Appendices & Supplements

Contents

APPENDICES

A. Usability Principles

In this appendix it is tried to identify the main things, which should be known by designers when they try to design good and user friendly systems. This type of system design process, referred to as usability design, consist of four key points also called principles. This design process puts a focus on the users and their interaction with the system, which is done by using the four principles made by [JDG] and [JDG2]. Some authors have also categorised these design principles into a number of three but in this project it was chosen to use the categorising of four principles. The four principles are:

- Early – continual – focus on users
- Early – and continual – user testing
- Iterative design
- Integrated design

In the following sections these principles will be explained in detail. There will also be a number of suggestions for methods used to achieve the principles.

Early – Continual – Focus on Users

When making a User-Centred Design it is important to have direct contact with the intended or actual users. Therefore the first to be done is to identify the actual user and also what they will be doing with the system. This means that the designer must actual meet with the users (face to face) and not only e.g. read about their behaviour etc. It is important for the designer to understand the users in order to make a User-Centred Design. There are several methods to get focus on users and in [JDG, p. 98-103] there is presented the following list:

Talk with users	Talk with the intended users and discuss their problems and what works well in the current situation.
Visit customer locations	Visit potential locations for the system to be used. The designers will typically find out things they could not imagine, when seeing the actual place where the system should be used.
Observe users working	Visit the workplaces of the users. The typical situation is that a system is a follow-on to existing systems. This means that it is important for the designer to get in contact with the existing system, and the best method to do this is to see it in use and thereby learn its good and bad features.
Videotape users working	It will be a great help if the other members of the design team can see the users working. Making a video recording could do this.
Learn about the work	Here the designer should be aware of what organisation

organisation	structure the system should work in. This means that it will be possible to take in account if there e.g. are different types of users using the system.
Thinking aloud	If the potential users are saying out loud what they are doing, and what they want, this gives the designer a good insight in the users work. Often this method will ensure that important details will be visible to the designer.
Try it yourself	The method will ensure that the designer will be able to see many of the user tasks in a different light. The designer will have to take a direct confrontation with the existing system and the corresponding tasks.
Participative design	This means that the intended users should be a part of the design team. If this is possible it will ensure that the potential users continuously inspect the design. This will minimize the efforts in getting feedback from the users.
Expert on design team	This is another method of getting relevant members into the design team. The expert (e.g. the potential user or another expert) will be connected to the design team as a consultant.
Task analysis	This is an analytical process used to determine the specific behaviours of the users of the system. Making scenarios sketching the working process. Breaking job activities into task units. The result is a list of functions needed in the system.
Surveys and questionnaires	To be able to make good questionnaires the designer must first talk to the users in order to get an idea of what to ask. The answers can be a good help in the design process.
Testable behaviour target goals	A testable behaviour target goal is a way of giving phrases like “user friendly” a more technical basis and to give an metric indication of what progress has been made with the new system. A testable behaviour target goal could be: How many of the users can do this operation in 5 minutes? And the passing mark could be e.g. 70%.

Early – and Continual – User Testing

The hypothesis is that it is not possible to make an optimal design the first time. This means that there should be some considerations during the design of the system. Continual user testing (from the design start) is a good measure of whether the system is heading in the right the direction or there should be made corrections.

The methods in early user testing are the following [JDG, p. 103-108]:

Printed or video scenarios	There should be made some user scenarios on e.g. paper and shown to members of the design team. The discussion about the user scenarios is ensuring that the design team has to discuss both the sketch of the user interface and the functionality behind.
Early user manuals	This gives the potential users a good opportunity to comment on the design because a user manual ensures that the design is easy to understand for the potential users.
Mock-ups	Making a mock-up will give both the users and the designers a good idea of the system and therefore is this a good starting point in a dialogue.
Simulations	Many system functions can be simulated (e.g. pencil and paper). The simulations are therefore a good help in the design process.
Early prototyping	By using software development toolkits, prototypes can easily be created. The prototypes can give the designers a quick and precise response from the users.
Early demonstration	This will ensure that the designer will have to go through the operations (have to explain the sequence of operations) and maybe will find out problems with the system.
Thinking aloud	This will mean that the designers will get to know what the users are thinking and where there are problems with the design of the user interface.
Make videotapes	A video can open the minds of the designer about a problem in the system.
Hallway and Storefront methodology	This method is about placing the system in a public place, where people can tryout the system. This makes it possible to get in touch with many different users.
Computer bulletin boards, forums, network and conferencing	This makes it possible to get in touch with users all over the world.
Formal prototype test	Most of the previous mentioned methods are informal and these give valuable information about the design. But where possible, it will be a good idea to make more formal test, which typically gives more accurate results.
Try-to-Destroy-it contests	At the end of the design process it will be valuable to get some users to try to make the system go down by pressing it to its limits.
Field studies	This will help discover problem with the system outside the secure world in the laboratory.

Follow-Up studies	These are studies carried out on the system after the release date.

Iterative Design

When developing a user-centred system, there has to be iterated in the design, because you cannot make it right the first time. The key elements are [JDG]:

- Identification of required changes
- An ability to make changes
- A willingness to make changes

The required changes will typically be identified in corporation with the users (user tests, thinking aloud, video recordings, etc). To apply the changes it is necessary to have both the ability and the willingness in the design team to make them happen.

In the following table some methods to achieve the iterative design are listed:

Software tools	It is necessary to have software tools that support the iterative design. This means that changes both in the user interface and in the functionality must be easily applied. Many of today's graphical software tools make changes in both the user interface and the functionality simple.
System development work organisation	The design team must be a part of an organisation that supports the User-Centred Design method. This implies that the whole organisation has a structure that can handle eternal changes and are willing to handle these changes.

Integrated Design

One of the key aspects in User-Centred Design is to ensure that all usability aspects are handled in parallel. This implies according to [JDG] that one person is responsible for all aspects concerning usability.

The methods that could be used to ensure an integrated design are:

Few persons in control of usability	As already stated only few persons should handle the usability. This ensures that changes in usability aspects of the system can be applied without the whole design team being involved.
Early focus on users	Applying early focus on the users will ensure the integration of the design because all user aspects will at an early state become clear to the designers.

B. Usability Heuristics

This appendix presents the usability heuristics made by Jacob Nielsen in the book "Usability engineering" [JN1] and from his homepage useit.com [JN3].

Usability heuristics are principles that can be used in an evaluation of almost any type of user interface [JN1, p. 115]. In this project the usability heuristics are used in the accomplishment of usability evaluations.

The primary reason for using the usability heuristics are that the use of the heuristics gives a systematic evaluation of a user interface. The heuristics are furthermore simple to use and is common knowledge in the task of usability engineering.

The usability heuristics contains the following principles [JN1, p. 20]:

- 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

In the following sections the usability heuristics are described in detail.

Simple and Natural Dialogue

A user interface should always be kept as simple as possible. The main reason is that the more information and possibilities that are available on a user interface the more the user has to learn resulting in more potential errors.

The user interface should match the users' tasks as good as possible. If the user interface is made according to the users tasks the needed navigation will be minimized and so will the errors.

Connected information should be clustered on the user interface and only what is needed for the current task should be displayed. Important information should be presented in a way that

requests focus (without overdoing it!) Too much information could result in that important information, features etc. is overlooked.

The user should be in control of the sequences in the task to be performed although the user interface should contain a suggested sequence for novice users.

If colours are used carefully they can contribute positively to the user interface. If not the colours can result in a more confusing and distracting user interface.

Speak the Users' Language

The terms used in the user interface should be kept in user terms and not in system oriented terms. If possible the language should be the native language of the users. The language does not necessarily have to be kept simple, but it should be kept in terms of the actual users' vocabulary, which means it is allowed to use technical/special language of the user.

It is important that there is a mapping between the information presented at the user interface and the users conceptual model of the tasks to be done.

One method to ensure to speak the language of the user is to e.g. let the users vote for their favourite terms in the design phase. Another solution could be the use of metaphors in the design, but there is always a risk that the metaphors could be misinterpreted.

Minimize the Users' Memory Load

The user cannot remember everything. Therefore should the system help the user whenever possible. Some examples of suggestions to minimize the users' memory load are presented below:

- When the user are to enter some data it is often easier to edit some default values rather than entering all data. This could hold for situations where the entering of the data are used often and it is typically the same data to be entered every time.
- In input fields it is often helpful to illustrate the required format, also before the errors are made.
- Use of generic commands. This means to use the same command for similar things to happen in different situation. E.g. a copy/paste command could work in different ways depending on what to copy/paste, but it would a good help for the user if the command was the same because he or she then will have one command sequence less to learn.
- The usage of icons makes the action easy to remember and recognise because of their figurative/metaphorical appearance.

Consistency

The user will be able to learn to use a system faster if there is kept a certain consistency in the design of the user interface. One example could be that a command entered always performs the same task and not something completely different in different situations.

Another way to achieve consistency is to use a standard user interface that is well known by many users. This will ensure that the users are typically already familiar with the layout and can concentrate on their tasks.

Consistency in a system is also achieved by making sure that the parts of the user interface that is to be used in more than once is fixed at the same location all the way through the different user interfaces.

Feedback

The system should always inform about what it is doing. There should also be a feedback telling how the system interprets user inputs. It is important also to give positive feedback to the user and not only feedback when something has gone wrong.

Important types of feedback are e.g. given as a warning message in situations where the user is about to perform an irreversible task.

There should also be some considerations about the consistency of the feedback. In some situations it would be appropriate to have a low consistency (e.g. minor problems that are solved automatically) and in other situations a high consistency is necessary (e.g. major software/hardware failures).

The response time is an important feedback factor. If the response time is too long and there is no feedback then the user might think that something went wrong in the process and might start a search for the cause. There are also situations where the response time could be too short and some feedback is necessary (e.g. in situation where the task is performed so quickly that the user do not believe it was performed correctly). In situations with long response time it is a good idea to use progress bars, explaining dialogues boxes etc.

Clearly Marked Exits

The user must never feel trapped in any situation, because the user will get a feeling of loosing control. There should therefore always be clearly marked exit/cancel/back/undo-options available. There will always be made errors so it is important to make it easy to

recover from errors. It should also be possible to escape tasks that take longer than e.g. 10 seconds.

Shortcuts

For the system to appeal both novice and advanced users it should make it possible to perform tasks by the use of shortcuts. Experts will benefit from such features when certain tasks are to be performed several times. Shortcuts include: "type-ahead" possibilities (do not need to wait for the system at all stages), historic that remember what the user has entered and default values in input fields etc.

Good Error Messages

Error messages are critical because the user is in an unexpected situation where the system failed to perform some decided task. The error message is an opportunity for the system to give the user a constructive help solving the problem.

The demands for error messages are:

- They should be made in clear language (users language). This means that error messages containing only some error code to be used by some administrator is abandoned. If the error message contains error codes they should be placed in the last part of the message or in some sub dialogue box.
- The error message should be precise rather than general in order to help the user with the specific situation.
- The error message should be constructive and even better it should try to guess what the user meant in the given situation and give suggestions how to solve the problem.
- They must be kept in a polite language and never blame the user.

One way to help the user to recover from errors would be to implement undo functions. A good error message should refer to supplement reading in e.g. an online manual.

Prevent Errors

Preventing errors is even better than to provide good error messages. Errors typically occur when the user e.g. has to spell out certain commands or data (typing errors). Typing errors can sometimes be avoided by the use of e.g. check boxes, selectable items etc. Errors can also be avoided by warning messages displayed before some critical task is to be performed. In general a well-structured user interface can prevent many errors, because the important information is presented clear and the state of the system appears distinctly.

Help and Documentation

There is typically a need for a supplement to the system itself in the shape of some user manuals. It should although be noticed that the users typically do not read the manual before they are forced to do it: when some error occurs, use of difficult unknown features etc.

The implementation of an online manual in the system is often a good solution because this manual is always at hand and it can be context sensitive. An online user manual should also be made according to the usability heuristics.

The user manual should have good index/search/task facilities and this goes for both written and online manuals. It is important to include typically user terms in the index/search/task facilities because the user will typically try these out at first in the search of the answer to a problem.

In the generation of a user manual the following three phases in a use of a manual must be considered thoroughly:

1. Search for the answer to the problem/task etc.
2. Understand what the manual says about the subject.
3. Apply the contents of the manual to the system

C. Usability Tests

In this appendix there will be a presentation of the tests, which have been used during the development phase. The tests presented in this section are tests made with participants who are chosen to be representative of the target end users. The tests are used to evaluate the system performance due to some given criteria. The typical situation will be that the criteria in the beginning of a system development phase will tend to be primary qualitative and as the developments moves forward the tests become more quantitative. There will be presented four different types of tests in this section:

- Exploratory test
- Heuristic evaluation
- Assessment test
- Validation test

Jacob Nielsen presents the Heuristic evaluation in [JN1, p. 115-155] and Jeffrey Rubin presents the other tests in [JR, p. 31-42].

The primary objective of usability testing is to:

- Identify and correct usability problems
- Create systems that are easy to learn and use
- Create systems that satisfy the user
- Create a history of benchmarks
- Minimize costs
- Increase sales
- Compete with other systems and/or proposals
- Minimize risks

In the following sections there will be a description of the different tests used in this project.

Exploratory Test

Used to *explore* something in the design. This type of test is normally used as a part of the first contact with the users.

