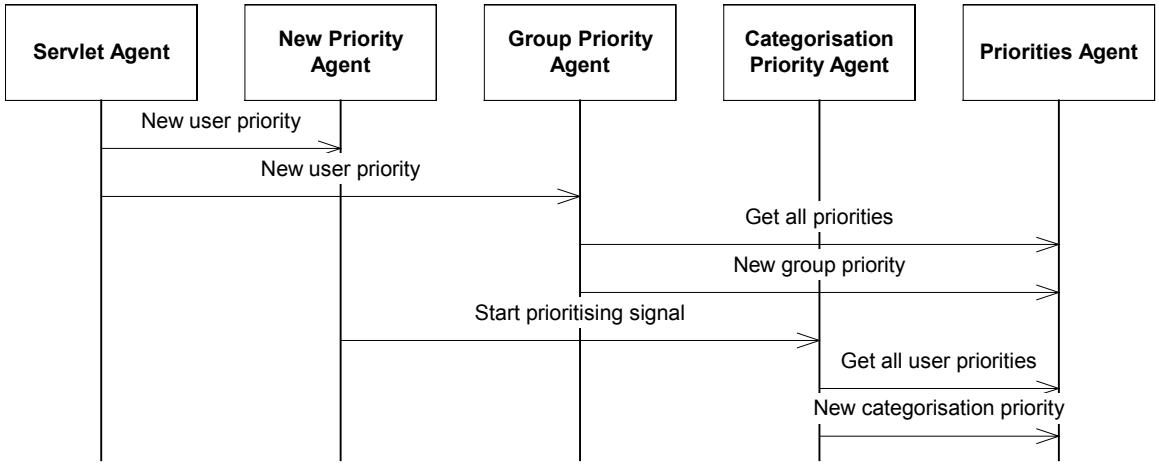

Figure 47 : Communication between agents when the user has prioritised some programs.

When the Servlet agent receives commands from the Servlets, the Servlet agent recognises the commands as new user priorities and the New priority agent and the Group priority agent are told because they subscribe for this service. The Group priority agent updates the user priorities for users in the same group and the New priority agent waits a specified amount of time, and then it signals the Categorisation priority agent to start prioritising. Then the Categorisation priority agent starts calculating categorisation priorities.

24.2 Servlet Agent

The responsibility of the Servlet Agent is to provide a link between the Servlets and the agents (see Figure 22), to receive messages from the Servlets and respond to them. The link is made with a TCP/IP connection where strings are exchanged. Each message exchanged is provided with a unique id, to assist in guiding the message back to the sender if there is a response.

Service	Description	Subscribers
New user priority	Inform the subscribing agents when there are new user priorities for programs.	New priority agent Group priority agent

In the inTelly system the Servlet agent is used to inform the agents that the user has prioritised a program. The Servlets does not wait for a response from the Agents.

24.3 New Priority Agent

The New Priority Agent observes the priorities done by the user, and informs the Categorisation agent when it has been a specific number of milliseconds (e.g. 5 minutes) since the user did the last prioritising. The reason for not calculating the categorisation priorities every time the user has prioritised a program is that this is a time consuming process.

Service	Description	Subscribers
Start prioritising signal	Signal the subscribing agents when a specific number of milliseconds have past since the last user prioritising.	Categorisation priority agent

Subscriptions on services	Advertiser
New user priority	Servlet agent

This simple intelligence could be build into the Categorisation priority agent, which then in a higher degree was able to control its own behaviour.

24.4 Categorisation Priority Agent

The Categorisation priority agent is the agent in the inTelly system that contains most intelligence; it uses this intelligence to calculate the categorisation priority, which is part of the AI-value shown to the user in the Program list (see Figure 27). The intelligence used for the calculating the categorisation priority is explained in detail in chapter 27. Categorisation.

Service	Description	Subscribers
New categorisation priority	Calculate the categorisation priorities for all users by categorising the program descriptions and titles.	Priorities agent
Prioritising done signal	Signal the subscribing agents when done prioritising.	Group priority agent Keyword priority agent

Subscriptions on services	Advertiser
Start prioritising signal	New priority agent
New programs	Programs agent
Get programs	Programs agent
Get priorities all users	Priorities agent
Get priorities one user	Priorities agent

The process of calculating the categorisation priority is a time consuming process when Neural networks are used, approximately 1-2 minutes for calculating all the user priorities for all users. A solution to this could be to use Bayesian networks, which is much faster.

The categorisation priorities removes in average approximately half the programs in the program list via the threshold value set to hide programs with an AI-value less than 50.

24.5 Group Priority Agent

The Group priority agent is part of the Personalisation agents that calculate the AI-values of the inTelly system. The group priority is found by comparing the priorities done by the users.

When several users have made a specific number of similar priorities (80% implemented) for the same programs they are considered a group. When one of the users in a group prioritise a program the group priorities for users within the group is set to the same priority for the same program. Groups are made at night when there have arrived new programs. The functionality of the Group priority agent is requested functional requirement 12 (see section 13.1 Functional Requirements to the TV-guide).

Service	Description	Subscribers
New group priority	When a new user priority is received it is propagated to the other users in the same group.	Priorities agent

Subscriptions on services	Advertiser
New user priority	Servlet agent

The group priorities are a canonical user profile where different users are divided into groups (see chapter 4. User Profiles for details).

24.6 Keyword Priority Agent

The Keyword priority agent is the simplest of the Personalisation agents. It uses the keywords the user has entered in the user profile and searches for them in the title or description of the different programs. When the agent finds a keyword, the keyword priority for this program is set to true.

Service	Description	Subscribers
New keyword priority	Search for user defined keywords in program descriptions and titles. If a keyword is found the according keyword priority is set to true.	Priorities agent

Subscriptions on services	Advertiser
Get user profile	User profiles agent

24.7 Repeaters Agent

The responsibilities of the Repeater agent are to detect Repeaters and update the user priorities accordingly. The Repeater agent detects the programs that are repeated. That is compare one program at a time, with the programs for one week ago. If two programs have the same title, channel and start time of day then it is considered to be a Repeater.

Subscriptions on services	**Advertiser**
Get programs	Programs Agent

The first thing that happens when new programs have arrived is that the programs are checked for the existing Repeaters, which are the programs that have earlier been considered as Repeaters. There is a table in the Persistent Storage that contains all the Repeaters. Another table contains the program ids and Repeater ids for each program that is detected to be a Repeater. When the new programs have been checked for the existing Repeaters, the remaining programs are checked for new Repeaters. When all the new Repeaters have been found, if any, the users priorities are updated according to the Repeater priorities for each user. If an existing Repeater is not detected a specified number of times, it is removed from the Repeaters list. The reason for not removing it right away, is that the program could be moved because of, e.g. a ball game.

24.8 Mail Service Agent

The responsibility of the Mail service agent is to send out mails to the users who have specifically requested to get a mail containing today's programs. The mail is sent at 6 o'clock in the morning and is made with XML and XSL in Java. The output of the XSL transformation of the XML is HTML, which is sent to the users. The user priorities and AI-values for each user is included in the program list sent to the users.

Subscriptions on services	**Advertiser**
Get programs	Programs agent

This functionality could easily be expanded further, to e.g. inform the user 10 minutes before a specific program starts. Another similar functionality could be to send SMS messages. It is also possible to make integration with outlook putting the different programs prioritised positively in the user's calendar.

24.9 Ripper Agent

The Ripper agent has the responsibility to fetch program data from an Internet information channel. The data can be available as XML-documents, HTML-documents or as a database

connection. The data is fetched each night just past midnight, it is only the data for the day seven days forward in time that is fetched. The data is fetched from some data provider like Ritzau (see Appendix J) or from each of the channels DR, TV2, etc.

Service	Description	Subscribers
New programs from provider	Sends the TV-programs to the subscribers. When the TV-programs are fetched from the data provider.	Programs agent

The data fetched are sent to the Programs agent that saves the data in the database.

24.10 Programs Agent

The Programs agent has the responsibility to save the programs received from the Ripper agent each night in the database. Then it informs the subscribing agents that new programs have arrived. The other agents in the system can ask the Programs agent for programs for a specific day.

Service	Description	Subscribers
New programs	Send the new programs to the subscribing agents.	Categorisation priority agent Keyword priority agent Repeater agent
Get programs	Send programs for a specific day to the requesting agent.	

Subscriptions on services	Advertiser
New programs from provider	Ripper agent

24.11 Priorities Agent

The Priorities agent has the responsibility to calculate the AI-values every time a priority is updated. It also manages the priorities table in the Persistent Storage. It inserts, updates and deletes priorities for the Agents in the inTelly system. The priorities are user, Repeater, categorisation, group and keyword priorities.

The categorisation, group and keyword priorities are calculated by the Personalisation agents and are used to calculate the AI-values shown to the user in the Program list. The calculation of the AI-values can be done in several ways, the categorisation priority and group priority is a number between 0-100, and the keyword priorities are true or false for all the programs. These three values should then be calculated to give an AI-value between 0-100. One way to

do it is to make a true in the keyword priorities worth a specific value, e.g. 20, and then add the three values and cut off if it exceeds 100.

Another way to calculate the AI-values is to use the categorisation value as the AI-value. And if there is a group priority this overwrites the categorisation priority, if there is a keyword priority the AI-value is set to 90.

The last method described is the one implemented. It should be tested in a longer period of time, which method is the best; by having several users use the system.

Service	**Description**	**Subscribers**
Get all priorities	Sends all the priorities for all users.	Categorisation priority agent
Get priorities for one user	Sends all the priorities for one user.	Categorisation priority agent

Subscriptions on services	**Advertiser**
New categorisation priority	Categorisation priority agent
New keyword priority	Keyword priority agent
New group priority	Group priority agent
New user priority	Repeater agent

24.12 User Profiles Agent

The User profiles agent makes the user profile available for the Keyword priority agent.

Service	**Description**	**Subscribers**
Get user profile	Sends the user profile to the subscribers when requested.	Keyword priority agent

The User profiles agent fetches the user profile in the database.

24.13 Test

Forcing the Ripper agent to fetch TV-programs tests the agents. When the data is fetched the other agents react, and the different Personalisation agents calculate AI-values. There is also found new Repeaters by the Repeater agent, this can be seen in the graphical user interface for the Repeater agent.

The other functionality of the Agents is that they should react when the user has prioritised enough TV-programs, so that it will make a difference for the AI-values. Prioritising several

TV-programs for one user and waiting until the Agents react tests this. It can be seen in the New priority agent that it receives a signal every time the user prioritises a program. And then the New priority agent signals the Personalisation agents and they start to calculate new AI-values.

24.14 Conclusion

The use of agents is functioning very well and gives a good modular design of the system. It is quite easy to construct an agent with the use of the Agent super class.

Different configurations of distributing the agents on several computers could be used to avoid the system slowing down because some agents use all the computing power. Load balancing agents could be made to automatically find the best possible way of distributing the agents.

25. Persistent Storage

In this chapter there will be a description of how the data in the inTelly system is stored. This includes both user data and TV-program data.

The possible choice of persistent storage for use in the inTelly system lies between the use of a file system or a database. The persistent storage has to support multiple user access, major data amounts and also the possibility of inserting, updating and deleting data from the storage. It has been chosen to use a relational database because the database allows multiple accesses to the data, it can handle huge data amounts and because there is a possibility of inserting, updating and deleting data.

Some of the available databases on the market are Access, Oracle and MySQL. The MySQL database is selected for this project because it is available for both Windows and UNIX/Linux, it has good online help features and because it is an Open Source database. The Open Source software makes it possible for anyone to use it for free.

25.1 Data

The data stored in the database are all kinds of data that are used by the inTelly system. The data that is necessary for the TV-guide to work is, e.g. TV-programs, Repeaters, priorities, user profiles etc.

The somewhat large amount of data in the system comes from the TV-programs and the system priorities. The TV-program data for one day will approximately consist of:

- 3000 programs per day (100 channels each with an average of 30 programs per day).
- Approximately 250 characters per program (the average of a days programs).

When this is summed up it gives a total of about 750.000 characters per day for all the programs and on a weekly base this gives 5.250.000 characters that have to be stored in the database. These numbers give rise to some speculations about the data type to be used in the design of the tables. The part of the data for a TV-program that varies most and is the largest part of the data is the detailed description. When stored there should be used a variable data type to save as much hard drive space as possible. When the data is no longer used is should be removed from the database.