When

The exploratory tests are conducted early in the development phase. Normally the test is conducted when the system is still in the preliminary design phase. At this stage the definition of the user should have been done. The functional specifications and early system models however might still be in a phase where they have not fully been defined.

Objectives

The goal with this test is therefore to evaluate/explore/investigate the effectiveness of preliminary design concepts, which also is known as the users conceptual model of the system. The test should also be used gain/verify knowledge of the intended end users (expertise, expectations, goal, etc.). This test might also give the designers an idea of the intuitiveness and usefulness of the system being developed. If some of the foregoing assumptions about the user or the system are wrong this could lead to usability problems later on in the design.

Methodology

The way of performing such a test is to establish a close interaction between the test participant and the test monitor to find out the consequence of the preliminary design ideas. This can be done through the use of prototypes, simulation or mock-ups of the intended system where the basic layout and organisation of functions are represented. When developing prototypes for these tests it is only necessary to concentrate on the functionality needed for the experiment.

During such a test of a prototype the test participant might perform a list of task if this is possible but is this not yet possible the user can simply "walk through" the different mock-ups and answer questions and/or comment on these mock-ups. The process for this type of test is informal and there is a lot of interaction between the test participant and the test monitor. Therefore the test monitor has the possibility to ask the test participant to give ideas about how to improve confusing areas. The essential ideas with this type of test are to focus on **why** instead of **how** and thereby reveal the participant's reactions and thoughts about the interaction.

Heuristic Evaluation

The heuristic evaluation is used as an alternative to the other tests. This means that this test is not performed by involving the end users, but will involve e.g. colleagues, which will perform the test.

When

The heuristic evaluation is most often performed in the early or middle stages of the development phase. If conducted in the early stage the first prototypes should have been defined but if the test is conducted in the middle of the design phase the fundamental or high-level design should have been established.

Objectives

The purpose of the heuristic evaluation is to determine how the user interface mock-ups or intended design are performing according to the usability heuristics given by Appendix B. It is the goal to obtain a list of usability problems from a user interface design so that these can be attended in the iteration.

Methodology

The method of conducting the evaluation is by looking at the different user interfaces and then trying to come up with an opinion about what is good and bad about the interface. First the test monitor should brief the test participant about the purpose of the test. The test itself is a walkthrough of the different user interfaces and commenting on these. After the test the test participant should be debriefed/interviewed by the test monitor for any suggestions and improvements that could be made to the system.

There is a close interaction between the test participant and the test monitor before and after the test but the only time where there should be any interaction with the participant during the test is when he/she is clearly stocked and getting unhappy about the situation.

Assessment Test

This test is used to point out the strengths and weaknesses and to get ideas of the first implementation of functionality and the user interface.

When

The assessment test is conducted either early or midway during the development process. Normally this type of test is conducted after the fundamental or high-level design has been established.

Objectives

The reason for conducting the assessment test is to expand the findings of the exploratory tests. This means that this type of test continues the process from the exploratory test though it goes deeper. It goes into the lower level functions and tests the usability of these.

The assessment test is essentially a test of the implementation of the intended design (primary **how** not so much of **why**). This means that the test should show how well a test participant could perform realistic tasks rather than explore how intuitive the system is. Therefore will the result of the test be a list of specific usability problems.

Methodology

This test is performed by letting the test participant perform a list of tasks rather than letting the participant walk through the screens and commenting. Therefore the interaction between the test participant and the test monitor will be less because the participant's actual behaviour is more important than his thought process.

Validation Test

This test should validate that the system is functioning like it is supposed to, and that it fulfils the demands set in the earlier stages in the development.

When

The validation test is conducted late in the development phase. The intention of this test is to confirm the usability of the system. This test does not, like the exploratory and assessment test, take place in the middle of the design phase. Typically the validation test takes place at the end design phase.

Objectives

The primary goal of the validation test is to validate the usability, against some predefined performance criteria or benchmark, competing products, etc. Another idea with the validation test is to make an integrated systems test where all the modules of the system are working together for the first time. Still another purpose of the validation test is to insure that there is no major flaws/problems in the system that then could force a recall of the product.

Methodology

Prior to the test the predefined criteria has to be formulated (what and how well the system should act). When this is done the criteria's for passing these earlier mentioned benchmarks or standards have to be defined. This could for example be to say that 70 percent of the participants have to complete a task within a given time limit. Prior to the test it is also necessary to define the actions that will be taken if one or more of the tasks should not meet the standard.

During the test there will be very little or no interaction between the test participant and test monitor. This test is similar to the assessment test where the test participant has to conduct some give tasks. The central focus here is also on collecting quantitative data.

Overview of the Tests and the Development Cycle

This section presents a short overview of the above presented test types. Figure 1 presents a graphical overview of the different test together with an indication of when to use in the system development. This was introduced by [LBL].

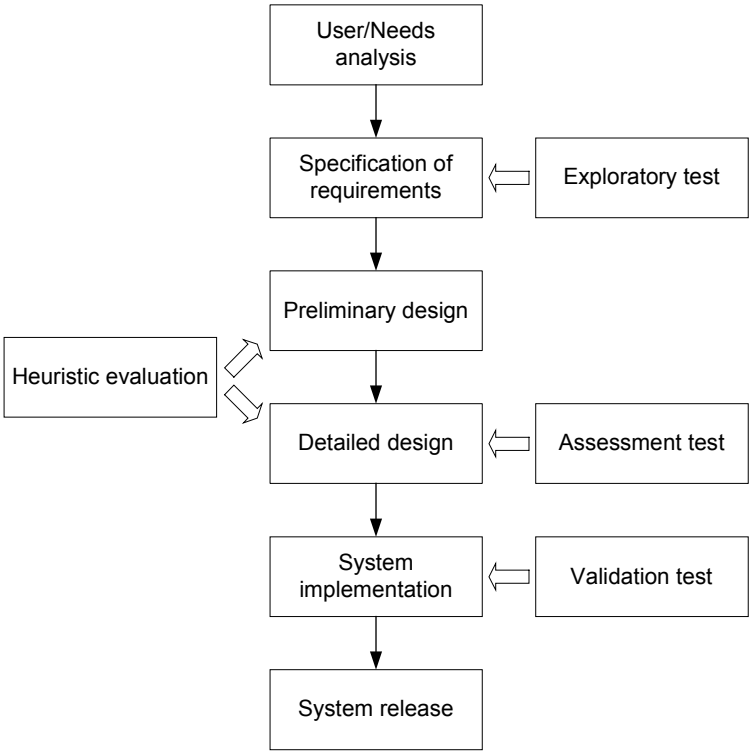

Figure 1 : Overview of the different types of test and where in the development they are used [LBL].

The first test is the exploratory test, which is conducted in the initial design phase. It has the purpose of giving some initial idea's about the users behaviour patterns and pros/cons of existing technologies. Next is the heuristic evaluation, which is used to find the usability problems in the preliminary design/mock-up. There is also conducted a heuristic evaluation when all of the detailed design has been implemented. The purpose here is again to find the usability problems. At the same time as the second heuristic evaluation is performed an assessment test is also conducted. The purpose of the assessment test is to evaluate the functionality of the intended design. At last there is the validation test that will evaluate if the usability goals for the system has been fulfilled.

D. KQML

This appendix describes the agent communication language KQML. It is based on the following papers: [YL], [TF], [RF] and [DM].

Why use KQML?

A reason for selecting KQML (Knowledge Query and Manipulating Language), as the agent communication language is that this is one of the most widely used agent communication standards [YL]. There are also a great variety of common known applications that already uses KQML as the communication language (JATLite, JAFMAS, Jackal etc.). Furthermore the KQML has shown to fulfil the needs and demands for the agent communication language used in this project.

The need for a common language is equally important in the communication between humans and in the communication between agents in a multi agent system. The reason is that the agents (and humans) must share a common language in order to be able to exchange information and knowledge. The agent language must naturally obey some common syntax rules, but furthermore the agents need to share a common understanding of knowledge in order to interpret their messages: They need a shared ontology.

Intelligent agents are typically able to express their beliefs, desires and intentions. This means that the language used should be able to handle a communication that consists of these issues. One way to obtain this sort of communication is through conversations and speech acts, which in contrast to simple message parsing allows agents to express their beliefs, desires and intentions in a dialogue with other agents. A simple example of a dialogue could be:

- Agent A asks agent B about the price of a product that B according to A should know.
- Agent B can then within the same speech act either give A the product price, ask for additional information about the product or say that the information is not in agent B's knowledge database.

As it can be seen the speech act can take several turns. The speech act will typically be terminated if agent B knows the answer immediately or if agent B does not know the answer at all. If agent B on the other hand need some more details about the product then it is possible to ask agent A about additional information about the query.

The agents also need to know how to communicate with other agents. This implies where and how to find other agents to communicate with.

The issues mentioned need to be dealt with by the agent communication language. One answer to the issues can and will in this project be the use of KQML.

The KQML Language

KQML is an agent communication language that focuses on an extensible set of KQML messages defining some permissible speech acts. A KQML message is called a performative (the message is intended to *perform* some action by virtue of being sent). The reserved performatives given by [TF] are not meant to be a closed set of performatives; an agent that uses KQML will typically only understand a subset of the reserved performatives and it is possible to make additional performatives if necessary. The only constraint is that performatives that are already defined in the KQML standard must satisfy the given specifications.

There are three layers in a KQML message (see Figure 2):

- Communication layer: This layer takes care of the low level communication of a KQML message (typically by the use of TCP/IP).
- Message layer: The message layer contains the KQML performatives.
- Content layer: This layer holds the content of the communication. The language used in the content layer is variable (specified by the communicating agents) and is specified in the message layer.

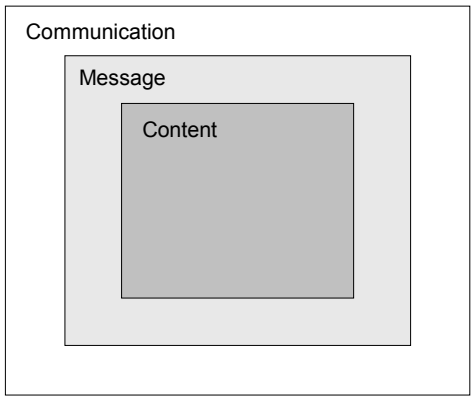

Figure 2 : The three layers into which the KQML language can be divided. It consists if the communication layer, the message layer and the content layer [DM].

Each performative is expressed as an ASCII string by the use of a Lisp-like syntax (balanced parenthesis list). The first word in the message specifies the performative type followed by a number of expressions or parameters each beginning with a keyword.

When using KQML an agent appears as if it manages a knowledge base. This knowledge base is not necessarily structured as a knowledge base, and this therefore calls for a translation of

this representation. This translation into a knowledge base abstraction is called the Virtual Knowledge Base (VKB) of an agent. KQML is the language used to communicate the contents of an agent's VKB.

The KQML (and other agent communication languages) is used to express agent 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. The performatives given by the KQML standard are introduced in the performatives section. The agent communication can be done in several ways when using KQML and the possibilities are described in the following section. The subsequent section will present the KQML language in details along with some examples of KQML messages. There are a number of keywords in a KQML message, which are described in the performative parameters section.

Performative Parameters

A KQML message is composed from a number of parameters and these parameters are identified by some given keywords. These parameters contains the content of the performative together with some additional information that describes the content and maybe the sender and receiver of the performative. The meanings of the most common performative parameters are presented in the table below [TF]:

Keyword	**Meaning**
:receiver	The receiver of the performative.
:sender	The sender of the performative.
:content	The information about which the performative expresses an attitude.
:reply-with	Indicates whether the sender expects a reply, and if so, a label to be used for the reply.
:in-reply-to	The expected label in a reply (given by the `:reply-with` parameter).
:language	The name of the representation language of the `:content` parameter.
:ontology	This parameter defines the name of the ontology used in the `:content` parameter. The agents must as already mentioned use a shared ontology in order to agree upon the meanings of the `:content` field (e.g. a term like "agent" can have different meanings depending on the semantics: software agent, CIA agent, etc).

The parameters presented above are reserved in a way that any use of them must be consistent with the meaning in table above.

Communication Example

The performative parameters and their use will now try to be exemplified. This is done by an example of how the communication between to agents may take place. Therefore is there below given an example of a speech act between two agents:

(a)

```
(ask-one
:sender joe
:content (PRICE IBM ?price)
:receiver stock-server
:reply-with ibm-stock
:language LPROLOG
:ontology NYSE-TICKS)
```

(b)

```
(tell
:sender stock-server
:content (PRICE IBM 14)
:receiver joe
:in-reply-to ibm-stock
:language LPROLOG
:ontology NYSE-TICKS)
```

[RF]

The above speech act consists of two performatives. The first performative is agent "joe" that asks "stock-server" about an IBM stock price. The "ask-one" performative tells the receiving agent that agent "joe" wants only one answer to the question given in the `:content` field. The second performative ("tell") is used to indicate that the contents of the message is in the sender agent's Virtual Knowledge Base (VKB). As it can be seen the label for the speech act is given by the `:reply-with` parameter as "ibm-stock". The ontology and language are both defined by the sender agent.

Performatives

Agents communicate via a standard set of KQML performatives, which specify a set of permissible actions that can be performed on the recipient agent.

In the KQML specification no less than 42 reserved performatives are defined. These performatives will not be described in details here but a deeper description of the single performatives can be found in [TF].

Performative names can be thought of the actions to be done with the content of the KQML message. These performative names are also reserved in the same respect as the reserved performative parameters. Below there is a presentation of the performatives together with a presentation of the main groups, which the performatives can be split into:

- **Basic informative performatives** is a way to communicate information. The three informative performatives are `tell`, `deny`, and `untell`. The `tell` performative returns positive information and `deny` returns a negative response. The `untell` performative specify that once true information is no longer true.
- **Basic query performatives** are used to ask other agents for information. There are seven basic query performatives. They are `evaluate`, `reply`, `ask-if`, `ask-about`, `ask-one`, `ask-all`, and `sorry`. The `evaluate` performative asks for a simplified representation of the information in the `:content` field and `reply` can be used to return this representation. The `ask-if` performative asks if the agent know anything about what is in the `:content` field. The `ask-about` performative is the same as the `ask-if` performative except that its response is all of the recipients knowledge about what is in the `:content` field. The `ask-one` and `ask-all` performatives are almost the same as the `ask-if` and `ask-about` performatives. The difference is that they have an additional field to ask for more specific information. At last the `sorry` performative specifies that sender understands the request, but has no answer.
- **Multi-response query performatives** are used to get several response oppose to one response in the basic query performatives. The two multi-response query performatives are `stream-about` and `stream-all`. The `stream-about` and `stream-all` performatives are equal to the `ask-about` and `ask-all` performatives except that they are asking for a series of performatives instead of a single performative with all of the information.
- **Basic effector performatives** are used to attempt to change the state of the receiver. There are two effector performatives, `achieve` and `unachieve`. The `achieve` performative which requests that the recipient try to make the sentence in the `:content` field true. The `unachive` performative which is a response to the `achieve` request if the operation was not successful.
- **Generator performatives** are like multi-response query performatives used to get information and these are `standby`, `ready`, `next`, `rest`, `discard` and `genetator`. The `standby` performative asks the receiver agent to prepare its response. When the receiver agent has prepared the information then it uses the `ready` performative. The sender will then use the `next` performative to get one of the prepared response or the `rest` performative to get all of the prepared/remaining responses. There is also the `discard` performative, which the sender can use to cancel the remaining responses. The `generator` performative is like the `standby` performative with an additional field to ask for more specific information.
- **Capability-definition performatives** are used to advertise agents' capabilities. The only capability-definition performative currently is the `advertise` performative which indicates that the sender is suited to process the performatives in the `:content` parameter.
- **Notification performatives** are used to keep the sender up to date on the state of something. The notification performatives are `subscribe` and `monitor`. The `subscribe`

performative indicates that the sender wishes the recipient to tell it about future changes and maybe new data. The `monitor` performative is much like the `subscribe` performative with an additional field specify more specific information.

- **Networking performatives** make it possible to pass directives to underlying communication layers. The networking performatives are currently `register`, `unregister`, `forward`, `broadcast`, `pipe`, `break` and `transport-address`. The first networking performatives is `register` which indicates that the sender can deliver performatives to the agent named by the `:name` parameter. Next is `unregister` that is the same as deny of a `register`. `Forward` indicates that the sender wants the `:to` agent to process the performative in the `:content` as if it came from the `:from` agent directly. The `broadcast` performative indicates that the sender agent wants the receiver agent to route the `broadcast` performative to each of the agents other agents in the system. The `pipe` performative tells that future messages to a given agent shall be routed to the `:to` agent as if the `:to` agent and the `:from` agent were directly connected. The `break` performative is used to break a `pipe`. At last there is the `transport-address` is used to define an association between a symbolic name for a KQML agent and a transport address.
- **Facilitation performatives** are used to acquire the services of another agent in some way. There is six facilitation performatives and these are `broker-one`, `broker-all`, `recommend-one`, `recommend-all`, `recruit-one`, `recruit-all`. The `broker-one` and `broker-all` performatives are used to process the request through one or all particularly suited agents. The `recommend-one` and `recommend-all` performatives are used to get the name of one or all the possible agents suited for the task. The last two performatives, `recruit-one` and `recruit-all`, are like the broker-one and broker-all performatives except the agent's responses are sent back to the original sender directly.

Now after having introduced the performatives, there will be an example of how these performatives actual works in a communications act.

Communication Between Agents

In the foregoing section the performatives by which agents communicate where introduced together with the necessary performative parameters. After this introduction there will now be a presentation of how they can be used in the communication between agents. But before that happens there will be a short introduction to some possible agent communication protocols and after that there will be a presentation of the possible ways agents can contact and communicate with each other by the use of the performatives presented.

Communication Protocols

There are a lot of information exchange protocols that can be used within agent communication. These protocols can be split into synchronously and asynchronously transactions. A synchronously transaction is where a client sends a query to a server and then waits for reply. Asynchronously transactions are where a client subscribes to a server's output. Messages can also be broadcasted out to a number of hosts instead of sending it to a specific host.

Agent Communication

There are several types of agent communication that can be used when agents have to communicate. It ranges from simple point-to-point protocol to more complex types that include the use of a facilitator agent.

A facilitator agent is an agent that maintains a registry of service names; forward messages to named agents or provide matchmaking services between information providers and clients.

Below there will be a presentation of five different types of agent communication. The presented communication uses a number of the above presented performatives to communicate the necessary information around. The list below presents the five different performatives used in the communication examples in this section.

- Ask
- Subscribe
- Broker
- Recruit
- Recommend

The communication examples below are based on a case where an agent A wants to get an answer to a question X. The answer to agent A's question is located in agent B knowledge base. Now to solve this task the agents can use the available facilitator agent F.

The fist type to look at is the simple point-to-point communication where an agent is aware of the other agents' existence and vice versa. If this is the case the agent has the possibility e.g. performing a query to another agent to get some information. This is also seen in Figure 3, where agent A sends an ask performative to agent B, which replies to A with a tell performative. In this case the facilitator agent F is not used.

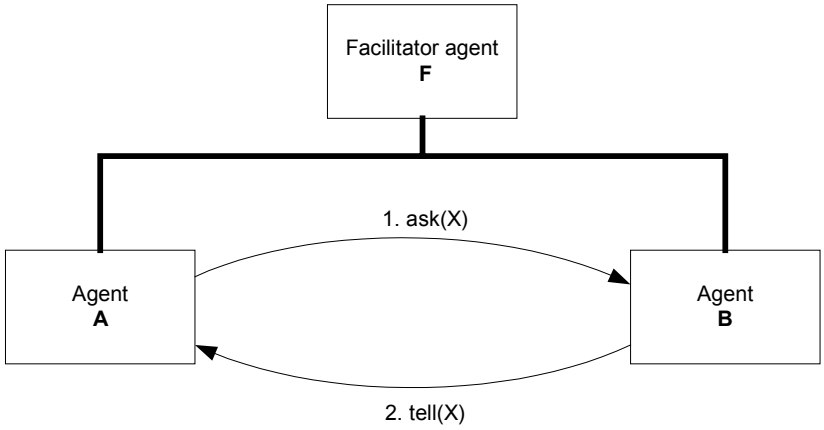

Figure 3 : Point-to-point communication between the agents A and B.

However if the is agents are not aware of each others existence or about which agent that may have the answer to a given question it is possible to use a communication like the one in Figure 4. In this example the agent A, which is seeking an answer to the question X uses subscribe to request the answer to its question from the facilitator agent F. Now the facilitator will tell agent A the answer as soon it gets aware of it but the only way for the facilitator to get the answer is to let agent B update the knowledge base of the facilitator. When this has happened the facilitator will then tell A the answer.

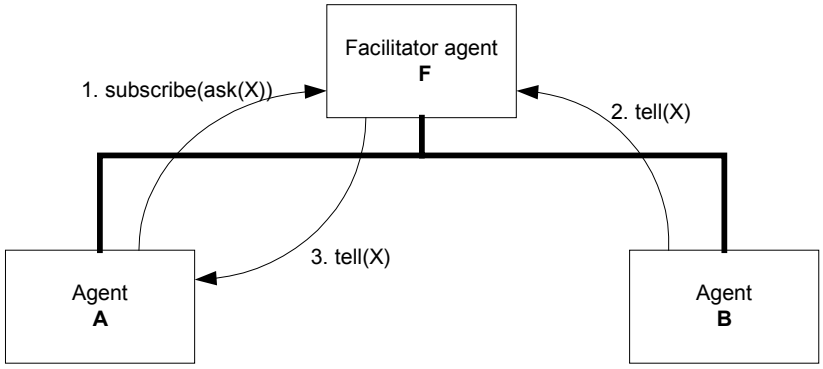

Figure 4 : In this example of agent communication A asks F to monitor for changes in its knowledge base.

Another possibility instead of subscribing on the answer to the question from the facilitator is by letting agent A ask the facilitator to find an agent that can answer agent A's question. This is done by letting agent A use the broker performative like on Figure 5. When the facilitator receives the advertise from agent B telling its ability to answer A's question the facilitator will ask agent B about the answer. Agent B will then tell the answer to the facilitator, which then tells agent A the answer.

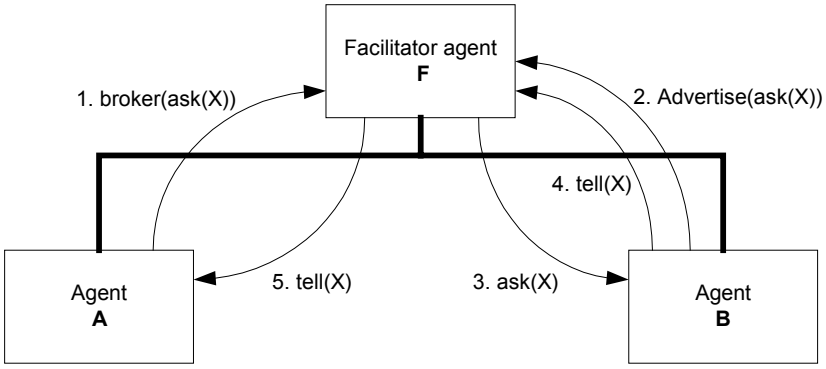

Figure 5 : Agent A asks via the broker performative F to find an agent that can perform a given performative.

A slightly different approach to solve the problem with finding an answer to the question of agent A is by using the recruit performative as on Figure 6. Here agent A informs the facilitator about its interest in getting the answer to its question. The recruit performative that is used here tells the facilitator to forward the question embedded in the performative to an agent that possible of answering it. This will in this case be the agent B. The recruited agent B will now tell its answer directly to the original sender, agent A.

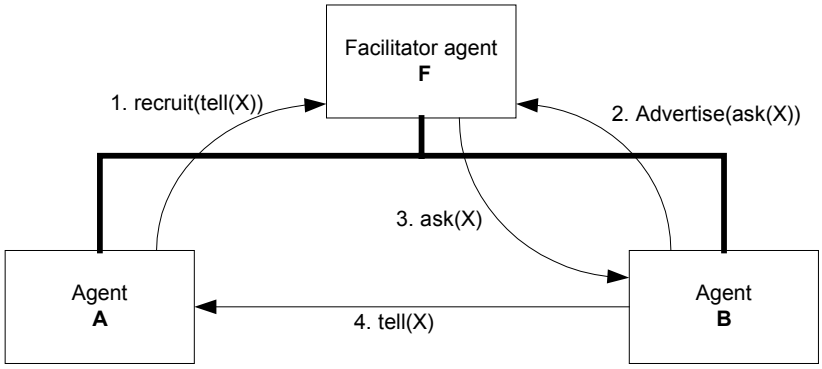

Figure 6 : This shows an example of the recruit performative, which is used to ask F to find an agent that is able to process the embedded performative.

The last of the five communication types that will be presented is the one shown in Figure 7. Agent A is still seeking for the answer to its question. To get an answer the agent A asks the facilitator agent to recommend an agent that advertises that it is able to answer the question. The facilitator will reply to agent A with the name of the agent that is suitable of answering the question, in this case agent B. Now that agent A is aware of an agent that can answer the question it may use the simple point-to-point communication described above where agent A asks agent B directly for the answer. It is also possible that agent A may sent a subscribe to agent B so that agent B informs agent A every time the answer to the question changes.

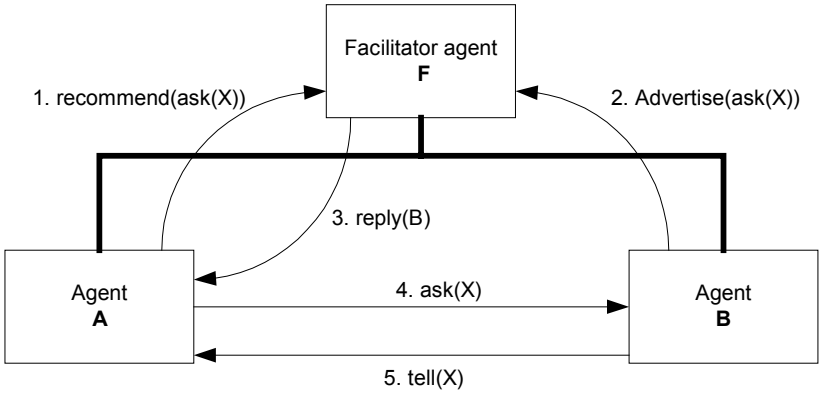

Figure 7 : This figure shows what happens when the facilitator agent receives a recommend performative.