The system priorities are the other source for large data amounts. When the system receives new TV-program data (e.g. for a new day, seven days ahead) it will calculate a system priority for each program and user. This will give 3000 new rows in the database per user per day and now imagine that the system then has 10.000 daily users. This will result in 30.000.000 rows

per day. Also here the data type should be considered because of the many rows in the database per day.

The examples above have showed that there can be expected an enormous traffic to and from the database. A solution to this could be to use real-time AI instead of letting the system process the data at night as it is done right now. With real-time AI it will be possible to process the data right when it is needed and so it will not be necessary to store all these system priorities for each single user. The advantage will be that the weights from e.g. the Neural network are the only data that have to be stored/updated once per day for each user. This means reducing the number of daily rows from 30.000.000 to about 10.000. It should be noticed that scalability of the system is not be considered in this project.

25.2 Database Design

In the inTelly system there are two databases, one for user and TV-program data and one that contains information that is used by the Categorisation. The first database is named intelly and the structure of this database is seen in Figure 48 and the other is named personalisation and can be seen in Figure 49.

inTelly Database

The inTelly database contains data about the user and the TV-programs. It is chosen to store the data in eight different tables. The structure of the database is shown in Figure 48. The marked fields in the figure are indicating that this is the primary key for the particular table. The relations between the tables are drawn as lines.

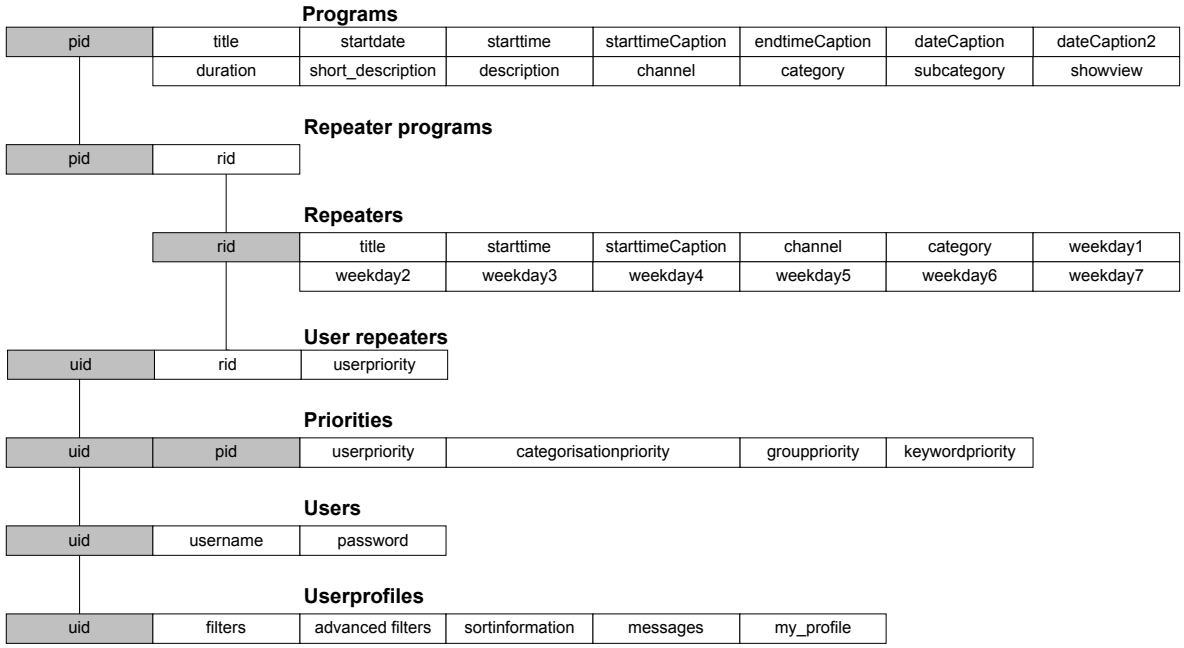

Figure 48 : The database structure for the inTelly database.

After having presented the structure of the database, the tables will be presented more thoroughly below.

Programs

The programs table encloses 15 columns that hold data about the TV-programs. The pid column holds the unique program id, which is generated by the *not null* and *auto increment* functions. The other columns contains the data about the TV-programs, e.g. title, start date, duration etc. This table also contains columns like starttimeCaption, dateCaption1 etc. and these are similar to the starttimeCaption presented in the repeaters table. The most interesting of the columns in this table is the description column that contains the detailed description of a program. The data in this column can vary from a few characters to about 300 or 400 characters per program, which therefore demands for the a data type with variable length. In this case the TEXT (see Appendix G) type was chosen.

Repeater programs

This table holds the relation between the single registered repeater and the actual program id. There are two columns containing the unique repeater id and the unique program id.

Repeaters

There is generated a unique repeater id by the *not null* and *auto increment* features in the database. The table also holds information about the repeater/program itself like title, start time etc. The more interesting columns in this table are the starttimeCaption and weekday1-7 columns. The starttimeCaption is an alternative way of representing the start time of the program. It is stored in the database because it is faster to retrieve from the database than generate it from the "original" starttime when the data has been retrieved. The weekday1-7 columns contain data regarding a repeater if it is shown or not. If a Repeater is shown Monday the value of weekday1 is set to 0, if the Repeater is not shown the next Monday weekday1 is increased to 1, and the next week 2, etc. If the Repeater is not shown at Tuesday, the weekday2 column is set to –1, etc.

User repeaters

The data about the Repeaters that the user has prioritised is stored in this table. It contains the unique user id given by the users table and the unique repeater id, which is identifying each repeater and the user's personal priority of a repeater.

Priorities

This table stores information about the users and the systems evaluation of TV-programs. The unique id in this table is a combination of the unique user id given by the users table and the unique program id given by the programs table. The table has a column for each kind of priority.

Users

The users table contains an id, username and password for each registered user of the inTelly system. The user id column uses the *not null* and *auto increment* features to generate a unique id for identifying each user.

User profiles

This table hold data about each user. The data in this table is e.g. the users email address and whether the user wishes to receive an email or not. The id in the table is the unique user id, which is given by the users table.

Personalisation Database

The personalisation database contains data used by the artificial intelligence. The data is stored in 2 different tables. The structure of the database is shown in Figure 49.

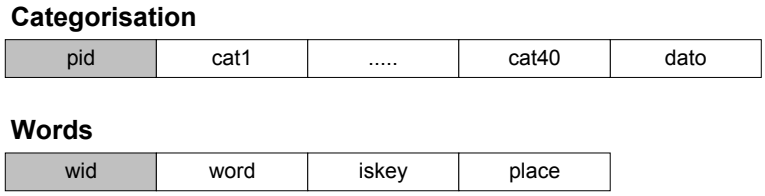

Figure 49 : The database structure for the personalisation database.

After having presented the structure of the personalisation database, there will be a presentation of the tables below.

Categorisation

The cat table encloses 42 columns that hold the necessary data. The 40 columns in this table contain the counts of matches for each of the categories. There will be a row for each of the programs in the programs table. The table also contains a date and the unique program id given by the programs table.

Words

This table is mainly to be seen as the reference vector used by the Categorisation (see section 27.2 Method). The table contains a list of words (keywords of the 40 reference films) used by the artificial intelligence to categorise the given program.

25.3 Test

It is found that the database is able to create and drop tables and it is also possible to insert, update and delete data from the different tables. It is possible for several users to use the system simultaneously, which indicates the possibility multiple user access.

The time for retrieving the 3000 programs has been measured and was found to be approximately 0.1 seconds.

25.4 Conclusion

The database has been tested and it is found that it support the demands mentioned earlier. It is also found that the database has a lot additionally functionalities that might be used to optimise it, e.g. for searching.

26. Agent Framework

This chapter contains a description of how the Agent framework is designed, with Agent Servers, Agents, Administrator agents, Facilitator agents and their graphical user interfaces. The Agent framework is used by the Agents described in chapter 24. Agents. There are made some functional requirements for the Agent framework, these are listed in section 13.2 Functional Requirements to the Agent Framework.

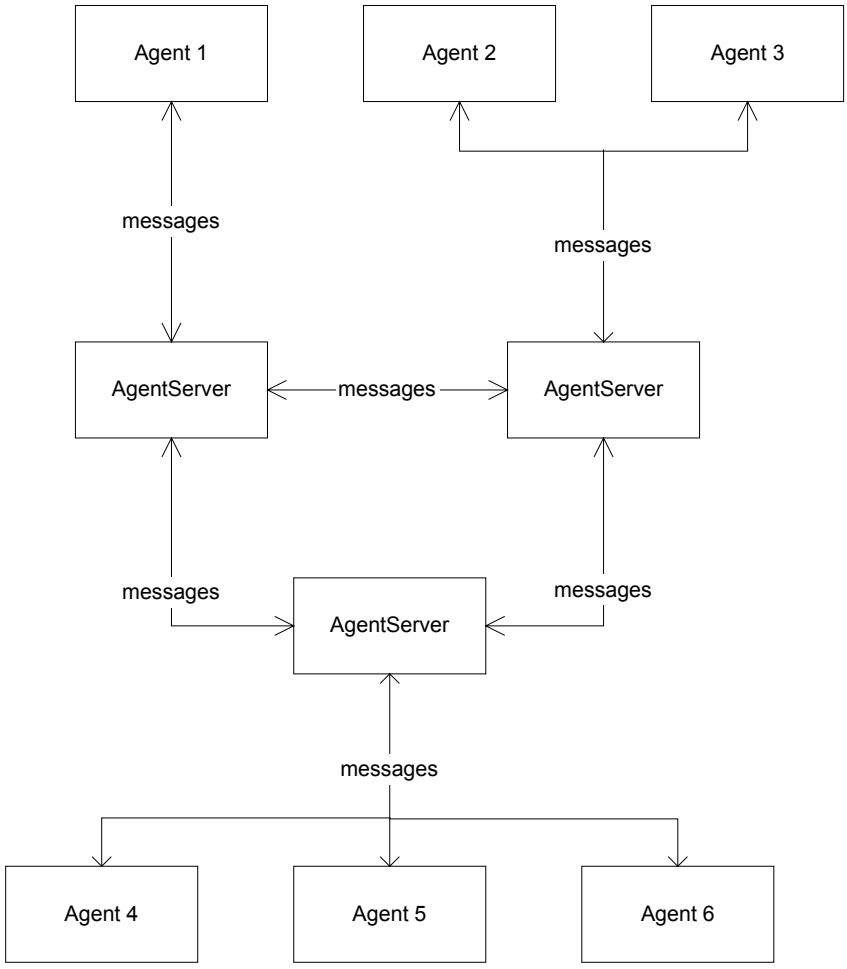

Figure 50 : Communication between the Agents via the Agent Servers.

The Agent servers can be located on different computers, which make it possible to distribute the workload of the agents on several computers. It also makes it possible to distribute agents in geographically distributed computers making it possible to control agents from different places. The Agent servers run as Java applications and can run in both Windows and Linux.

The agents are located as threads on the Agent servers; the placement of the different agents should be decided by how much the agents communicate with each other. Agents that communicate much together should be located on the same Agent server because the communication between two agents on the same Agent server is based on function calls instead of TCP/IP, and the function calls are much faster. When two agents communicate they

send a KQML-message, which holds the name of the receiving agent (see Appendix D for details on KQML). Then the Agent servers deal with the administration of the messages. The Agent server where the agent is located sends the message to the Agent server where the specified agent is placed, and delivers the message to the specified agent. The placement of the agents on different Agent server is transparent for the agents when sending messages. In this way it is possible to move the different agents around from server to server without the agents have to worry about sending the messages to the correct Agent server. This makes the agents mobile, which is one of the functional requirements for the Agent framework (see the section 13.2 Functional Requirements to the Agent Framework).

The Agent framework is made as a Java package that can be used in other projects without any altering. The class diagram of the Agent framework is presented in Figure 51.