The above shown examples has made it clear that a main task for the facilitator agent is to help other agents with finding appropriate agents that can perform the necessary performatives.

E. XML and XSL

This appendix will give an introduction to XML and XSL. The XML is designed to describe and contain data and XSL is the style sheet language of XML and it is used to transform XML into another document e.g. HTML, XML, PDF, etc.

XML

XML (eXtensible Markup Language) is a standard for representing data. Data is stored as normal text, where tags surround the data. The tags are named to describe the data. An XML-element look like this:

```
<description>data</description>
```

A start tag and an end tag surround the data. The programmer decides the tag names, so it is a very open standard. But then again there is some restrictions on how the XML must look like, which is very strict, not like HTML which is not very sensitive to errors. In HTML the parser ignores if there is a tag it does not understand, and rarely generate errors, it just displays something different. If XML encounters an error only an error message is shown. Some of the restrictions in XML look like this:

- Every start tag must have an end tag.
- Only one tag surrounding the entire XML, the root element.
- XML is case sensitive.
- All tags must be properly nested

Properly nesting look like this:

```
<program>
  <title>News</title>
</program>
```

Not like this:

```
<title>
<program>
News
</title>
</program>
```

An XML-element can contain more XML-elements, like this:

```
<programlist>
  <program>
    <id>10224219</id>
    <title>Bye Bye Bluebird</title>
    <startdate>20010224</startdate>
    <starttime>0620</starttime>
```

```
      <duration>1:40</duration>
      <channel>TV 1000</channel>
      <category>Film</category>
      <subcategory>Drama</subcategory>
      <description>Efter at have boet nogle år i udlandet…</description>
      <showview>154878</showview>
    </program>
    <program>
      …
    </program>
</programlist>
```

The top node in this XML-document is the programlist tag and that node contains several program elements.

There are some escape characters in XML:

Escape character	Character	Description
&lt;	<	less than
&gt;	>	greater than
&amp;	&	Ampersand
&apos;	'	apostrophe
&quot;	"	Quotation mark

If there is the need of having many of the characters that have to be escaped, then there is the possibility to insert a CDATA section. Such a section makes the parser able to read the escape characters without the need to escape them. A CDATA section looks like this:

```
<![CDATA["
this is the text that will be ignored,
here you can put <<<&&'"<"> all you want
"]]>
```

These types of section can e.g. be used to insert regions of JavaScript.

The XML-document should start with a tag either describing that this is an XML-file, like this:

```
<?xml version="1.0"?>
```

Or with a reference to the XSL-file formatting the data, in the client browser, like this:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

The reference is a reference to a namespace (xmlns) located at some URL.

XSL

XSL (eXtensible Stylesheet Language) can be used for different purposes, for transforming the XML-file, sort and filter XML-files and/or format XML-files, e.g. to a HTML-file which can be viewed in a browser. The XSL-file is stored as an XML-file it self. There are more possibilities with XSL, but these will not be described here (see [W3] for details). XSL is actually three different languages [W3]:

- XSLT is a language to transform XML
- XPath is a language to define XML parts or patterns
- XSL Formatting Objects is a language to define XML display

The version of XSL used is MSXML 3.0 from Microsoft; this should be 100% compatible with the W3C recommendation. For more info see [XML].

The XML-document is build up like a tree, with different nodes in this tree; these nodes are matched in the XSL-document. First in the XSL-file the root node should be matched, like this:

```
<xsl:template match="/">
</xsl:template>
```

Inside this match of the root node, the start of the HTML can be placed, this is what will be shown in the browser:

```
<HTML>
   <BODY>
      <xsl:apply-templates select="Programlist">
      </xsl:apply-templates>
   </BODY>
</HTML>
```

The content of the script tag will be explained later. The interesting tag here is the apply-templates, this will force the XSL-file to apply the template specified, this template is the defined later in the XSL-file. The templates match the nodes in the XML-document. This is the template applied before:

```
<xsl:template match="Programlist">
   <xsl:apply-templates select="program">
      <xsl:sort select="priority"/>
   </xsl:apply-templates>
</xsl:template>
```

When the Program list node template is matched, there are program nodes available. The program nodes are then sorted by priority, via. The xsl:sort element. The program elements will then be applied to another template, like this.

```
<xsl:template match="program">
    <xsl:value-of select="title"/>
    <xsl:value-of select="startdate"/>
</xsl:template>
```

Then finally the value of the different nodes, which is the actual data, is printed out via. the xsl:value-of element.

XPath

An example of the XPath standard used to reach some specific nodes:

```
//xsl:apply templates[@select='program']/@select
```

The // in the start indicates that this is the root of the XML-document, the next “xsl:apply templates” is the name of the element, which means that this XPath should find the elements in this XML-tree which has the name “xsl:apply templates”. Notice that the XSL-file is the XML-tree that is been examined. The next that happens is a filter, filtering out the nodes which has an attribute (@ means attribute) called “select” which equals “program”. What is wanted in the end, is the select attributes in all the nodes which is called xsl:apply templates, which has a “select” attribute that equals “program”.

F. Java Servlets

Java Servlets are used as a layer to provide answers to HTTP-requests, usually to provide dynamic content to websites, by providing access to e.g. databases. Java Servlets can be used instead of other Server side scripting languages like ASP, PHP, etc.

When using Java Servlets it is possible to make session variables, this means that programming with web pages no longer needs to be stateless, because it is possible to save variables for a session like in ordinary Java programming techniques. This is not possible in ordinary HTML programming because variables in e.g. client-side JavaScript is reset each time the HTML page is called. Session variables can be set and fetched with:

A way of sending data from the client browser to the Java Servlets is by using URL queries, forms and cookies. These values can also be fetched in the Java Servlets.

The web server used for the website should support Java Servlets, several web servers support Java Servlets, e.g. Microsoft IIS, Apache and others. Sun who developed Java has also made a server that supports both HTML and Java Servlets that is open-source. It can be fetched at: [SUN1]

A Java Servlet differs from an ordinary Java class in several ways. It has to be inherited from the HTTPServlet super class. Then it has to override the methods:

- initialise
- destroy
- doGet
- doPost

Initialise is called when the Java Servlet is called the first time, then different variables can be initialised, database connections can be opened, etc. When shutting down the web server, the destroy method is called, then it is possible to shut down database connections, etc. When the Java Servlet is called from a browser via a HTTP-request the doGet method is usually called, the servlet should send back a response as either HTML, XML, etc. When the Servlet is called from a HTML page where a form has been submitted, the doPost method is called instead of the doGet method, this way it is possible for the programmer to process the form data.

See Suns homepage for the specifications on Java Servlets [SUN2].

G. MySQL Data Types

This appendix will present the data types used. There are used three different data types in the databases developed for the inTelly system and these types are:

- VARCHAR
- TEXT
- INT

The VARCHAR is a variable-length string and can be declare to be any length between 1 and 255. VARCHAR only uses as many characters as needed, plus one byte to record the length. When a VARCHAR column is assigned a value that exceeds the column's maximum length, the value is truncated to fit.

The TEXT type is a binary object that can hold a variable amount of data. The TEXT type can have a maximum length of 65536 bytes and requires only as many bytes as there is characters plus two bytes. If a value is assigned to a TEXT column that exceeds the columns maximum length, the value is truncated to fit. The TEXT type column can in most respects be seen as a VARCHAR column that can be as big as 2^16.

In MySQL the keyword INT is a synonym for INTEGER. INT columns are numeric columns and the range of an INT column is –2147483648 to 2147483647 (2^31) if it is signed. The unsigned the size of the columns range is the same but the endpoints shifts up to 0 and 4294967295 (2^32). The storage requirement for an INT is 4 bytes. If an INT column is assigned a numeric value that exceeds the column type's allowable range, the value is truncated/clipped to fit the appropriate endpoint this resulting value is stored.

H. Neural Networks

The principles of neural networks will be presented in this appendix. A neural network is at some level meant to model the functionality of the brain. The neurons in the brain are connected in a network through which they send and receive electrical pulses. If the accumulated electrical inputs to a neuron exceeds a certain threshold the neuron "fires", which means that the neuron sends out electrical pulses to the neurons connected to the output of the neuron. The connections between the neurons varies dependent upon how strongly connected they are. The connections between the neurons are adjusted through a learning process.

One of the strengths of a neural network is that it can be used in situations where the structure of some problem is not well defined. An example could be recognition of patterns in a noisy image (e.g. handwritten letters). This application is called generalisation [NNO]. In such a case it would be almost impossible to write a normal algorithm that is able to detect the letters, because people write letters differently and because a person will write the same letter differently. It is possible to make this pattern recognition in a neural network that has been trained with a number of letters written by some of persons. In the learning process the neural network is presented to the training letters and their corresponding result and the weights of the neural network is adjusted accordingly.

Another typical application of the neural network is to classify data (called abstraction [NNO]). This means that a neural network is able to classify some input vectors into groups of stereotypes. (e.g. by the use of Kohonen maps).

In the following sections the generalisation method is considered. There are a number of learning algorithms of which only back-propagation is considered.

Perceptron

In 1950 Frank Rosenblatt developed a learning algorithm for a single layer network called a perceptron [GFL, p.666]. An illustration of the perceptron can be seen in Figure 8.

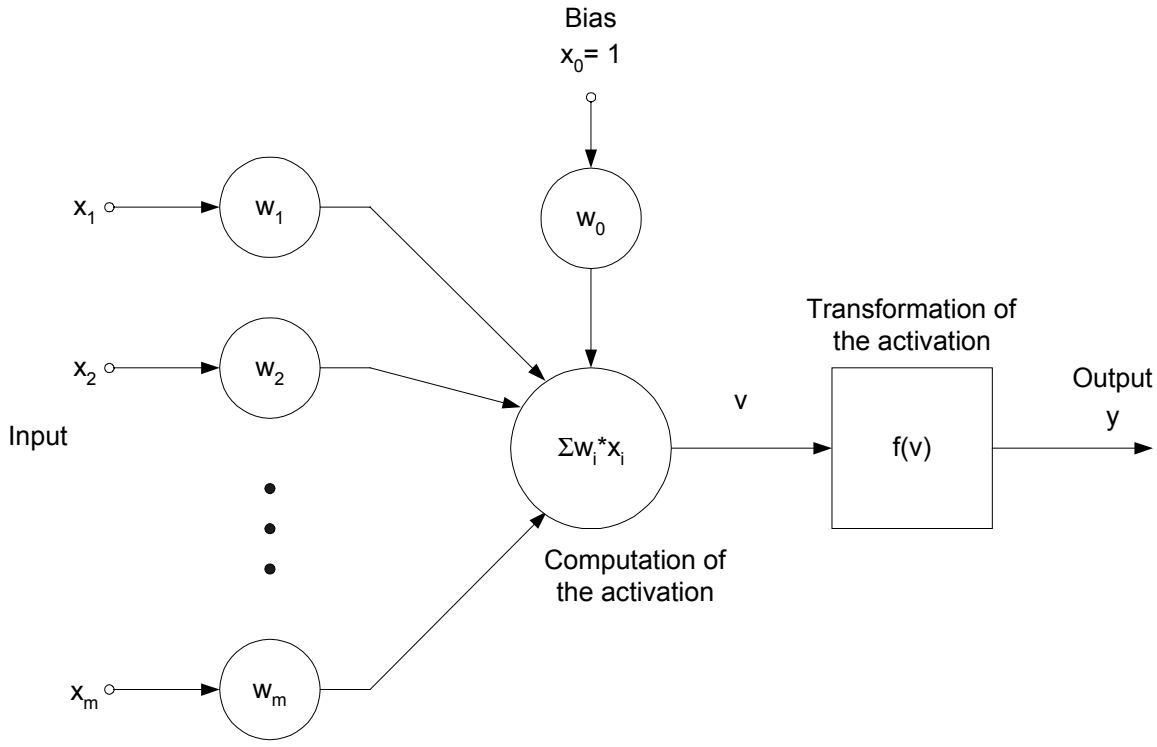

Figure 8 : A sketch of the perceptron.

The parameters in the perceptron are:

Parameter	Description
Input (x_0, x_1, x_2, .. , x_m)	The inputs to the perceptron. The values of each input will in most applications typically be in the interval between 0 and 1. Notice that the bias input is always 1.
Weights (w_0, w_1, w_2,.. w_m)	The weights determine the level of connection between each input and the perceptron. A weight close to 0 will mean that the connection is weak. The values of the weights can be any real number.
Calculated activation (v)	The activation is calculated by summarising the weighted input values. This value can be any real number.
Output (y)	The output of the perceptron. The output value is a transformation of the activation. The transformation function (also called the activity function) is typically one of the functions illustrated in Figure 9 and the output value interval is dependent upon the function used.