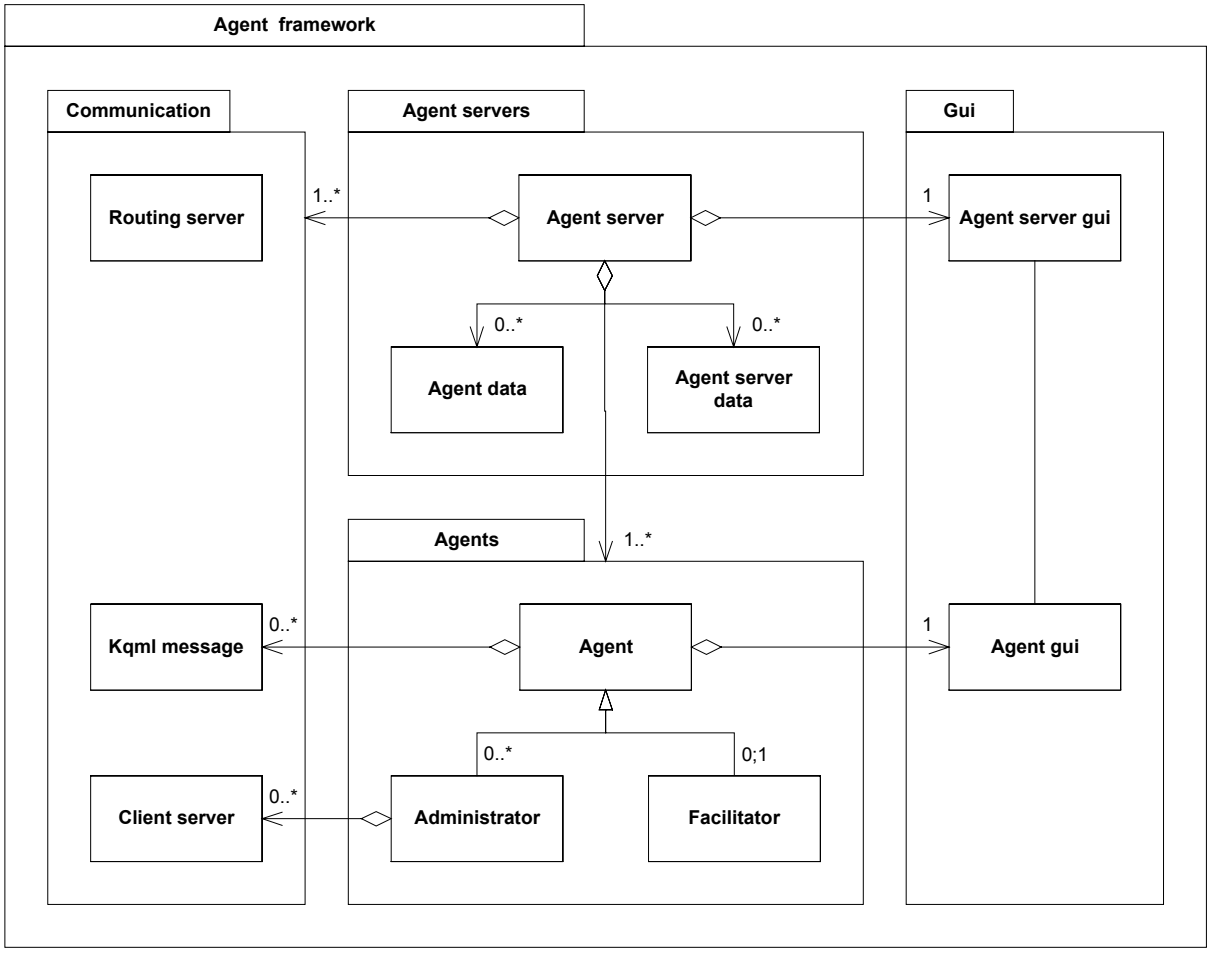

Figure 51 : Class diagram of Agent framework.

In this chapter there will be a description of the packages in the Agent framework and the classes within them. The Agent server uses all the classes in the communication packages. There is always one Routing server present on an Agent Server. There is also an Administrator agent present on each Agent server and only one Facilitator agent is present in the entire Agent framework.

The graphical user interface to the Agent server and Agents has not been part of the User-Centred Design method, because this is intended for different users, e.g. administrators, these user interfaces will be described in this chapter.

All agents are subclasses of the Agent super class, including the Administrator and Facilitator.

The communication between the agents is based on the agent communication language KQML (see Appendix D for details), which is implemented in the Kqml message class. The Agent server and the Agent use this class to exchange messages.

26.1 Events

When using the Agent framework several events happens, when starting an agent, moving an agent, killing an agent, sending messages, creating Agent servers, etc. As an example of an event, the Move agent event is described (see Figure 52).

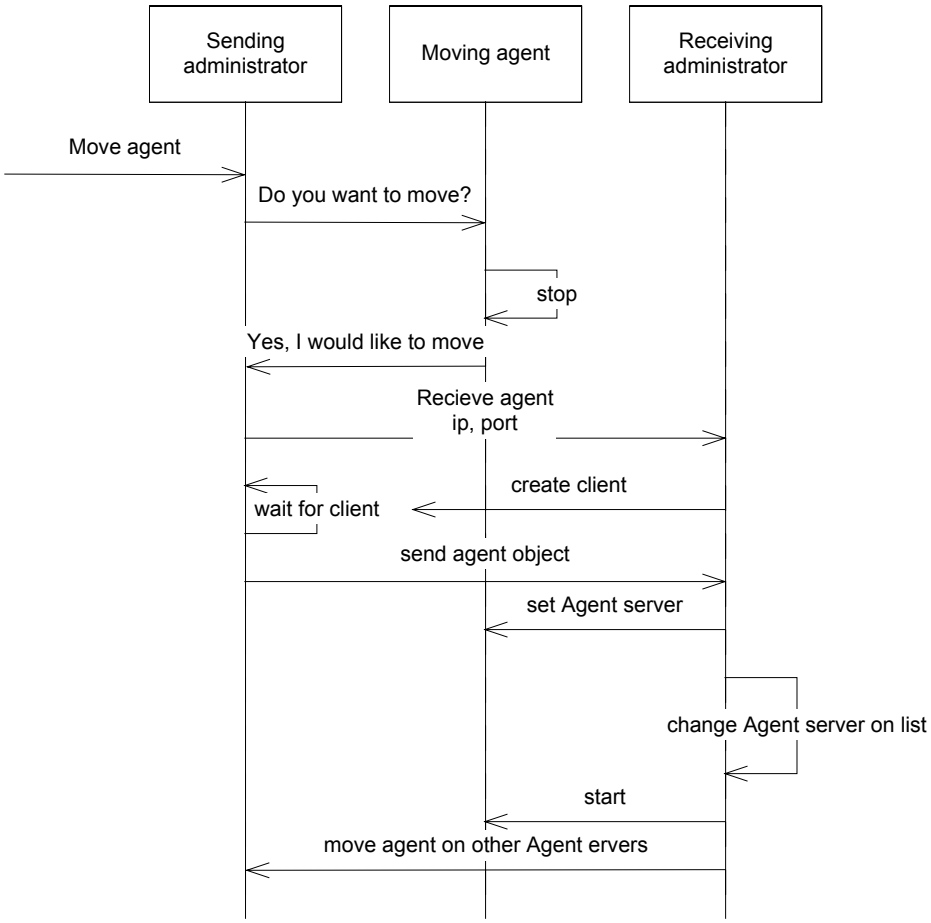

Figure 52 : Move agent event.

The user can move an agent by use of the Agent server user interface. Otherwise agents can also move themselves or other agents. When an agent is moved the Administrator agent on the Agent server where the agent is located is contacted. Then the moving agent is told to

move and if it is willing to move it stops running. The agent is then send through a TCP/IP-connection to the new Agent server, where it is started again. Then the graphical user interfaces and the lists in the Agent servers are updated.

26.2 Agent Server

The Agent server can be populated by agents, and provide communication between the agents based on KQML. One of the Agent servers in the inTelly system is made a master that the other Agent servers make the initial contact with. Each Agent server has an administrator agent and a routing server.

Agent Server

The Agent server contains a list of all the Agents in the system and a list of all the Agent servers in the system. This way all the Agent servers know where a specific Agent is placed. If the systems are not too large, it is quite possible to administer all the Agents in this manner. If the Agent framework is used for systems with several thousand agents it should be reconsidered to make the administration in another way, or splitting the Agents up in two or more Agent frameworks.

The Agent server is used to control the Agents and Agent servers in the system. It is possible via the graphical user interface (see section 26.5 GUI) for the Agent server to create, move and kill Agents among other things.

Agent Server Data

The Agent server data class holds information about one Agent server. These data are sent to the other Agent servers in the system. The variables contained are:

- Name
- Ip
- Port
- ClientServer

The ClientServer is an instance of the ClientServer class in the Communication package. This holds a TCP/IP connection to the Agent server. The reason for making such a data class is that it is much easier to administrate when sending data about the different Agent servers.

Agent Data

The Agent data class is much similar to the Agent server data class; it contains data about an Agent instead of an Agent server. The variables in this class are:

- Name

- Agent server name
- Object

On the Agent server that holds the Agent, the Object variable contains the actual instance of the Agent. On the other Agent servers this variable is set to null. This is the actual thread that holds the Agent. This is used to administrate the Agent and send and receive messages through and sending an agent.

26.3 Agent

An agent should provide some basic functionality that is requested for agents (see 3. Agents for details). The Agent servers provide the ability for the Agents to be mobile, by making it possible to move an Agent from one Agent server to another. This makes the Agent able to move from one computer and environment to another.

Agent

The agent super class is a class, which a programmer can use to make new agents. Each agent has some predefined methods, which should be filled out with the functionality and communication specific for this agent. These methods are:

- doInitialise
- doRun
- doAction
- doKqml
- doXml

The doInitialise method can be used to initialise the agent, it is only called once when the agent is started. It is mainly used to make advertisements and subscribtions. The doRun method contains the functionality of the agent; this is called again and again when the agent is running with a specified interval. The doAction is a method that can be called by the user from the user interface of the agent, this makes it possible for the user to force an event, e.g. making the Programs Agent simulate that new programs has arrived. This functionality could be enhanced to make it possible to call different actions in the same agent. The last two methods: doKqml and doXml is used to take action on the different messages received from the other Agents.

To make a new agent, the only things needed to do, is to make a new class that is inherited from the Agent super class and the different methods mentioned should be overloaded.

Administrator Agent

On each Agent Server there is an Administrator agent present, they are created in the constructor of the Agent server and cannot be killed. The Administrator agent takes care of creating, moving, killing, etc. agents on each Agent server.

When moving an Agent from one Agent server to another, the Administrator agents on the two servers makes a TCP/IP connection using the ClientServer class in the Communication package. The sending Administrator agent stops the Agent from running in its thread, and serialises the Agent, this serialised object is then sent through the Client-Server pair between the Administrator agents. The receiving Administrator agent on the new Agent server then starts the agent again.

Facilitator

The purpose of the Facilitator agent is to administrate the advertisers and subscribers of services in the system. When an agent needs a service, it asks the facilitator agent for the name of an agent that advertises the needed service.

The scenario that happens when an Agent advertises a service and another Agent subscribes for that service is illustrated in Figure 53, this is based on the communication described in chapter 3. Agents in Figure 7.

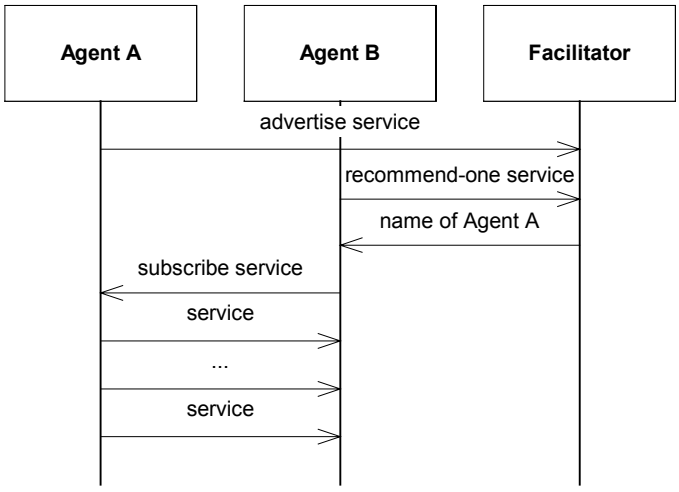

Figure 53 : Sequence diagram for advertise-subscribe communication between agents.

When the subscription is in place, the subscribing agent will in the future receive this service, e.g. the Categorisation agent subscribes for TV-programs from the Programs agent and receives the programs every time new programs arrive. If an Agent does not want to subscribe for a service any more, it can send an unsubscribe message to the advertising agent.

The facilitator holds two vectors; one containing the advertising agents and their services, and one holding agents who has sent a recommend message and has not received a name on an advertising agent because no agent has advertised this service. When an agent advertises the requested service the agent that has requested the service is contacted.

26.4 Communication

In the Agent framework different means of communication is used. There is made classes for creating Client-Server pairs and there is made a class for creating KQML-messages. The Routing server is used to administer the client-server pairs.

RoutingServer

Each Agent server has a Routing server which listens at one port for incoming request, e.g. a new server is being created and wants to register with the Agent server master. Then the Routing server delegates a port for the new Agent server to use to communicate with the Agent server master, then this port is open for communication between the two.