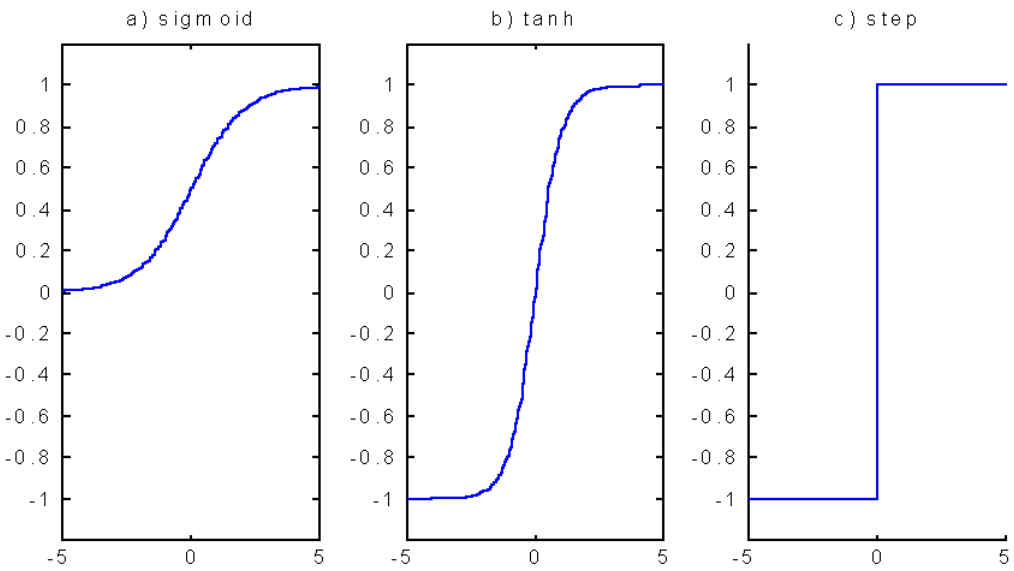

Figure 9 :
a) The sigmoid (or logistic) activity function. f(x) = 1/(1 + exp(-x)).
b) The hyperbolic tangent activity function. f(x) = tanh(x) = 2 / (1 + exp (-2x)) – 1.
c) The step activity function. Notice that this activity function is not differentiable.

In this project only the sigmoid (or logistic) activity function will be considered. The main reason is that the sigmoid is typically used as the activity function due to the fact that it has a simple first derivative, which makes it possible to use the delta rule (see below 0. Delta Rule). In this project the back-propagation learning algorithm will be presented and used.

Single and Multi Layer Network

A neural net consist of a one or more layers. In Figure 10 there is a presentation of a single and a multiplayer network. It should be noticed that the number of weight layers in this project is chosen to indicate the number of layers in the network.

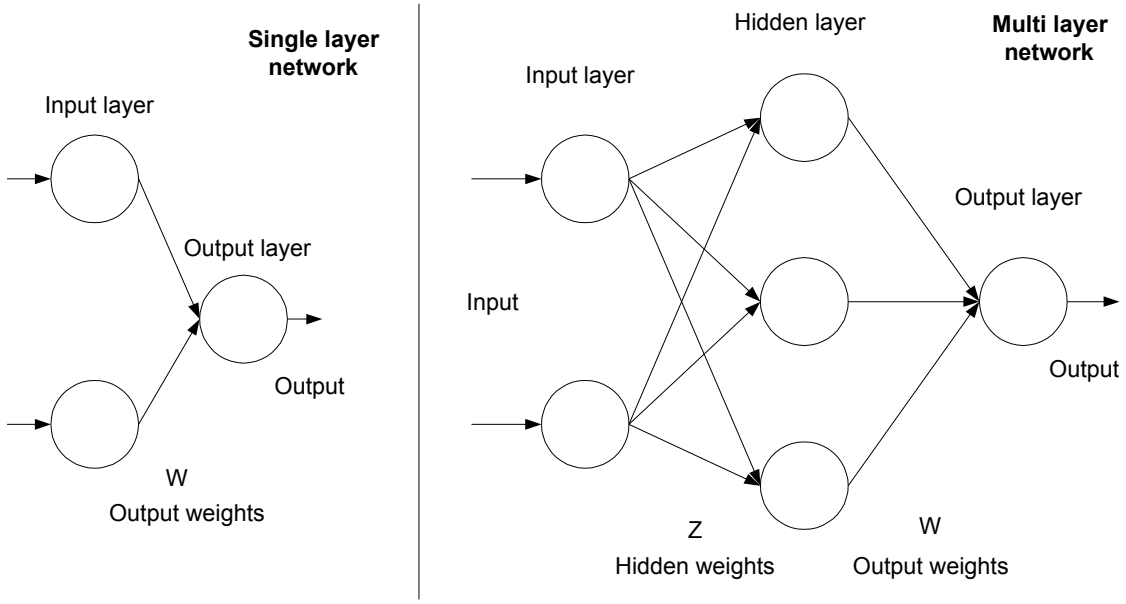

Figure 10 : An example of a single layer and a multi layer network (two layers). The number of layer weights indicates the number of layers in the network. This means that the single layer network consists of an input layer and an output layer, while there is added one or more hidden layers in the multi layer network.

The number of layers determines the characteristic of the decision boundary. The decision boundary is a figure that illustrates a complexity of the relationship between the input and the output of the neural network. The more layers the more complex input-output relationship could be achieved in the network. In Figure 11 there are shown three examples of different decision boundaries [CMB].

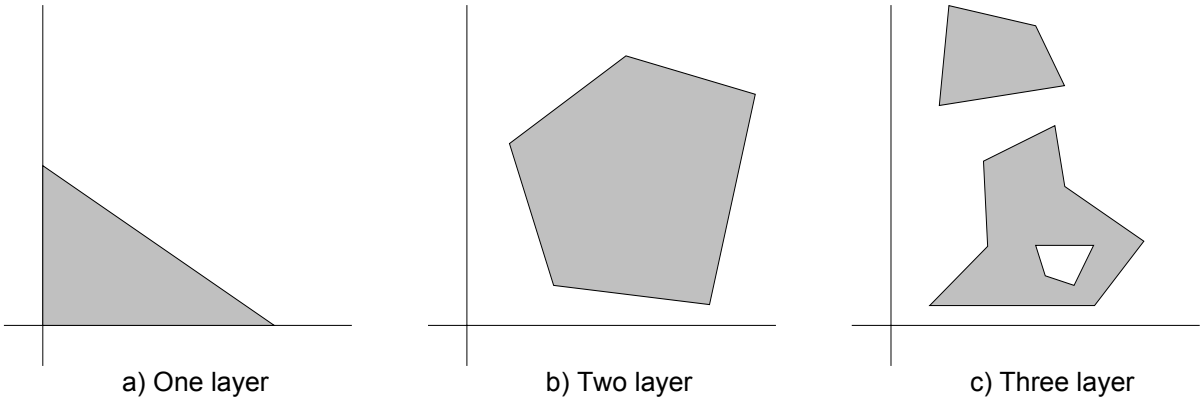

Figure 11 : Examples of different decision boundaries. The shape is dependent upon the number of layers in the neural network. For simplicity the examples presented can be achieved in neural networks having threshold activity functions (e.g. the step activity function previously shown).

Delta Rule

The delta rule is based upon the idea of an error surface calculated from every possible network weight configuration. Given some network configuration it is possible to use the principle of Gradient Decent Learning to find the direction into which the error is reduced

most rapidly. In order to use the delta rule the activity function most be differentiable, which is the case for the sigmoid function.

The basis for the delta rule is the mean squared error found by summing the squared error for each node in the neural network:

$$Error = \frac{1}{2}\sum_i (t_i - y_i)^2$$

where t_i is the target value for each output node and y_i is the actual output of each node. The training vectors determine the target values. The objective is to find a formula for the rate of change of the network error as a function of changes in the weights at node *i*. The change in a particular weight *k* can be found by taking the partial derivative of the error at each node with respect to the weight w_k at that node. It can be shown that the result is [GFL, p.672-675]:

$$\frac{\partial Error}{\partial w_k} = -(t_i - y_i) \cdot f'(v) \cdot x_k$$

where *f'(v)* is the first derivative of the calculated activation and x_k is the input to the weight *k*. The minimization of the error must be in the direction of the negative gradient, which results in the following delta rule calculating the weight adjustment:

$$\Delta w_k = -c \cdot \frac{\partial Error}{\partial w_k} = -\eta \cdot [-(t_i - y_i) \cdot f'(v) \cdot x_k] = \eta \cdot (t_i - y_i) \cdot f'(v) \cdot x_k$$

η is the learning rate, which defines how much a weight moves in each learning iteration.

Back-Propagation

In a multi layer network the adjustments of the weights is complicated due to the fact that the network error only can be measured at the output nodes. It is difficult to identify the source of some error at the output nodes because the error is accumulated over a number of layers in the network. One solution to this problem is the back-propagation algorithm.

The principle of back-propagation is to propagate the error from the output nodes backwards through the hidden layers in the network. The back-propagation is based upon the delta rule. In the adjustment of the output nodes the delta rule is used directly:

$$W_{k,n+1} = W_{k,n} + \Delta w_k = W_{k,n} + c \cdot (t_i - y_i) \cdot f'(v) \cdot x_k$$

$W_{k,n+1}$ is the new value of the output layer weight.

In order to adjust the hidden layer weights Z it is necessary to make an estimate of the error, which in the output layer is given directly by t_i - y_i. The estimate is given by [HA, p.74-75]:

$$\hat{e} = W_n \cdot (t_i - y_i) \cdot f'(v)$$

W_n is the weights of the succeeding layer (e.g. the output layer in a two layer network). This results in the following calculation of the hidden layers weight adjustment:

$$Z_{k,n+1} = Z_{k,n} + \Delta z_k = Z_{k,n} + \eta \cdot \hat{e} \cdot f'(v2) \cdot x2_k$$

Z is the hidden layer weights, $f'(v2)$ is the first derivative of the calculated activation in the hidden layer note and $x2_k$ is the input to hidden layer weight Z_k.

The back-propagation algorithm can be used in a neural network with any number of layers.

Training

To train a neural network there is needed a number of input vectors together with the corresponding output vectors (also called target values). The typical training process is to loop through the input and output vectors one at a time adjusting the network weights. And when the last set of vectors have been applied then start over again until some stop criteria is fulfilled. This is typically when some given maximum summarized mean squared error is reached on the training vector sets. Another stop criteria could be to stop the training process after a predefined number of training loops (e.g. 20000), but this solution will not ensure that the error has some well-defined maximum level at the end of the training. On the other hand the first stop criteria can in some situations take very long time to reach. It is therefore appropriate to combine the two stop criteria so that the training process is stopped when one of them is fulfilled.

I. Bayesian Networks

This appendix is primary based upon [FVJ], [HUGIN] and [SA].

Introduction

Bayesian networks are also called Bayes nets, causal probabilistic networks (CPNs), Bayesian belief networks (BBNs) or simply belief networks. Bayesian networks are used to model domains that are characterized by uncertainty.

A Bayesian network consists of a set of nodes and a set of directed edges (arrows) between these nodes. Edges reflect cause-effect relations within the domain. These effects are normally not completely deterministic (e.g. disease -> symptom). The strength of an effect is modelled as a probability:

If B **then** P(A) = 0.75
If C **then** P(A) = 0.65

It is possible to read the above as a rule, but this is not correct therefore a different notation is used:

$$P(A \mid B) = 0.75$$
$$P(A \mid C) = 0.65$$

It is possible to combine the two above causes. This means that there is needed a specification of the conditional probabilities:

$$P(A \mid B, C)$$

Where B and C each can take the states **yes** and **no**. In order to build up a Bayesian network it is necessary to specify the strength of all combinations of states for the possible causes in all the nodes.

Fundamentally, Bayesian networks are used to update probabilities when information comes in. The mathematical basis for this is Bayes' theorem:

$$P(A \mid B)P(B) = P(B \mid A)P(A)$$

Qualitative and Quantitative Representation

An example of a simple Bayesian network is given in Figure 12.

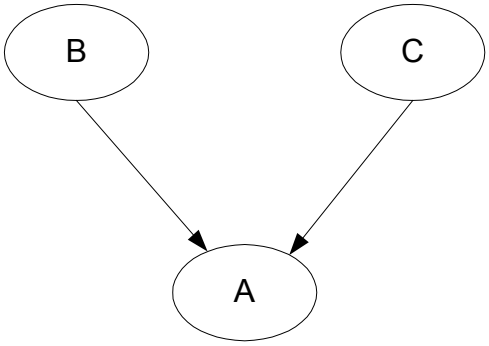

Figure 12 : A simple Bayesian network. The A node has two parent nodes: B and C. The direction of the edges represents the casual dependencies (from B to A and from C to A).

The graphical representation of the Bayesian network is the qualitative representation of a Bayesian network. There is also a need to specify the quantitative representation. An example of the quantitative representation is given in the following tables:

P(B)

B = true	B = false
0.3	0.7

P(C)

C = true	C = false
0.6	0.4

P(A|B,C)

	C = true		C = false	
	B = true	B = false	B = true	B = false
A = true	0.75	0.55	0.95	0.15
A = false	0.25	0.45	0.05	0.85

Given the qualitative and quantitative representation makes it possible to propagate evidence it the Bayesian network.

Propagation of Evidence

To propagate evidence in a Bayesian network a method given by [SA] and [HUGIN] is used. The Bayesian network is transformed into a junction tree in which all calculations are performed rather than in the original Bayesian network. The method collects the nodes in set of nodes called “cliques” separated by “separation sets”. The transformation consists of four steps [SA, p.26-27]:

1. **Generation of a moral graph**. The graph is made by adding links between all parents of each node. All directions indicated by the arrows are removed.
2. **Triangulation of the moral graph**. A graph is triangulated if all loops with length > 3 have a chord. There should be added links (undirected) until this is achieved.
3. **Form the cliques**. A clique is the largest set of nodes that are all mutually connected after the triangulation. The probability tables of each clique are initialised as the product of the relevant quantitative data.
4. **Form the junction tree**. The junction tree is made by connecting the cliques such that the connections form a tree with a junction tree property. This means that if two cliques contain a common set of nodes then all cliques on the path between them must also contain the same set of nodes.