ClientServer

The ClientServer class is used to make either a client or a server in a Client-Server pair. Which of the two that is created is based on the constructor called, if the ClientServer class is instantiated with an IP-address a client is created and it connects to the server at the specified IP-address. If the ClientServer class is instantiated without an IP-address a server is created with the IP-address of the computer where it is started.

KqmlMessage

The KqmlMessage class is an implementation of the KQML standard described in chapter 3. Agents and Appendix D. An object of the class is constructed with a text string containing the KQML-message, the class parses the message to check if it is a legal KQML-message and prints out an error message if not.

The KQML-message for subscribing for the TV-programs from the Programs agent is as follows:

```
String recommendString = "(recommend-one "
                  + " :sender " + this.agentName
                  + " :receiver facilitator "
                  + " :reply-with idE "
                  + " :language KQML"
                  + " :ontology kqml-ontology"
                  + " :content (ask-all"
                           + " :sender " + agentName
                           + " :receiver Dummy"
```

```
                                + " :language xml"
                                + " :ontology Programs"
                                + " :content (<programlist><program>)"
                                + " :in-reply-to dummy"
                                + " :reply-with dummy))";
    KqmlMessage recommend = new KqmlMessage(recommendString);
    sendMessageToAgent(recommend);
```

First the string is created, then an instance of the KqmlMessage is created with the string, and then the message is sent.

26.5 GUI

The GUI package contains the graphical user interfaces for the Agent server and Agent in the Agent framework. This user interface has not been part of the user interface developed using the User-Centred Design method. This has been done because of time constraints and because this user interface is intended for a different user group, e.g. administrators.

Agent Server GUI

The graphical user interface of the Agent server is to be used for administering and controlling the Agents and Agent servers in the Agent framework. It is possible to monitor the actions performed by the Agent server and to see which Agents and Agent servers are present in the system.

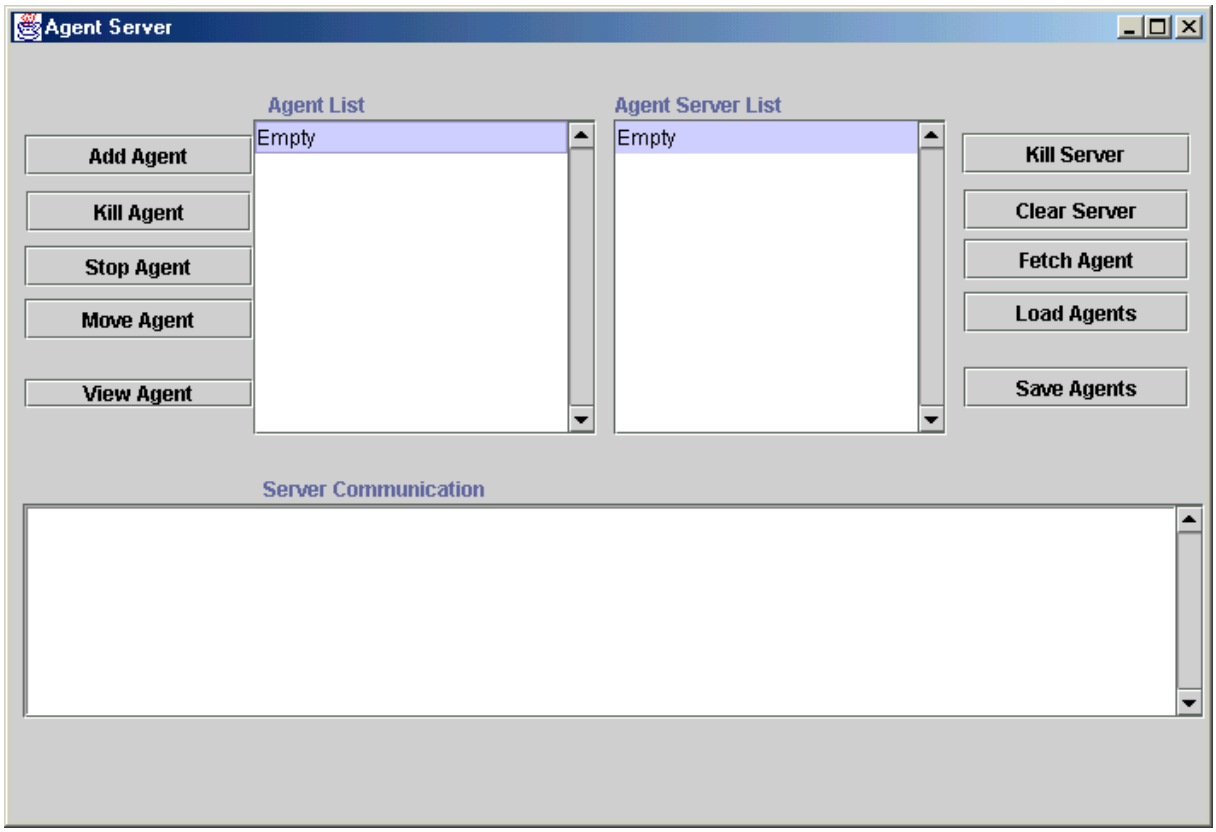

Figure 54 : Graphical user interface of the Agent Server. In this user interface it is possible to control and administer the agents and the agent server.

In the left side of the Agent server graphical user interface there is present some buttons for administrating the different Agents present on the different servers. When moving an agent, the agent is marked in the list, and the Agent server where the agent is supposed to move to be marked in the Agent Server list. Then the Move Agent button is pressed and the agent moves. The View agent button is used to view the status of the marked agent; double-clicking on the agent in the list can also do this. The status that is showed is presented as the graphical user interface presented in the next section.

The functions for administrating the Agent servers are presented in the right side of the user interface. The Clear Server button makes all the agents on one Agent Server move to another Agent Server, this function could be used when a computer has to be shut down, e.g. for maintenance. The Fetch Agent function fetches one agent to the current Agent Server.

The functionality for loading and saving agents has not been implemented. But could be implemented easily much the same way as the moving of an agent. Instead of sending the agent to a TCP/IP-stream it should be sent to a file-stream.

Agent GUI

The graphical user interface for an Agent is used to observe and administer the messages and actions performed by the agent. It is also possible to start an action implemented in the doAction method in the Agent.

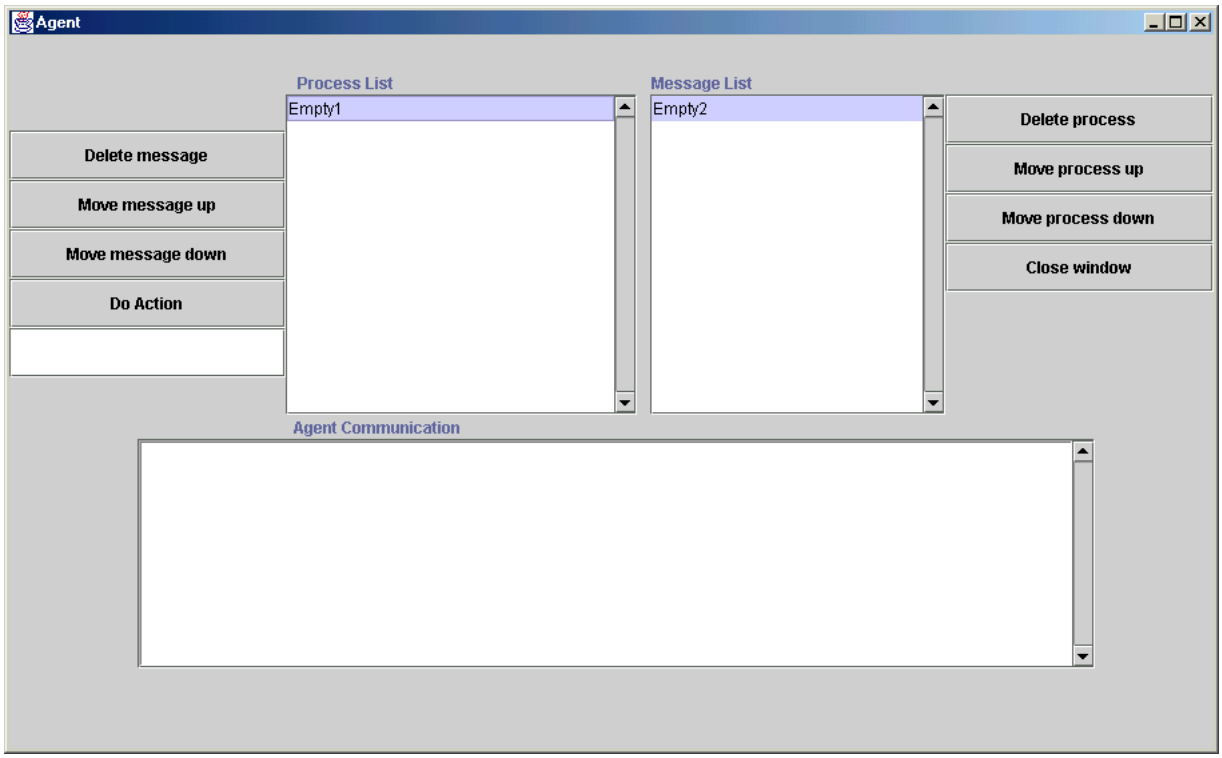

Figure 55 : Graphical user interface of an Agent. Each agent has this general user interface, where e.g. the communication can be observed. It is also possible to make a unique user interface for each agent, with specific features for that agent.

The only implemented function in this interface is the Do action function that calls the doAction method in the current agent. The text field is used to call the doAction method with a parameter, e.g. which day the Programs Agent should send data from.

The Agent has an internal process vector, which contains different tasks the agent should perform according to the type of agent. These processes the Agent will perform as quickly as possible and there could be set some priorities on these processes, so more important processes are executed before not so important processes, this functionality is although not implemented but the agents are prepared for it.

When the Agent receives messages from the other Agents, these messages are put into the message list, if the message contain some task to be done (they mostly do) then the content of the message is put in the process list.

26.6 Test

To test the Agent framework three Agent servers are started on three different computers. On one of the Agent servers an Agent is started, this Agent is a test Agent that does nothing besides writing a line every second in the standard output stream and wait for messages from other Agents. Then it is possible to see that the agent is running. The next thing that happens is that the Agent is moved from one Agent server to the next, now it can be seen that the Test agent starts to write at the other computer. The first Agent server is shut down and the other two Agent servers are still running. On the third Agent server another Test agent is started, and this agent sends a message to the first Test agent. It can then be seen that the first Agent receives the message. The third Agent server is cleared for agents and is killed. The two agents are killed.

26.7 Conclusion

The Agent framework makes it possible to make Agents in a quick and comprehendible manner, a programmer does not have to know much to make an agents and do not have to know anything about the Agent server. The agents provide delegation, intelligence, communicative, reactive and autonomous behaviour, and provide the extra functionality of mobility (see chapter 3. Agents).

There is not designed any intelligence to make the agents them selves control which actions they want to perform. The agents in the form they have in the inTelly system, is not much different from ordinary objects, apart from their ability to move to other Agent servers (mobility) and their ability to operate without any input from a user (autonomous). But this control of the Agents own behaviour is easily added to the Agent class.

27. Categorisation

The purpose of the categorisation is to be able to sort the TV-programs according to the users interests and to be able to filter out TV-programs that are irrelevant for the user. The interests of the user are defined by the earlier prioritising done by the user.

The result of the categorisation is the categorisation priority, which is part of the AI-values presented in the Program list. The AI-values is calculated from the: categorisation priority, group priority and keyword priority. When the AI-values are calculated the TV-programs can be sorted by these values, and a threshold can be set to filter out irrelevant TV-programs.

This chapter presents the considerations and development of the categorisation priority. The development of this intelligent evaluation consists of several advanced topics, which easily could be split into a number of projects itself. It is therefore necessary to narrow down the number of investigated solutions and methods to test and implement in this system. When possible solutions and methods are left out for further investigation it will be clearly stated.

The rest of this section will contain the following issues in the presentation of the method developed:

- A short introduction to artificial intelligence.
- An introduction to the method that has been developed this project. This includes artificial intelligence, categorisation and interests of the user concerning TV-programs.
- Presentation of the implemented method.
- Test of the chosen solution.
- Conclusion.

27.1 Artificial Intelligence

It is difficult to define artificial intelligence. One of the problems is that intelligence itself is not very well defined or understood. The problem of defining artificial intelligence is therefore first of all a problem of defining intelligence. This means to cope with answers the following questions and many others:

- What happens when learning occurs?
- What is intuition?
- Can intelligence be measured on the observable behaviour only or does it require a particular internal structure?