Before there can be propagated evidence in the junction tree the cliques and separation sets need to be initialised [SA, p.27-29]:

- **Clique initialisation:** The clique is a table based upon the given quantitative data. The clique is initialised by calculating the marginal probability distribution of any of the nodes in the cliques.
- **Initialisation of the separation sets:** Each link in the junction tree has a separation set. The content of the separation set is joint probability distribution for the nodes that are common to the two cliques joined by the link.

It is now possible to propagate evidence in the junction tree. The overall principle in the propagation is to update the clique where some evidence is known and propagate the result to the other cliques through the separation sets.

Example

In this section there is a presentation of calculations on a simple Bayesian network. In Figure 13 the Bayesian network is illustrated.

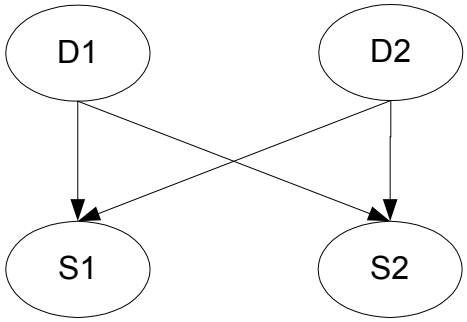

Figure 13 : Bayesian network. D1 and D2 could e.g. represent two different deceases. S1 and S2 could represent two different symptoms.

The quantitative data is given in the tables below:

P(D1)

D1 = true	D1 = false
0.1	0.9

P(D2)

D1 = true	D1 = false
0.05	0.95

P(S1|D1,D2)

	D1 = true		D1 = false	
	D2 = true	D2 = false	D2 = true	D2 = false
S1 = true	0.7	0.6	0.9	0.2
S1 = false	0.3	0.4	0.1	0.8

P(S2|D1,D2)

	D1 = true		D1 = false	
	D2 = true	D2 = false	D2 = true	D2 = false
S2 = true	0.4	0.1	0.9	0.3
S2 = false	0.6	0.9	0.1	0.7

The moralisation of the graph is presented in Figure 14. The directions of the edges are removed and there is added a link between the parent nodes.

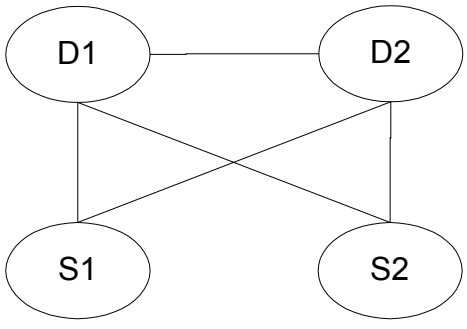

Figure 14 : The moralised graph.

The graph is triangulated without adding further links (there are no loop with a length > 3). In Figure 15 the resulting cliques and junction tree is presented.

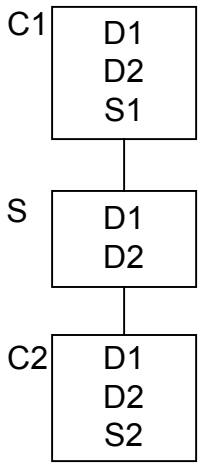

Figure 15 : The resulting junction tree containing of two cliques (C1 and C2) and one separation set (S).

The separation set connecting the two cliques consists of two nodes: D1 and D2. The next step is to initialise the cliques and the separation set:

C1

		S1	
D1	D2	true	false
true	true	0.0035*	0.0015**
true	false	0.057	0.038
false	true	0.0405	0.0045
false	false	0.171	0.684

The calculations are:

*: P(D1=true) * P(D2=true) * P(S1=true|D1=true,D2=true) = 0.1 * 0.05 * 0.7 = 0.0035

**: P(D1=true) * P(D2=true) * P(S1=false|D1=true,D2=true) = 0.1 * 0.05 * 0.3 = 0.0015

etc.

As it can be seen the quantitative data are used in the clique initialisation.

C2

		S2	
D1	D2	True	false
true	true	0.002	0.003
true	false	0.0095	0.0855
false	true	0.0405	0.0045
false	false	0.2565	0.5985

The initialisation of the separation set S is determined by marginalizing one of the adjoining cliques (C1 or C2):

S

D1	D2	
true	true	0.005
true	false	0.095
false	true	0.045
false	false	0.855

If some evidence is found then it is possible to propagate this into the junction tree. One example: It is given that S1 = true. The first step is to update and normalise the clique containing the S1-node (C1):

C1

		S1	
D1	D2	true	false
true	true	0.0129*	0
true	false	0.2096	0
false	true	0.1489	0
false	false	0.6287	0

*: 0.0035 / (0.0035 + 0.057 + 0.0405 + 0.171) = 0.0035/0.272 = 0.0129
etc.

The resulting C1 is then propagated into the separation set:

S'

D1	D2	
true	true	0.0129
true	false	0.2096
false	true	0.1489
false	false	0.6287

Finally the evidence is propagated to the clique C2 by multiplying C2's table by the ratio between the new separation set (S') and the old separation set (S):

C2

		S2	
D1	D2	true	false

true	true	0.0052*	0.0077
true	false	0.0210	0.1886
false	true	0.1340	0.0149
false	false	0.1886	0.4401

*: 0.002 * 0.0129 / 0.005 = 0.0052
etc.

It is now possible to determine the probability of S2:
P(S2=true) = 0.0052 + 0.0210 + 0.1340 + 0.1886 = 0.3488

From the above example the principles of the creation and propagation of evidence in a Bayesian network is presented. In order to use the Bayesian network both a quantitative and a qualitative representation is needed. When the junction tree has been developed and the cliques and separation sets have been initialised it is possible to propagate evidence in the network.

J. Competing Products

The different competing products are presented in this appendix. There will be looked at the well-known paper TV-guides, Text TV and Internet TV-guides as competing products. There are been made some definitions to clarify what is meant if e.g. there is talked about a program. These definitions are seen in the list below:

- **Channel** – TV-station, e.g. DR1, TV 2, 3, 3+, etc.
- **Program** - A TV-Program at a station, e.g. X-files.
- **Category** - The type of a program, e.g. movie, documentary, etc.

Paper TV-guides

There are two main groups of paper TV-guides, which will be inspected in this section. The first is the newspaper and the other the magazines only containing TV-programs.

Newspaper

The newspaper TV-guide normally list their TV-programs in columns close to each other which makes it possible to put a lot of programs into one page. This type of TV-guide is good to give a quick view over a lot of stations. Another good thing about the newspaper is that it can be used when sitting in front of the TV. This sort of TV-guide normally contains no details about the programs or if there should be any details, these will only be concerning very few of the TV-programs. It is also only a few newspapers that contain a weekly overview of all the TV-programs for the next seven days. One of the negative sides about a newspaper is that it cannot be updated after they are printed.

TV-magazine

These other types of paper TV-guides are as the newspapers easy to overview and can also give a fast overview of a lot of channels. Normally these TV-magazines includes a 7-(14) days program and not as the newspaper only one day. These magazines also have a simple daily structure and there are presented more details about more TV-programs. This 7/(14) days program is a good idea when going on vacation and the VCR has to be programmed before going but again as with the newspaper you cannot be sure about them because they cannot be updated after they are printed.

Text TV as TV-guide

The text TV is also a source for finding what is coming on TV. The text TV is a part of each channel and the data presented there is only for one or two channels. Therefore it is necessary to change station for each channel there should be checked. The text TV is slow, because it has to run through all pages to find the one page containing the TV-programs. Here there is

also only a few of the programs the has attached details and if the user want to view these he/she has to change page e.g. press 311 to see details about one program, this is not intuitive.

A positive thing about text TV is that it is located where you need it, in front of the TV. Another thing about the TV-guides on the text TV is that they are always updated because the TV-stations can edit them all the time. Another advantage is that the text TV is simple to use as long as the user just wants to get an overview of the days TV-programs. An example of this is seen in Figure 16.

Figure 16 : A screen dump of DR's text TV.

Internet TV-guides

There are two main types of Internet TV-guides. The first is the TV-guide that belongs to a specific channel/station only showing this stations/channels programs. The other type is the independent TV-guide, which is not attached to a specific channel. An advantage of these Internet TV-guides is that they are available online at all times and that they are easily updated.

TV 2

The TV-guide at TV 2, do not only contain data for their own channel but also for many other channels. You find this TV-guide at http://tv.tv2.dk/ or see Figure 17. As most Internet TV-guides it has a program list and a filter with the following options:

- **Channel**: All channels, dr1, dr2, etc.
- **Time**: Today, tomorrow, 2., 3., …, 7. day; All day, 0-6, 6-12, 12-18, 18-24
- **Category**: All categories, children, documentary, movies, lottery, news, series, etc.

There is a search facility for searching in all programs for the next week. It also provide a TV right now option that makes it possible to see what is shown right now on TV.

The front page also contains a shortcut to a page with the numbers of viewers for the last seven days on all the Danish channels. At last TV 2 also have very detailed description on programs shown by them, on their own homepage.

Figure 17 : The front page of TV 2's TV-guide.

Billed-bladet

This TV-guide is found at http://www.billed-bladet.dk/tv.php3 and a screen dump presenting the front page can be seen at Figure 18. It contains data for about 60 channels. The first that is presented to the user when arriving to the site is a list of programs from Danish channels, which are being shown right now. It is possible to change this list of programs by using the filter shown to the left of the program list. In this filter it is possible to choose which time to view: Divided into two select boxes; one with the next seven days and one with: morning,

afternoon, evening, the rest of the day, rest of the week, right now, all day. It is also possible to choose which category of programs to view like movies, sport, etc. At last there is the possibility of choosing which channels to view and this is divided into language (Danish, German, etc.) or channel type (Movies, Sport, etc.).

An additional functionality on the page is the search facility that makes it possible to search the next seven days for some specific keyword. There is also the possibility of getting a short description of the program and/or to see the show view code for the program if any. This TV-guide has also the possibility of creating individual user profiles. When creating such a user profile the user first have to define if he/she is an expert user or novice user. Next step is choosing the time of day and the weekday that is of interest. Last step is to choose the channels and categories, which is done by a great number of checkboxes. This user profile is then saved as an individual defined shortcut and presented on the right side of the site. It is possible to have more than one of these individual defined shortcuts to different settings concerning channels, time, search words, etc. The programs that these shortcuts result in can be emailed to the user each morning. There are also predefined shortcuts: Movies right now, Sports right now, Danish television right now, Billed-bladet recommends.

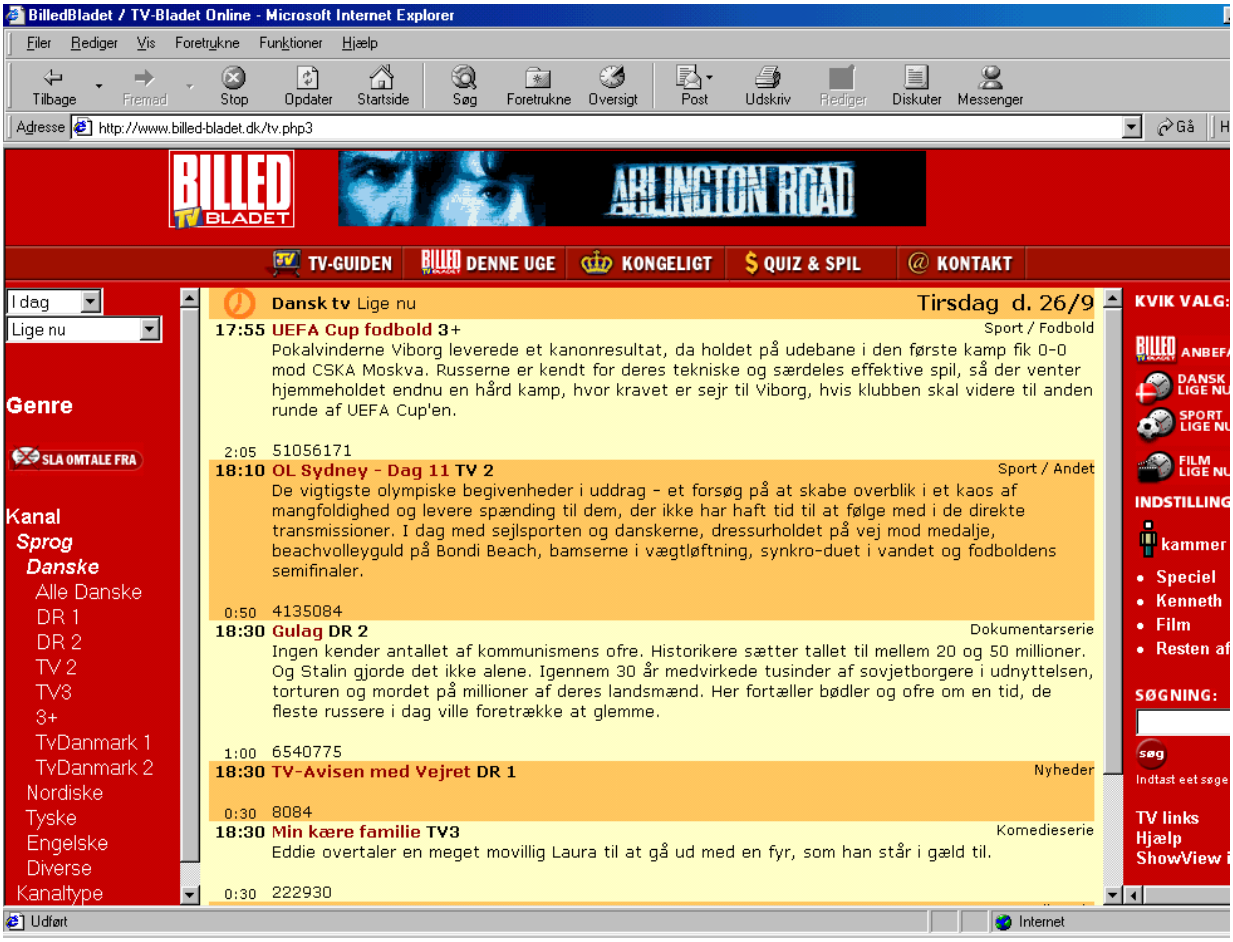

Figure 18 : Billed-bladet's TV-guide presenting the Danish programs that are shown right now. The description and show view code is also shown.