This discussion will not be an issue in this project and for that reason the definition used in this project will be:

"Artificial intelligence (AI) may be defined as the branch of computer science that is concerned with the automation of intelligent behaviour". [GFL, p.1]

The list of artificial intelligence applications is very long (expert systems, control, natural language understanding, pattern recognition, planning, machine learning etc.)

27.2 Method

In this project there has been put weight in making the creation of the user profile as simple as possible. The main reason is that users in the Observation test pointed out that this is an important issue. This means that the necessary information needed to filter and sort the TV-programs according to the user's needs should be collected with a minimal disturbance of the user. It has therefore been chosen to combine the integrated notepad with the collection of user interests. As already described (in section 21.4 Program List) the user can put programs that he or she will see or maybe will se in the notepad. Furthermore the user can remove TV-programs from the list by selecting "Do not want to see". This means that the system has three different levels of evaluations given by the users. In addition to this information the system also has access to the features of the TV-programs: title, detailed description, etc. The user profile information (user priority) and the features of the TV-programs will be used in order to make a categorisation priority of TV-programs, which will be presented in the "AI"-column in the user interface.

TV-program Features

In order to determine the characteristics of a TV-program it is necessary to extract some features from the program. The possible features that can be extracted are dependent upon the available data for each TV-program. The typical features available for each TV-program are listed below:

- Title
- Channel
- Category (film, news etc.) and in some situation also a subcategory (films: thriller, action etc).
- Start date
- Start time
- Duration
- Description in plain text (some TV-programs does not have any description).

In this project only TV-programs containing a detailed description will be considered in the generation of the categorisation priority. This means that TV-programs with no detailed description always will get an AI value of '50'. There could naturally be developed methods

that can categorise the TV-programs from the duration, time of day, category etc, but this will not be an issues in this project.

The feature extraction of TV-program data will concentrate on the title, the category and the detailed description. The methods to categorise this text data will be considered in the following section.

Text categorisation

In this project a simple text categorisation algorithm has been developed. The method is based upon the assumption that TV-program descriptions are using a relatively narrow amount of different words in each category. Therefore a simple method of categorisation has been chosen.

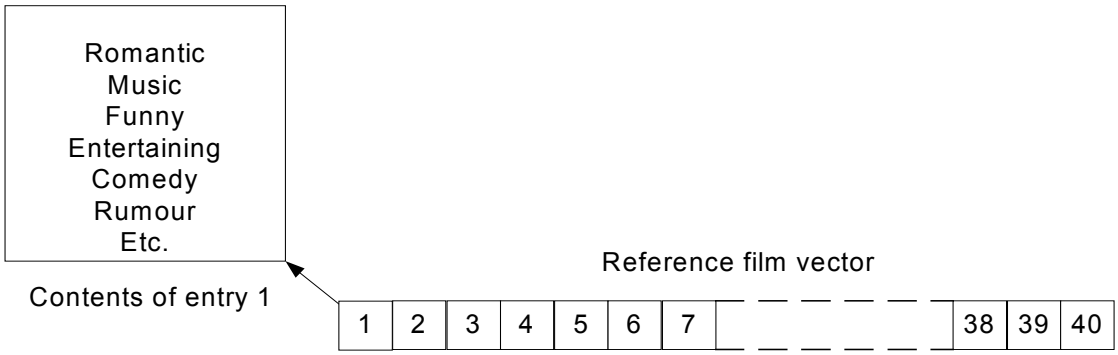

Figure 56 : An example of the reference vector. Each entry in the reference vector contains keywords from a film.

This method consists of 40 reference TV-programs in each category (only programs from the category of films will be developed and tested, but the principle is considered to easily adapt to any category). The features, keywords, of the 40 reference films are extracted manually. The reference films forms a vector, where each entry consists of the keywords of one film. See Figure 56 for an example. The 40 reference films have been chosen in order to make a compromise between minimizing the dimension of the feature space (a large feature space will result in a corresponding increase of computational time) and add more features than are already given by TV-data provider in terms of film subcategories (action, thriller, etc).

The reference films have been manually selected in order to ensure that they represent all film categories. When a new film is to be categorised there is made a search in the keywords in the reference vector and the number of occurrences are summed up for each entry in the vector. An example (description: “A romantic comedy from 1987.”) of a categorisation of a film containing 2 words placed in entry 1 is shown in Figure 57.

An example of a categorised film vector

Figure 57 : An example of a film categorised by the reference vector. The number in each entry corresponds to the number of matching words between the reference film vector and the description of the categorised film.

The keyword match has both been tested by exact match and by a simple heuristic word stemming, which is described below in the Film Categorisation section.

The pattern obtained in the reference vector will be used to determine how relevant a film is to a user based upon previous priorities.

Learning and Prediction

The method chosen consist of two parts:

- **Learning**: This will include collecting and interpreting the user priorities and the corresponding categorisation vectors.
- **Prediction**: From the above learning the system must predict the users interests concerning future TV-programs.

There are several methods to be considered in order to learn and predict the user priorities: Rule based systems, Fuzzy logic, Neural network, Bayesian network, etc. It has although been decided only to consider Neural and Bayesian networks in this project. The reason is that they are both considered to be well suited in the task of learning and predicting from the features given by the reference vector. The method to be used will be closely related to pattern matching and statistical reasoning with uncertainty, which are applications that are naturally related to Neural and Bayesian networks. It is therefore considered interesting to compare the two methods. In Appendices H and I there is a short presentation of Neural and Bayesian networks.

Implementation

The Neural and Bayesian network has both been implemented in Java. The main characteristics are presented below:

Neural network

- In the implemented Java classes it is possible to construct multi layer Neural network with one or two hidden layers.
- The number of nodes in each layer can be selected arbitrary (min. 1 node in each layer).

- There has been implemented a Matrix class in order to simplify the calculations in the feed forward and back-propagation methods [HA, p.68-82]. The Matrix class contains standard methods for matrix calculations (addition, subtraction, multiplication, transpose, dot product).
- A multi layer network is constructed by defining the following input parameters:
    - Number of input nodes.
    - Number of hidden nodes in first hidden layer.
    - Optional: Number of hidden nodes in the second hidden layer.
    - Number of output nodes.
- The feed forward method has an input vector as the only input parameter (a Matrix object). The vector size must correspond to the number of input nodes. The method returns a Matrix object containing the output vector after a feed forward through the network. The sigmoid activity function is used.
- The back-propagation method is called with the following input parameters:
    - Learning rate.
    - Input vector (a Matrix object).
    - Target vector (a Matrix object).

  The method adjusts the network according to the back-propagation algorithm (see Appendix H). The return value is the mean squared error between the Target vector and the output vector (calculated by making a feed forward with the input vector) before the adjustments of the weights.

Bayesian network

- The implementation of the Bayesian network classes contains the possibility of creating Bayesian network given the cliques of the junction tree (see also Appendix I). This means that the user of the Java classes must know the junction tree before he or she can create and use the Bayesian network. The reasons for omitting the construction of the junction tree (e.g. performing the triangulation) are:
    - The task of constructing the junction tree is difficult to implement.
    - The structure of the junction tree will in this application be unchanged during the test.
    - The result of the junction tree can easily be verified by the use of existing tools (in this project Hugin Lite 5.4 was used – the latest version of the program can be downloaded from [HUGIN]).
- The cliques can contain 2 or 3 variables. Each clique is constructed with input parameters defining the variable names and the corresponding quantitative representation of the variables (conditional and a priori probabilities).

- The separation sets are constructed by two input parameters, which are the two clique objects that are connected. The variables present in the separation set is automatically detected and initialised.
- The implemented Bayesian network can only be created with Boolean random variables. It is not possible to create network with e.g. conditional Gaussian variables.
- After the creation of the cliques and the separation sets it is possible to add evidence to the network and to extract the needed probabilities:
    - A method (setVar) is used to set the given evidence for a variable for both cliques and separation sets. The input parameters are the variable name and the evidence for the variable (0-1).
    - Another method (getProb) is used to get the probabilities for a variable. The necessary input parameters are the variable name and the corresponding clique. It should be noticed that the propagation of evidence in the Bayesian network would ensure the probability would be the same no matter what clique is used (as long as the variable is in the clique).

27.3 Test

In this section the developed method for learning and prediction is tested. The test will as already mentioned only be performed on the category of films, but the project group has reason to believe that the method can be applied to any category as long as there is some detailed description of the TV-programs available. This will although not be tested in this project.

Reference Films

The test is performed on a total of 110 films, which are collected from a video distributor (www.blockbuster.dk). The test is based on the assumption that the description of the films taken from the distributor somewhat corresponds to the description of the films given by the TV-data provider. The reason for not taking 110 film descriptions from an Internet information channel is mainly that it is necessary to get some test persons to evaluate the films according to the scale given in the system (“Want to see”, “Maybe want to see” and “Do not want to see”) and it is more likely that the participants knows the films taken from the video distributor rather than films taken from a weeks TV-programs. This should therefore ensure a more reliable evaluation. The films are before the test evaluated by 4 persons after the already mentioned scale.

The 110 films are used in the following manner:

- 1-40: The first 40 films are used to create the reference film vector. The project group extracts the keywords manually.

- 41-80: The second 40 films are used as learning films. The learning films and the test persons evaluations are used in the training of the Neural network and as the quantitative representation of the Bayesian network.
- 81-110: The last 30 films are used to evaluate the prediction of the Neural and Bayesian network compared to the test person's evaluations.

The 40 learning films are considered appropriate. This number is considered to be enough to give an indication of the success of the chosen methods. The 40 learning films corresponds to a user of the TV-guide has evaluated 40 films, which must be estimated to take several days and it can therefore be discussed whether a system that needs more learning films is useful at all. The 30 prediction films are considered to be enough to minimize the uncertainties of selecting random films. If e.g. only 5 prediction films were used there would be a significant risk of errors due to the low number of films.

Film Categorisation

The categorisation of the learning and prediction films is performed by the use of the reference film vector. It should be noticed that it has been chosen to transform the categorisation vector into a Boolean vector, where all entries containing hits (>0) are set to 1. This transformation is performed because the Bayesian network only can handle Boolean quantitative data. Additional tests on the Neural network have although been made in order to evaluate the effect of using other than a Boolean vector.

As already indicated there are tested two methods in the keyword matching. The first method is simply a direct match where the word found in the film description must exactly match the keywords in the reference vector in order to be considered as a hit. The other method proposed is inspired by word stemming, but is a much simpler method that is called heuristic word stemming. The words from the film to be categorised are considered to be a keyword if there are keywords in the reference vector that matches the first four characters of the word to be tested (words of less than four characters still requires an exact match). This method will naturally not work perfectly because a word like "test" will result in a match if e.g. "testament" is a keyword in the reference vector.

Configuration

The user evaluations are transformed into the following target values:

- Want to see: 0.9
- Maybe want to see: 0.5
- Do not want to see: 0.1

The reason for not using 1.0 and 0.0 is that these values are asymptotes to the sigmoid activity function used in the Neural network (see Appendix H – Figure 9). This means that the activation of the output nodes would have to reach infinity for the solution to be found, which is impossible [HA, p.83].

Neural network

The Neural network has in the test been made by 40 inputs corresponding to the 40 entries given by the reference vector. There has been used a multi layer network with 1 hidden layer, with 20 hidden nodes and 1 output node. The input learning vectors corresponds to the entries of the categorisation matrices obtained from the learning films and the target values are determined by transforming the user evaluations (0.9, 0.5 and 0.1). The stop criterion is a summarised mean square error of 0.01 based on all the input learning vectors (see also Appendix H).

Bayesian network

The Bayesian network has been constructed as three separate junction trees. A junction tree is made for each evaluation value ("Want to see", "Maybe want to see" and "Do not want to see"). There graphical (qualitative) representation for the Bayesian network "Want to see" is presented in Figure 58. There is a corresponding graphical representation for each of the other evaluation values.