Teledanmark

The TV-guide from Teledanmark is found at http://www.kabeltv.dk/HTML/guide/cgi/guide.tvg?function=search and a screen dump can be seen at Figure 19. This TV-guide has about the same functionality as Billed-bladet's TV-guide. There is a filter where it is possible to select the time of day, the day, the channel and the category. There is a search facility and a short description is shown per default but this TV-guide at Teledanmark does not have the show view codes. The Teledanmark TV-guide also differs from Billed-bladet in their way of using the user profile. It is only possible to have one profile and when you create such a profile there is added the following to the filter:

- **Channels** - My channels
- **Time** - My time
- **Category** - My categories

The My channels and My categories options gives a number of predefined channels and categories that can be selected from a list. Here it is also possible to see which channels are available in which areas. In the My time option it is possible to specify the timeslot right as the user wishes, (e.g. 18.00-23.00), instead of using the predefined timeslots like 06-09, 09-12 etc.

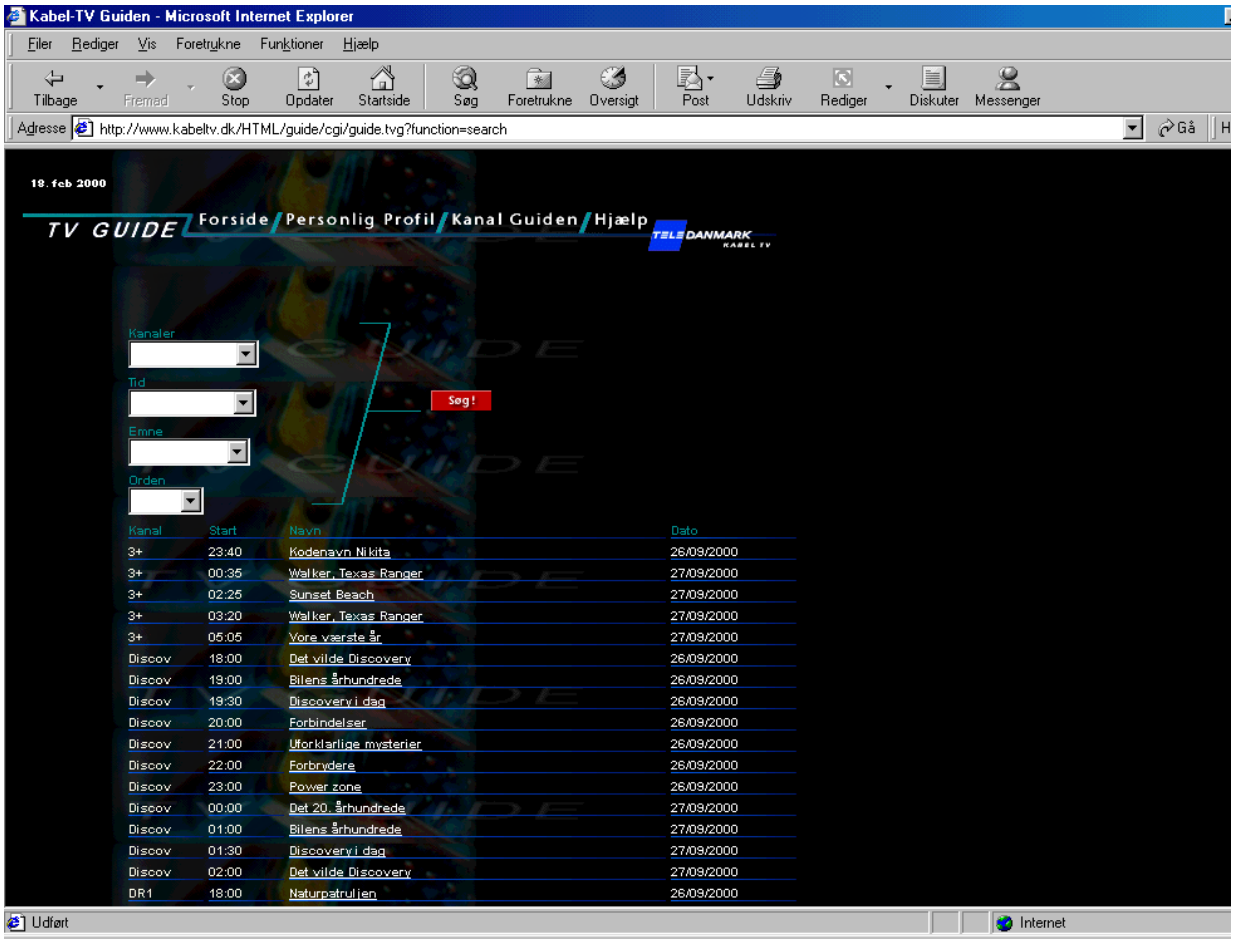

Figure 19 : Teledanmark's TV-guide showing program data for a logged in user.

Ritzaus medieservice

Another Internet TV-guide is Ritzaus medieservice, which is located at http://www.rmas.dk/ or see a screen dump at Figure 20. This site has the same standard features as the two above-mentioned TV-guides with a program list and a filter. The filter has the following options:

- **Channel**: All channels, DR1, DR2, etc.
- **Time**: A graphical calendar
- **Category**: All categories, Common, children, documentary, business, erotic, movies, etc.

Beside the above-mentioned filter options there is also a search facility to search in this weeks programs.

Besides the TV-guide, Ritzau also has a media service, which makes information about the different programs on all stations available for other sites/TV-guides, and they offer a direct connection to their database. It is also possible to make a personal homepage with a TV-guide.

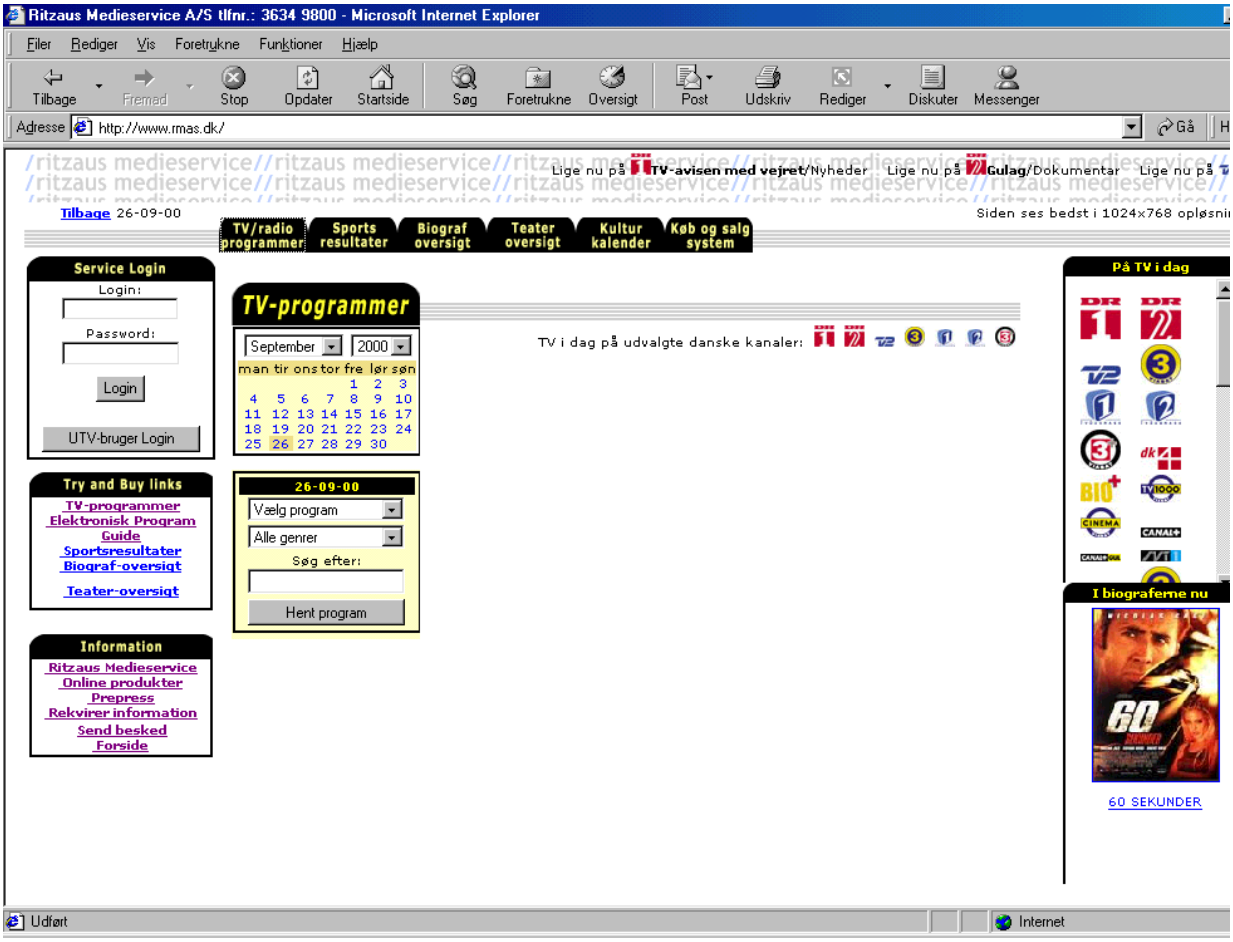

Figure 20 : The TV-guide at Ritzaus medieservice.

Suggestions for Our TV-guide

Now having looked on a number of TV-guides there will be looked upon what might be used from the different TV-guides. The above studied TV-guides showed that there should be a program list and a filter menu together with some additional functions. The first thing to look upon is the filter, which should contain some or all of the options listed below:

- **Channels** - My channels, All channels, DR1, DR2, etc and these should be divided into language (Danish, English etc.) and categories (Movies, Sport, etc.).
- **Categories** - My categories, All categories, children, documentary, movies, lottery, news, series, sports, entertainment, nature (the categories are mainly decided by the service that provides the TV-program data)
- **Time of day** - All day, 0-6, 6-12, 12-18, 18-24, the rest of the day, right now
- **Date** - Today, Wednesday, Thursday, …, Monday, rest of the week

The additional functions that could be present at a TV-guide is:

- Search facility
- Sorting by time, channel, category
- Details: On/Off
- Show view codes: On/Off
- Shortcuts to newsgroups concerning TV

The possibility of creating some sort of individual user profile could also be made. This will give the user the possibility of adjusting the TV-guide to fit his/hers needs. This user profile might contain the following:

- User type definition (expert user, novice user)
- Channels
- Categories
- Time of day
- Weekdays
- Keywords
- E-mail TV-programs

K. inTelly Homepage

In this appendix there will be a short presentation of the inTelly systems homepage. The page can be found at the address: http://inTelly.dk/project or a screen dump of the front page can be seen at Figure 21. The web site is available in both Danish as well as English.

The idea with the inTelly.dk homepage is used to present the inTelly project. The purpose of the web site is to provide information about the project and its development throughout the project period. Another intention with the homepage is also to get in touch with the possible users of the inTelly system. The first contact with the users is done in the initial design phase where there has been developed a questionnaire that is used to gather information about the users and their habits concerning news reading and find interesting TV-programs. The result of this questionnaire is presented on the homepage because this helps to keep in touch with the users that participated in this first exploratory test.