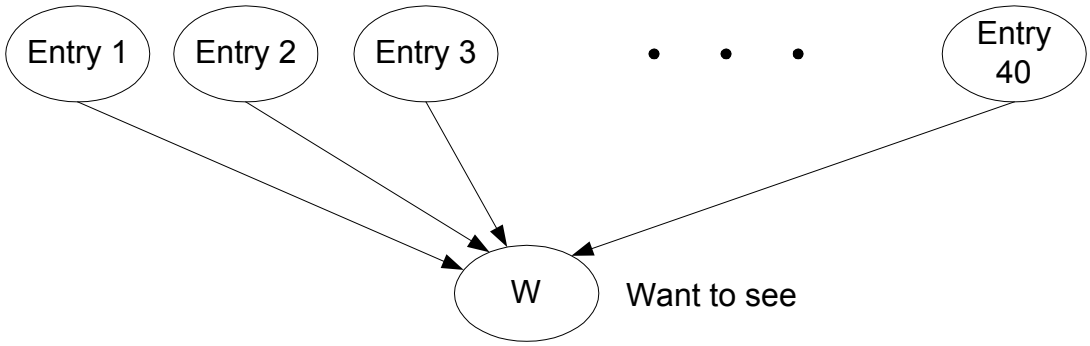

Figure 58 : The graphical representation of the "Want to see" Bayesian network. The entry nodes correspond to the categorisation vector from a film.

The quantitative data used in the creation of the Bayesian network is formed by the entries of the categorisation vector and the user evaluations. The user evaluation determines to which junction tree the data is added.

Output values

The prediction output values from the Neural network will be in the interval between 0 and 1. The following list is used in the transformation of the values into the corresponding evaluation prediction:

Output value interval	Transformed evaluation
0.00 – 0.33	"Do not want to see"
0.34 – 0.66	"Maybe want to see"
0.67 – 1.00	"Want to see"

The prediction output values from the junction tree are determined by calculating the weighted probability between the three output values:

$$P(R) = \frac{0.9 \cdot P(Want) + 0.5 \cdot P(Maybe) + 0.1 \cdot P(Dont)}{P(Want) + P(Maybe) + P(Dont)}$$

The resulting probability, *P(R)*, is transformed like the output values of the Neural network.

Test Result

The test is performed by adding one learning film at a time to the collection of learning vectors to be used in the construction of the Neural and Bayesian network. This means that the prediction is evaluated as there is gradually added more learning vectors. The error measure is summed error that describes the number of misses in the 30 prediction films. A miss is defined to occur when the output value is outside the given interval compared to a user evaluation (e.g. if the predicted output value is 0.4 and the user evaluation is "Do not want to see").

The final result of the tests is shown in Figure 59 and Figure 60. The first figure is with the exact match while the second is the heuristic word stemming method. All 30 prediction films are used.

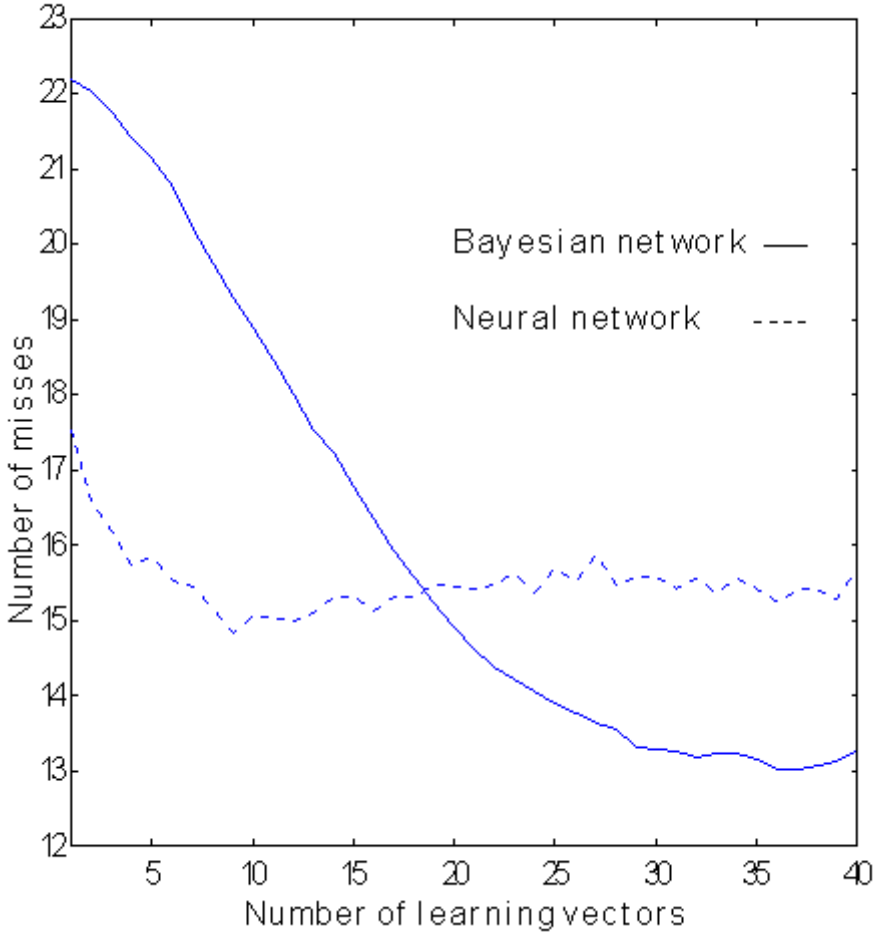

Figure 59 : Number of misses as a function of number of learning vectors. The categorisation method is based on an exact match. The solid line is the result of using the Bayesian network, while the dashed line is the result of the Neural network.

It should be noticed that the learning vectors are added one by one so that the effect of adding learning vectors gradually. A random generator will get an average of 20 misses corresponding to a hit rate of 33% (the number of hits in proportion to the number of prediction films).

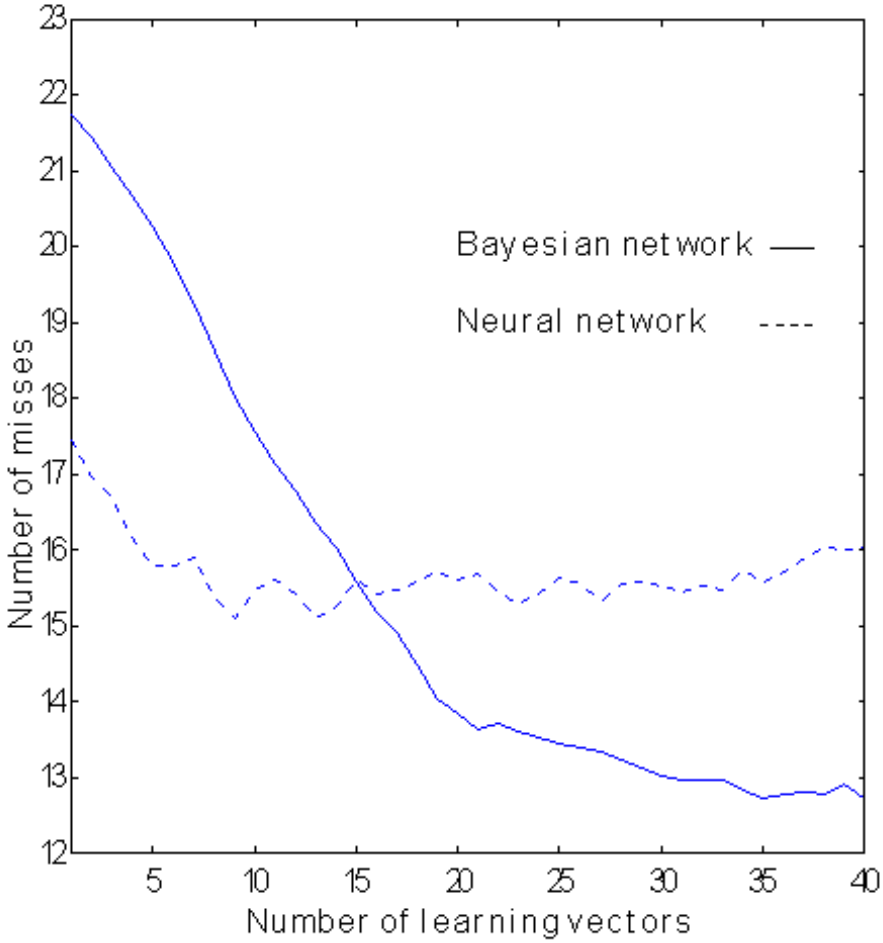

Figure 60 : Number of misses as a function of number of learning vectors. The categorisation method is based on heuristic word stemming. The solid line is the result of using the Bayesian network, while the dashed line is the result of the Neural network.

The figures show some immediate results:

- The Bayesian network starts out whit a significant higher number of misses, but after 40 learning vectors the situation is opposite.
- The Neural network performs better than the Bayesian network until more than 15-18 learning vectors are used.
- The number of misses in the Neural network seems to reach a plateau after about 10 learning vectors.
- The Bayesian network on the other hand seems to reach a plateau near 40 learning vectors.

There is a strong indication of that Bayesian performs best when using 40 learning vectors. The final values when using 40 learning vectors are:

Type	Number of misses	Hit rate
Neural network with exact match	15.68	48%
Bayesian network with exact match	13.25	56%

Neural network with heuristic stemming	16.03	47%
Bayesian network with heuristic stemming	12.75	58%

The number of major misses has also been calculated. A miss is considered to be major if the test person has evaluated a film as "Want to see" and the predicted value is categorised as "Do not want to see". This is a major miss because this film will be placed in the hidden programs in the program list and this is critical due to the fact that the user would like to see the film. The reversed situation where the user has rated a film as "Do not want to see" and the predicted value corresponds to "Want to see" is not critical because the user just can ignore the suggestion. The major misses are shown in the following table:

Type	**Number of major misses**	**Hit rate**
Neural network with exact match	4.84	84%
Bayesian network with exact match	3.25	89%
Neural network with heuristic stemming	7.2	76%
Bayesian network with heuristic stemming	2.5	92%

Evaluation

The result of the test does clearly not reach the performance goal of a hit rate of 90%. The best result is obtained using a Bayesian network with heuristic word stemming (58%). This is not satisfying, but further investigation of the method must be performed in order to evaluate whether it is a usable method or not in the given task.

The Bayesian network clearly performs better than the Neural network when using 40 learning films. There is an weak indication of that the heuristic word stemming is more appropriate than the exact match when using Bayesian networks, while the opposite seems true when using Neural networks.

The Neural network performs better than the Bayesian network until more than 15-18 learning vectors are used. This could indicate that the network selection in the final design should be based on whether it is important to achieve a faster learning in the beginning of the use of the TV-guide (Neural network) or a slower but more precise learning (Bayesian network).

It should be noticed that the computational resources used in the training of the Neural network is significantly more demanding compared to the Bayesian network where the quantitative data can be immediately calculated. In the Neural network the weights are adjusted in a time consuming training process.

It should be noticed that the Neural network parameters have been tested in different configurations, but the result is similar. This means that the result does not change

significantly by adding a layer or more hidden nodes or both, which indicates that the problem is not the decision boundaries (see also Appendix H). There has also been performed some tests on changing the input vectors to the Neural network: e.g. letting the input values to the Neural network be a normalised categorised vector instead of the Boolean values without any success.

In order to investigate the method further there is a list of ideas presented below, which would be interesting to examine in future work:

- A dynamic reference vector could be implemented. This could be done by letting the system find and add keywords from new films into the best matching entry in the reference vector.
- The above idea leads to the problem of a reference vector that just increases in size. Therefore would a method that moves and removes keywords in the reference vector would be interesting to investigate further.
- Increasing the size of the reference vector is also an idea to be tested further, which although makes the computations more demanding.
- A combination of the matching methods should be tested: If there is no exact match then try the heuristic word stemming.
- The Bayesian network could be improved to handle other than Boolean input values, which means that the dynamics of the categorisation vector could be fully utilized.

27.4 Conclusion

The tests have showed that the method did not succeed in the achieving the performance goal of a hit percentage of 90. The best method tested did only reach 58%. This indicates that the method needs are serious improvement in order to fulfil the performance goal or to use a completely different method in the design.

The project group has although showed a first attempt in developing a new and simple method for categorising and personalising text. And the result could therefore be seen as the initial work of the method, where all the possibilities are not completely investigated. This could be done in the future work of the project group or by a third party. Some of the ideas of the possibilities are presented in the Evaluation section above.

The method has although been considered to be useful in the TV-guide, because the hit rate is 92% if the error measure is major misses. And the hit should also be better when using the group and keyword priorities.

28. Mindpass Integration

Because this project has been developed in cooperation with Mindpass A/S there will now be a description of the project results that might be useful for Mindpass A/S.

The different features that might have the interest of Mindpass A/S is the:

- TV-guide
- Agent framework
- Categorisation
- Ripper
- User Centred Design method

28.1 TV-guide

As described in the follow-on systems, Mindpass A/S has a toolbox with services for use on a portal (see section 8.1 Follow-on Systems). The services placed in this toolbox all have a separate data layer and layout layer. This separation of the data layer and layout layer has also been implemented in the inTelly TV-guide. The similar architecture of the two systems makes it quite easy to make the TV-guide a part of the Mindpass A/S toolbox. Another advantage in the inTelly system that makes the integration with the existing Mindpass A/S technology possible is the use of XML. The use of XML and SQL makes it easy to create an interface to new data sources other than the MySQL database used right now. The necessary changes for e.g. getting data from a new database will be the change of the driver and maybe minor changes to the SQL-queries. These changes also go for the Servlets that have database access.

Another thing besides the separation of data and layout layer that makes the inTelly system easy to integrate is the use of Java. The use of Java makes the source code independent of the operating system as long as the operating system has the possibility of running a virtual java machine for execution of code.

28.2 Agent Framework

During the project there has been developed an Agent framework that can be used in many different situations. The Agent framework is developed in Java, which gives it a great flexibility. In the framework it is possible to keep an eye on the communication as well as the condition of the different agents furthermore the framework has the possibility of creating, moving or killing the agents. The programming of a new agent is also made simple and non-demanding so that it will ease the future use of the Agent framework. This Agent framework gives Mindpass A/S a good starting point for experimenting with this technology and it also gives them the possibility to create very flexible and distributable code.

28.3 Categorisation

The categorisation methods developed by the project group might also be an interesting part for Mindpass A/S. The methods are simple but non-standard methods, which have been developed by the project group. In connection with this categorisation there has been developed some Java classes that contains:

- Learning and feed-forward of multi-layer neural networks.
- Construction and propagation of knowledge in Bayesian networks.

28.4 Ripper

There has been designed a piece of ripper software that retrieves the information of a given Internet information channel. This ripper is used both by the project group and Mindpass A/S to retrieve data from different Internet information channels with.

28.5 User-Centred Design Method

During this project there has been used the User-Centred Design method, which puts focus on the user of the system to be designed. There has been performed several iterations in the design phase and there has also been potential users involved in testing the different prototypes. The main benefit for Mindpass A/S is the knowledge that the project group acquired concerning user testing and user involvement in the development of a system. The description of the method used in the project can be used directly by Mindpass A/S.

28.6 Conclusion

The outcome of this collaboration with Mindpass A/S has been rewarding for both the project group and Mindpass A/S. The project group were able to draw on the expertise of the employees at Mindpass A/S. The result for Mindpass A/S was first of all the software and knowledge developed in the project, that is the Agent framework, a Ripper, the actual TV-guide and also the method developed in the project can be used as a project development method. Another advantage of this project is that it makes a closer connection between Mindpass A/S and the university.

29. Validation test

The objective of the validation test is to perform a test and evaluation of the usability goals presented in 14.1 Usability Goals. In the following sections the test plan, test result and conclusion are presented. It should be noticed that the complete test plan and test result can be found in the Test report.

29.1 Test

The different aspects of the usability goals are presented in the 14.1 Usability Goals and are repeated below:

- Usefulness
- Effectiveness
- Learning
- Errors
- Acceptability
- Enjoyableness

The evaluation of the usability goals are performed by letting test participants perform some specific tasks on version 1.0 of the system and fill out an evaluation scheme after having tried the system. The participants are found by sending an email to all the advanced users that stated that they would like to be test participants in the questionnaire (see Test report – Test result - Questionnaire). The test was made with 6 test participants that mailed back that they were willing to do the test.

The test is made by letting the participants solve the given tasks on their own. This means that the test monitor only will help the user if there are fatal errors like system failures. The participants are asked to think aloud during the test. This will include both what they are doing and what they feel about using the system during the test. The tasks performed by the participants are recorded by a screen-capture software (see Camtasia on the CD-ROM), where also the sound is recorded. Furthermore there is a camcorder that records the expression of the participants during the test (located next to the computer monitor recording the face of the participant).

The evaluation scheme is made by letting the participants evaluate some statements concerning the use of the system by the use of a Likert-scale.

Both the list of tasks and the evaluation scheme can be seen in the Test report – Test plan – Validation test.

29.2 Evaluation

There are made some video clips with some of the primary observations made in the test, these videos are located on the CD-ROM in the directory Validation test. An extract of the test result can be seen in the following list:

- Most of the users catch the primary concept of the new features presented in the system. See Videos 1,2,3,4,7 and 8.
- Most of the participants did not know what to do after they have created a user profile. But the other navigation in the user profile works very well. See Videos 9 and 10.
- The principle in the Repeater list is somewhat difficult to completely understand. (The different principle of prioritising programs in this list is not immediately understandable). See Videos 5 and 6.
- The speed of the system was generally considered to be slow.
- The facility of prioritising the TV-programs was in several situations immediately understandable without any introduction (neither the general help nor the program list introduction). See Videos 2,3 and 4.
- Only 67% of the participants were able to find out what to see in TV for one day in less than 3 minutes. The usability goal was a percentage of at least 70.
- All statements in the evaluation scheme have passed the test (83% or above have answered “neutral”, “agree” or “strongly agree” to all statements).

The usability of the new facilities in the TV-guide is considered to be of major importance. The participants were all able to use these facilities after trying out all the tasks. It should be noticed that there is still some doubt about the possibilities and the works of the features at the beginning of the use. One of the test participants used the introduction to the TV-guide and was immediately familiar with the principle in the integrated notepad and the possibility of prioritising the programs. This could indicate (and only indicate!) that a solution to the problem could be to make the introduction more visible to new users. In version 1.0 the introduction is step three in the create-user-wizard, but can also be found in the system help. In the test only one person actually read the messages in the wizard and this is not satisfactory. It should be noticed that the test situation is an artificial situation and it is likely that more of the end users will read the wizard messages due to the fact that in a normal use there will not be handed out a predefined list of tasks that the user must try to perform. One reason for the missing use of the wizard and messages in general could therefore be that the test participants primary concentrate on solving the tasks instead of reading the wizard properly.

There was discovered one major usability problem observed when the user has created the user profile and was in doubt about what to do next. This would be an important thing to correct (e.g. adding a button/link to the program list) before a release of the product, both

because 5 out of 6 users had problems in this situation and because the users were seriously in doubt about what to do.

The principle in the Repeater list was somewhat difficult to understand. It was not clear for several of the participants that is possible to prioritise repeating TV-programs in this list and that it will effect the program list. This should be stated clearly in a release version of the system. One problem might be that the Repeater list and the program list are almost identical and the user might therefore not be aware of the differences. A solution could be to make them distinct and in that way indicate to the user that the two lists do not work in the exactly same manner. This solution could although disturb the consistency in the Website. Another solution might be to show an additional explanation and/or illustration when the user enters the Repeater list for the first time. But too much text has showed to annoy the user (see Test report – Test result – Assessment test).

The speed of the system was generally considered to be slow. One participant suggested that the system should run on a computer with a faster Internet connection. This will not solve the problem because the primary reason for the lack of speed is the speed of the computer. The task of sorting e.g. the program list is a demanding task and a solution could therefore be to use a more powerful computer (than the PII – 266MHz used in the test). It should also be noticed that one main reason for the relatively slow system speed in the test is that the screen capture software (see Camtasia on the CD-ROM) uses a significant part of the system resources. This will naturally not be a problem in a normal use of the system.

The principle of prioritising the programs was immediately understood by several of the participants. This indicates that the problems concerning this feature discovered in the earlier versions of the system seems to be solved.

Only 67% of the users did find out what to see in TV for one day in less than 3 minutes. This means that the corresponding usability goal was not reached. This could be considered as a major usability problem, but the two participants that used over 3 minutes was relatively close to the demand (3:45 and 3:27). Therefore it is not considered to be a usability problem. It should also be noticed that this duration would be minimized when the TV-guide personalises the TV-programs for the user when he or she has prioritised some TV-programs in the programs and Repeater list.

The result of the evaluation scheme indicates that the system is successful concerning the acceptability and enjoyableness. A more thoroughly presentation can be found in the Test report – Test result – Validation test.

29.3 Conclusion

The inTelly system version 1.0 fulfils the usability goals except the time constraint of finding out what to see in TV. The Validation test has showed to be useful although the cam recording of the participant's expressions during the test was less useful. This means that the screen and sound recording is considered to be enough in a future use of this test method.

The result of the Validation test will in this project be the last user test performed on the system. In a real situation the results obtained would be used to make a release version of the system. The usability problems discovered in the test are considered to be relatively easy to correct if a release version of the system should be made.

30. Conclusion

This chapter presents the functional requirements and testable goals listed in the Initial design.

Each demand to the system is evaluated by the use of the following symbols:

✓ : The demand is completely fulfilled.
(✓) : The demand is partly fulfilled.
- : The demand is not fulfilled.

There will be a short discussion when a demand has not been completely fulfilled.

Functional requirements to the TV-guide	
1. There should be filters helping the user to get an easy overview of the many TV-programs available.	✓
2. It should be possible for the user to define the amount of information displayed for each program. It is possible for the user to decide if the short description of a program should be shown or not, but it is not possible to remove or add columns in the Program list. This possibility is omitted primarily because it can give rise to usability problems when the user can control the contents of the user interface.	(✓)
3. There should be some notepad function where the user can place the programs he or she wants to see.	✓
4. It should be possible to create/update/delete/view a user account.	✓
5. There should be some guidelines/wizard that helps the user in the creating the user account.	✓
6. It should be possible for the users to add keywords. This is only possible for all the programs, and not as requested for each category.	(✓)
7. The user should be able to make a printout of the TV-program.	✓
8. It should be possible to port the TV-program to other devices (Palm, WAP). This possibility is omitted because of time constraints. But the system is prepared for using other devices, because of the use of XML and XSL.	-
9. It should be possible to record selected programs. This possibility is omitted because of time constraints and additional hardware demands.	-
10. The TV-guide should give the user the possibility of getting a reminder about selected TV-programs. It is possible to receive a mail each morning containing personal programs for the day. The possibility of using SMS, etc. is not implemented because of time constraints and it was given a low priority.	(✓)

11. It should be possible for the user to tell the system, which TV-programs he or she likes/dislikes and the system should use this information to prioritise the TV-programs in order to personalise the TV-guide.	✓
12. The system should prioritise the TV-programs based at prioritising made by users with similar interests.	✓
13. There should be a search feature making it possible to search in the TV-programs.	✓
14. It should be possible for the user to select and deselect repeating programs.	✓

Functional requirements to the Agent framework	
1. The Agent framework should support the agent characteristics given in chapter 3. Agents (e.g. mobility, flexibility etc).	✓
2. It should be possible to distribute the tasks and agents on a number of computers in order to achieve a load balance.	✓
3. It should be possible to manually remove and add agents during runtime.	✓
4. During runtime it should be possible to observe the status of the agents and the agent communication.	✓

Performance goal	
1. The system priority must match the user priority with an accuracy of 90 percent. This test is difficult to perform because the system have to be used for a period of time by several users, and then there has to be made a test where the users prioritise the different programs. The test that has been made is made on the categorisation priority developed, which showed that it was able to get a hit rate of 58 percent. But the hit rate with only major misses was 92 percent. See section 27.3 Test for details on the test.	-
2. The TV-guide should be able to detect 90 percent of a week's repeating programs.	✓
3. The load time for one page in the inTelly Website should be maximum 10 seconds at a modem speed at 56kbit/s (one user). The load time is approximately 20-40 seconds on a 56kbit/s. This can be improved by optimising the data send to the browser.	-

It should be noticed that the usability goals have been tested and evaluated in the Validation test (chapter 29).

CONCLUSION

31. Reflection

This section presents some reflections on the primary issues in this project. This means the User-Centred Design Method, the Agent framework, the Personalisation of the TV-programs and the overall end product: the TV-guide.

31.1 User-Centred Design Method

The experiences of using the method through all phases in the development are presented below. The comments are based on the four design principles in the User-Centred Design method.

The experience gathered by making a project like this by the use of this method is very valuable. This experience can be used in future projects, e.g. the Test report provide a lot of arguments for designing user interfaces in a specific way because of the tests made. When the method has been used for several projects a library is gathered of Test reports that provide arguments for designing the user interface in different ways.

Early and Continual Focus on the Users

The users have been put into focus in all stages of the development of the TV-guide. In the Pre-analysis part there was focus on the users without directly user involvement. The Pre-analysis part was used to define the system and the end users, who in this project is advanced Internet users. Several user involvements were achieved through out the rest of the development by performing various user tests (described below). The early and continual focus on the users has been considered valuable. It was possible to adjust the system to the user's needs at the different stages in the development and the users came up with valuable ideas and suggestions. The system is based upon the user requirements and to get the users job done.

Early and Continual Focus on User Testing

There were used several user tests during the development of the TV-guide. The user tests have covered most of the usability tests presented in [JR]. Furthermore Jakob Nielsen's heuristic usability evaluation was used in two tests. A total of five user tests were performed during the development and one validation test performed on the final TV-guide. The task of performing user tests is rather demanding and the size of the Test report illustrates the work put into the user tests. It should therefore been considered whether this additional work has been worth the effort. The project group is not in doubt that the TV-guide has reached a higher level of usability than would be achieved without user testing. In the Initial design phase the user tests supported the development of the functional requirements and testable goals and the design of the preliminary user interface by ideas and suggestions (both

explicitly and implicitly). In the Iterative development part the user tests were used to evaluate the versions of the TV-guide and the results formed the basis for the succeeding versions. A Validation test was used to verify the usability goals listed in the chapter 14.1 Usability Goals. The tests and evaluations used in the different stages of the design could be a subject of discussion. The project group determined when and which tests to be used at an early stage of the development. It would therefore be appropriate to reflect upon the choices made.

The questionnaire and the observation test performed in the Initial design phase were as already mentioned helpful in the development of the functional requirements, testable goals and the first version of the system. The questionnaire did mostly contribute with implicit suggestions (e.g. the participants did mention their favourite Internet news providers and TV-guides) and it could be argued whether this method was effectively. Furthermore the time taken to construct the questionnaire was significant in order to make it acceptable and easy understandable. The time spent and efforts put into making the questionnaire has although been considered acceptable. If the different user tests are compared, it is the questionnaire that contributes least to the development compared to the effort. Apart from the suggestions obtained from the questionnaire result the project group also received a number of email addresses from interested participants. These email addresses were used in the gathering of participants to the Validation test saving valuable time.

In the Iterative development phase there were used two heuristic evaluations (version 0.1 and 0.3) and one assessment test (on version 0.2). The chosen tests and evaluations are considered appropriate. It could although be questioned whether the first heuristic evaluation (performed by the project group) is the best choice at this early stage of the development. The result of the heuristic evaluation is not normally used to make a choice between two or more user interface designs. A more formal comparison test at this stage could be considered in future work.

The purpose of the Validation test performed on the final version of the system was to test and evaluate the usability goals and to get some demonstration videos. The test proved to be successful in this task.

Iterative Design

The principle of developing prototypes, testing and evaluating them is considered very useful. There is considerable difference from the first version of the system (version 0.1) to the last iteration of the system (version 1.0). As the corresponding section in the Main report (Iterative development) illustrates this principle have been used ending with a final version of the system (1.0). By applying tests and evaluations to each prototype the project group has received a number of arguments to be used in the succeeding prototype. These well-grounded

arguments could not be achieved elsewhere and therefore must the value of this principle not be underestimated.

Integrated Design

The Integrated Design principle consists of applying all usability aspects early in the development (this includes user manuals, help facilities, system installation etc). The situation in this project and any other university projects is somewhat artificial. This means that e.g. the system installation has not been a part of the project. Nevertheless the project group has in co-operation with Mindpass A/S evaluated the possibilities of implementing the TV-guide as a service in their list of services. These considerations have naturally involved the system installations aspects in all stages of the development. The TV-guide is based on Internet and a user manual has there for not been considered appropriate and a user manual to the Agent framework (e.g. to the system administrator) has not been made due to the fact that this was out of the scope of this project. The help facilities in the system has although fully been considered from the beginning of the development. The resulting help facility is considered to be working well and the test participants that used the help facility in the Validation test could easily find what they were looking for. It should although be mentioned that the help facility can naturally be improved by e.g. adding a context sensitive help.

31.2 Agent Framework

It was decided to find out whether agents are suitable in solving the tasks in an Internet based TV-guide. An Agent framework has been developed with a number of facilities (distribution of agents on several computers, administration of agents, move agents between agent servers, etc). The first implemented version of the system did use the agents to all tasks in the TV-guide, but it was found inappropriate in some situations. The result was that e.g. the Servlets got direct access to the database, because this significantly speeds up the process of e.g. getting TV-data to the browser. This illustrates one of the weaknesses in this Agent framework: The needed communication between the agents will in some situations slow down specific tasks compared to more conventional programming techniques. This means that the agents in this project only are used in tasks that can be performed without demanding time constraints. An example could be the task of updating the user groups, which only should performed once a day and therefore can be updated at night. The use of agents has throughout the development had several advantages. It is relatively easy to add new functionalities to the system by adding a new agent. Furthermore it has proved to be easy to correct errors and make adjustments in the development of the system, because agents can easily be removed and added from/to the system.

This project delivers the basic functionality of agents by the Agent framework, which could be exploited much further than done in this project.

31.3 Personalisation of Information

The personalisation of the TV-guide has primary been concerned about developing a new method to personalise films. The project group will although claim that the method easily is adapted to other TV-categories as long as they have a textural description. The method did not live up to the performance goals (58% compared to the goal of 90%). But it is considered possible to increase the performance of the method by making further changes and adjustments. It should also be noticed that the use of keywords and group priorities is considered to increase the hit rate of the system priority. This has although not been tested due to the fact that this test is time demanding (a number of users must use the system for a longer period in order to evaluate the effect of these personalisation methods).

31.4 The Final TV-guide

The final system is an Internet TV-guide that is considered to be close to the release version if it was to be put out on the Internet permanently. The use of Java and database makes the system flexible both concerning where to run and the TV-data available.

32. Conclusion

This project has focussed on the following issues:

- User-Centred Design
- Agents
- Personalisation of information

The reflection on the issues has been presented in chapter 31. Reflection.

The User-Centred Design has proved to be a development method that is very useful when designing systems like the TV-guide. The task of performing the user tests is although significant, which should be taken into account when the project schedule is made. The project has proved to succeed in making this schedule realistic and it was not necessary to make significant adjustments to the overall goals of the design due to time constraints. The user tests proved to be valuable in the development of the prototypes during the system development and they are all considered to worth the extra work that is necessary in order to make a useful test.

The use of agents was also a major issue in this project. Some of the pros and cons in using agents have been explained. The agents implemented in this project are not well suited to perform tasks that have narrow time demands, but the use of agents has resulted in a design that is flexible and the functionality is easy to change according to future tasks. There is furthermore a possibility of distributing the task on several computers in order to make a load balance, and to make agents that could control this load balancing.

The method developed to personalise the TV-programs did not succeed in fulfilling the performance goal. But it is considered to the initial work of a method that might show to be appropriate in simple personalisation applications if it was investigated further.

The final version of the TV-guide is considered to be a system with a high level of usability obtained primary by the use of User-Centred Design. The development of the Agent framework and the personalisation of the TV-programs have given some suggestions to how these issues could be implemented and used in practise although specially the personalisation needs further investigations.

BIBLIOGRAPHY

[AC]	Agent sourcebook Alper Caglayan, Colin Harrison ISBN: 0-471-15327-3 John Wiley & Sons, 1997
[ÁRL]	Brugermodeller og brugerprofiler i intelligente søgesystemer Árni Rúnar Loftsson Danmarks biblioteksskole, Aalborg afdelingen, 8 semester, 1999
[CL]	Making usable, useful, productivity enhancing computer applications John D. Gould, Stephen J. Boies and Clayton Lewis Communication of the ACM, January 1991, Vol. 34, No.1
[CMB]	Neural networks for pattern recognition Christopher M. Bishop ISBN: 0198538499 Clarendon Press 1995
[DIC]	Online web dictionary http://www.dictionary.com/
[DM]	KQML – A Language and Protocol for Knowledge and Information Exchange Tim Finin, Richard Fritzson, Don McKay and Robin McEntire
[EB]	Brugerorienteret interaktionsdesign II Egil Boisen http://www.stud.hum.ku.dk/boisen/mm2000/mm09.html
[FVJ]	An introduction to Bayesian networks Finn V. Jensen ISBN: 1-85728-332-5 Aalborg University, 1996
[GAC]	Marketing Research, sixth edition Gilbert A. Churchill, Jr. ISBN: 0-03-098366-5 The Dryden Press, 1995
[GFL]	Artificial Intelligence, Third Edition George F. Luger & William A. Stubblefield. ISBN 0-805-31196-3. Addison Wesley Longman, Inc. 1998.
[HA]	Neural networks Hervé Abdi, Dominique Valentin and Betty Edelman ISBN: 0761914404 Sage Publications, 1999

[HSN]	Software Agents: An overview Hyacint S. Nwana Knowledge Engineering Review, Vol 11, No 3, pp. 205-244, 1996
[HUGIN]	Hugin Lite 5.4, Help The latest version of Hugin Lite is available at http://www.hugin.com/
[IBM]	User Centered Design - IBM IBM http://www-3.ibm.com/ibm/easy/eou_ext.nsf/publish/570printview IBM homepage
[JB]	Constructing intelligent Agents with Java Joseph P. Bigus and Jennifer Bigus ISBN: 0-471-19135-3 John Wiley & Sons, 1998
[JDG]	How to design usable systems John D. Gould IBM Research Center, Hawthhorne, Yorktown Heights, New York 10598, 1988
[JDG2]	Designing for usability: Key principles and what designers think. John D. Gould & Clayton Lewis Communications of the ACM 28, 300-311, 1985
[JF]	Multi-Agent Systems Jacques Ferber 0-201-36048-9 Addison-Wesley, 1999
[JN1]	Usability Engineering Jakob Nielsen ISBN: 0-12-518406-9 Academic Press, 1993
[JN2]	Designing Web Usability Jakob Nielsen ISBN: 1-56205-810-X Academic Press, 2000
[JN3]	Useit.com Jakob Nielsen http://www.useit.com/ http://www.useit.com/papers/heuristic/severityrating.html Jakob Nielsen
[JR]	Handbook of usability testing Jeffrey Rubin

	ISBN: 0-471-59403-2 John Wiley & Sons, 1994
[KL]	Spørgesksemaundersøgelse – Gør det selv på PC, 2. udgave Michael Feder, Tine Vedel Kruse og Claus Nielsen. ISBN: 87-7848-479-0 Udgivet af kommunernes Landsforening Kommuneinformation, december 1999 Jydsk Centraltrykkeri
[LBL]	Introduction to Usability Engineering – PE Course/slides Lars Bo Larsen Aalborg University, 2000
[LU]	http://www.cognetics.com/lucid/lucid.html Cognetics, 1999
[MSQL]	MySQL Database server http://www.mysql.com/
[NNO]	Neural Networks Overview http://members.nbci.com/_XMCM/omarvision/tutorials/neuralnet/overview.htm
[NRJ]	On agent-based software engineering, Artificial Intelligence, Volume 117, number 2 Nicholas R. Jennings 0004-3702 Elsevier, Amsterdam, March 2000
[NS]	Notes on Intelligent Software Agents Nahid Shahmehri and Patrick Lambrix Linköping University, 1998 http://www.ida.liu.se/labs/iislab/courses/Agents/paper/chapter2.html
[OL]	JavaScript library http://www.bosrup.com/web/overlib/
[RF]	KQML as an Agent Communication Language Tim Finin, Richard Fritzson, Don McKay and Robin McEntire ACM Press, November 1994
[SA]	Medical Decision Support Systems Based on Casual Probabilistic Networks Steen Andreassen Aalborg University, 1999
[SAM]	The Software Agents Mailing List FAQ http://dent.ii.fmph.uniba.sk/ui/faqs/agent_faq.html
[SF]	Is it an Agent, or just a program? A taxonomy for Autonomous Agents

	Stan Franklin and Art Graesser University of Memphis, 1996
[SM]	ZIP stat statistics Simon Mikkelsen http://www.zipstat.dk/ 1998-2000
[SUN1]	Java Servlets, download http://java.sun.com/products/servlet/download.html
[SUN2]	Java Servlets http://java.sun.com/products/servlet/index.html
[TF]	DRAFT Specification of the KQML Agent-Communication Language Tim Finin and Jan Weber June 1993
[TV]	Internet TV-guide http://www.tv-guide.dk
[W3]	Web Building Tutorials http://www.w3schools.com/xsl/default.asp Nextra AS
[YL]	Agent Communication Languages: The Current Landscape Yannis Labrou, Tim Finin and Yun Peng IEEE March/April 1999
[XML]	Microsoft XML parser for Internet explorer http://msdn.microsoft.com/xml/general/xmlparser.asp