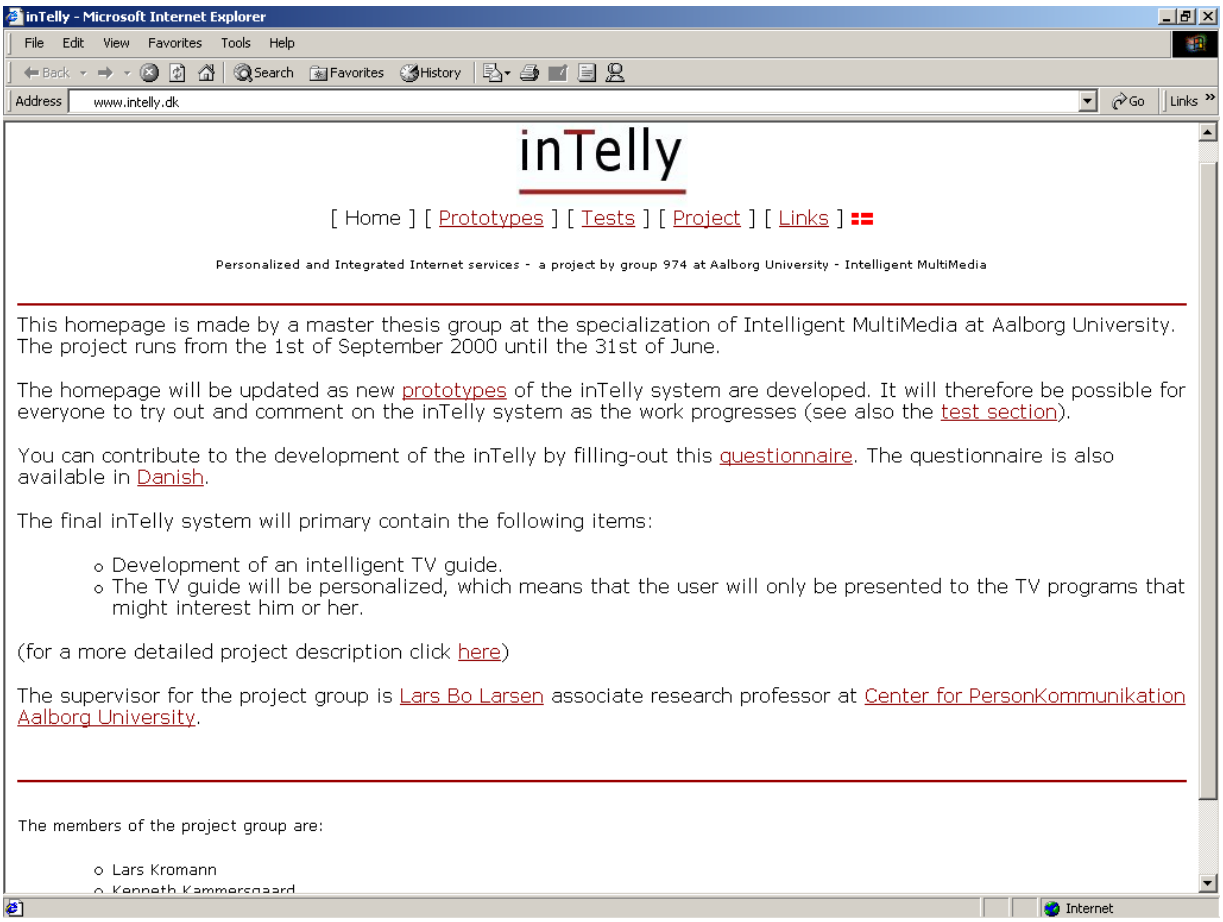

Figure 21 : The inTelly.dk homepage.

The homepage is also used to show the different prototypes that are developed during the iterative design phase so that the possible users has the possibility to commenting on the design without participating in one of the four test where the users were involved.

L. ZIP Stat - Statistics for inTelly.dk

This appendix will present the statistical data collected by ZIP-Stat (for more details see [SM]) about the inTelly.dk website. ZIP-Stat is a free web statistics service that collects a number of data about the user when he/she enters a given site. The data is used during the design of inTelly.dk to define the most used browsers and resolutions. First there will be presented data about the browsers used and next the screen resolutions are presented.

Browsers

The percentage of the different browsers used to access the inTelly.dk website is seen in the table below. It is seen that the Microsoft Internet Explorer 5.X is most used browser followed by Netscape 4.X.

Browser	Hits	Percent	Graf
MSIE v5.X	314	81%	
Netscape v4.X	63	16%	
MSIE v4.X	6	1%	
Other browsers	4	1%	

Resolution

The different screen resolutions used by the users presented are presented below. It is seen that the widely used resolution is 1024x768 that scores 61%. The second most used resolution is 800x600 and this scores 14%.

Resolution	Hits	Percent	Graf
1024x768	238	61%	
800x600	56	14%	
1280x1024	32	8%	
Other	14	3%	
1152x900	13	3%	
1152x864	9	2%	
640x480	7	1%	
1280x960	4	1%	
1600x1200	3	0.8%	
1152x870	3	0..8%	

1280x976	2	0.5%	
800x572	2	0.5%	
1280x998	2	0.5%	
1152x816	2	0.5%	

SUPPLEMENTS

M. User Interface Version 0.1

This supplement will present the first mock-up of the user interfaces. The arguments for why the user interface is designed the way it is, is presented in the *Main report – 16. inTelly Version 0.1*.

Create User

The create user interface (see Figure 22) is where a new user has the possibility of registering him or her self. The user has to type in a username and password. The password has to be typed in twice to assure its correctness. The user presses the create user button (Danish: "Opret bruger") and the user is the created.

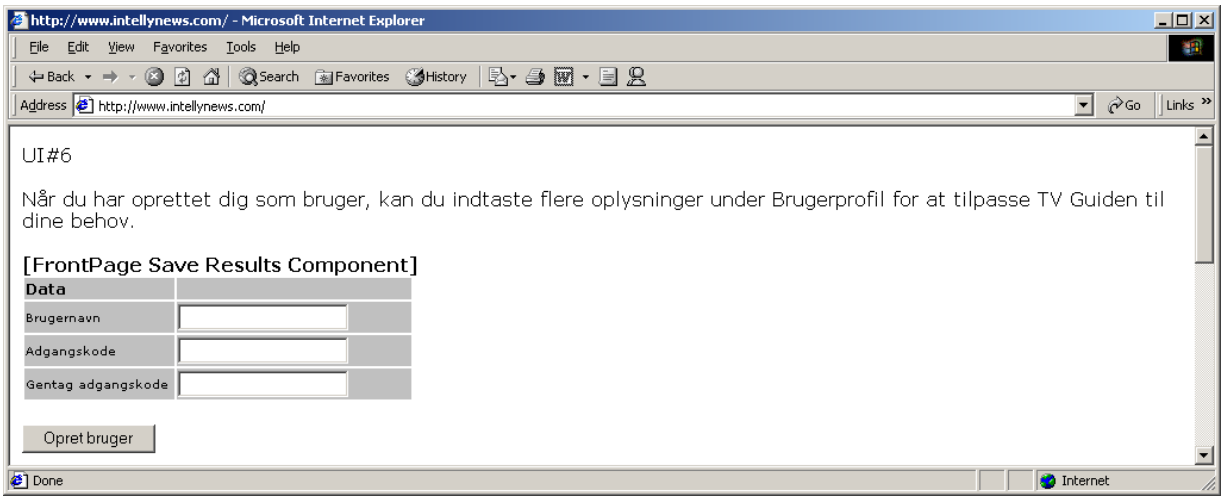

Figure 22 : Screendump of create user screen

When the user tries to create a user with an already existing user name this will return an error message and give the user the possibility of trying again (see Figure 23).

Figure 23 : The create user screen where the user has tried to create him/her self with an existing username.

User Profile

When a user has been created, he or she has to fill out the user profile to use the features of a registered user. The user profile consists of a front page (see Figure 24) describing the user profile and its purpose and six pages containing data.

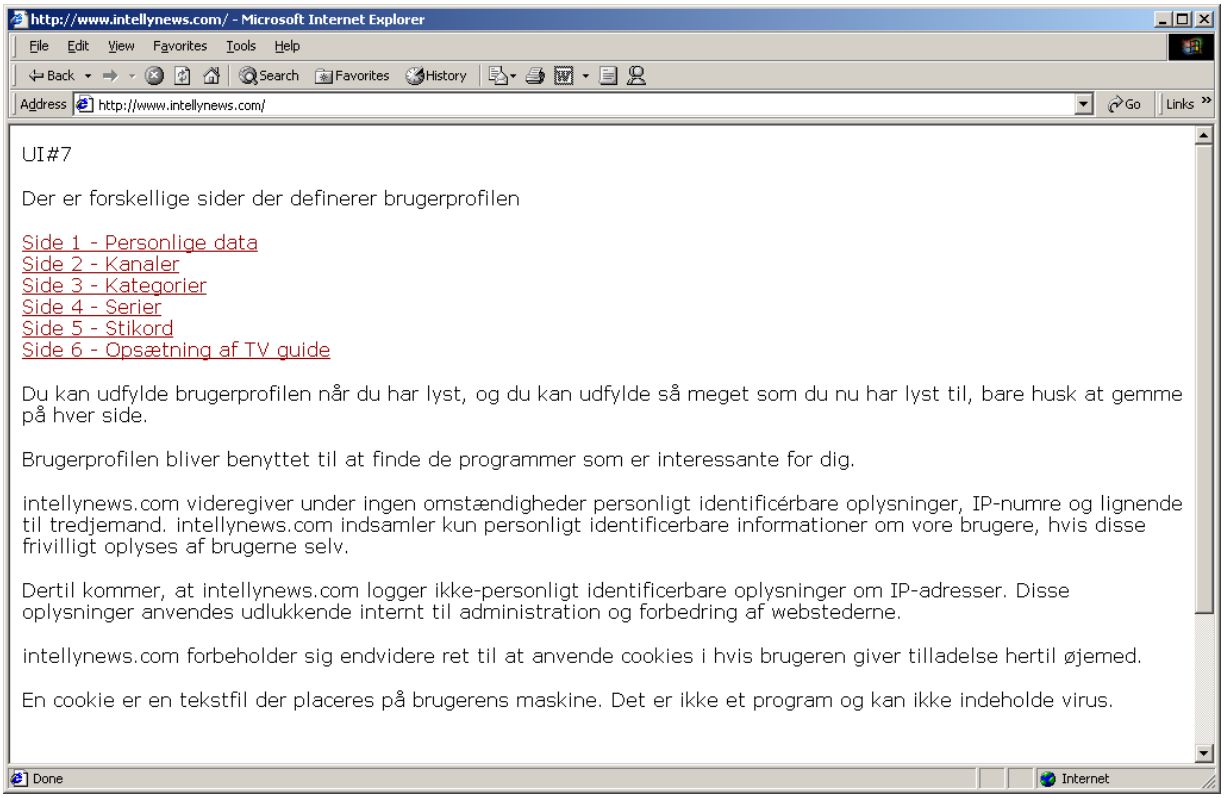

Figure 24 : The user profile front page.

The first page containing data is the page called personal data where the user enters some data about him or her self (see Figure 25).

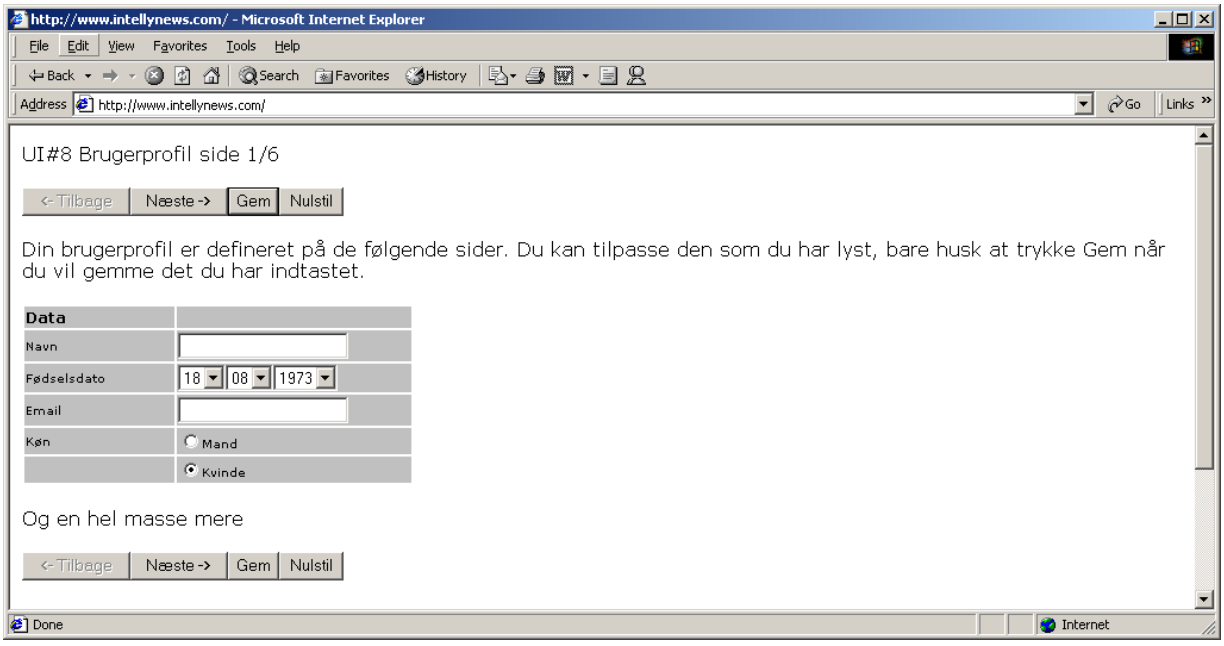

Figure 25 : The personal data page from the user profile.

In Figure 26 the channels page is presented. Here the user has the possibility of selecting the channels that he/she wants to get TV-programs presented from. The user is having the option of choosing the specific channels one by one or choosing a whole group of channels.

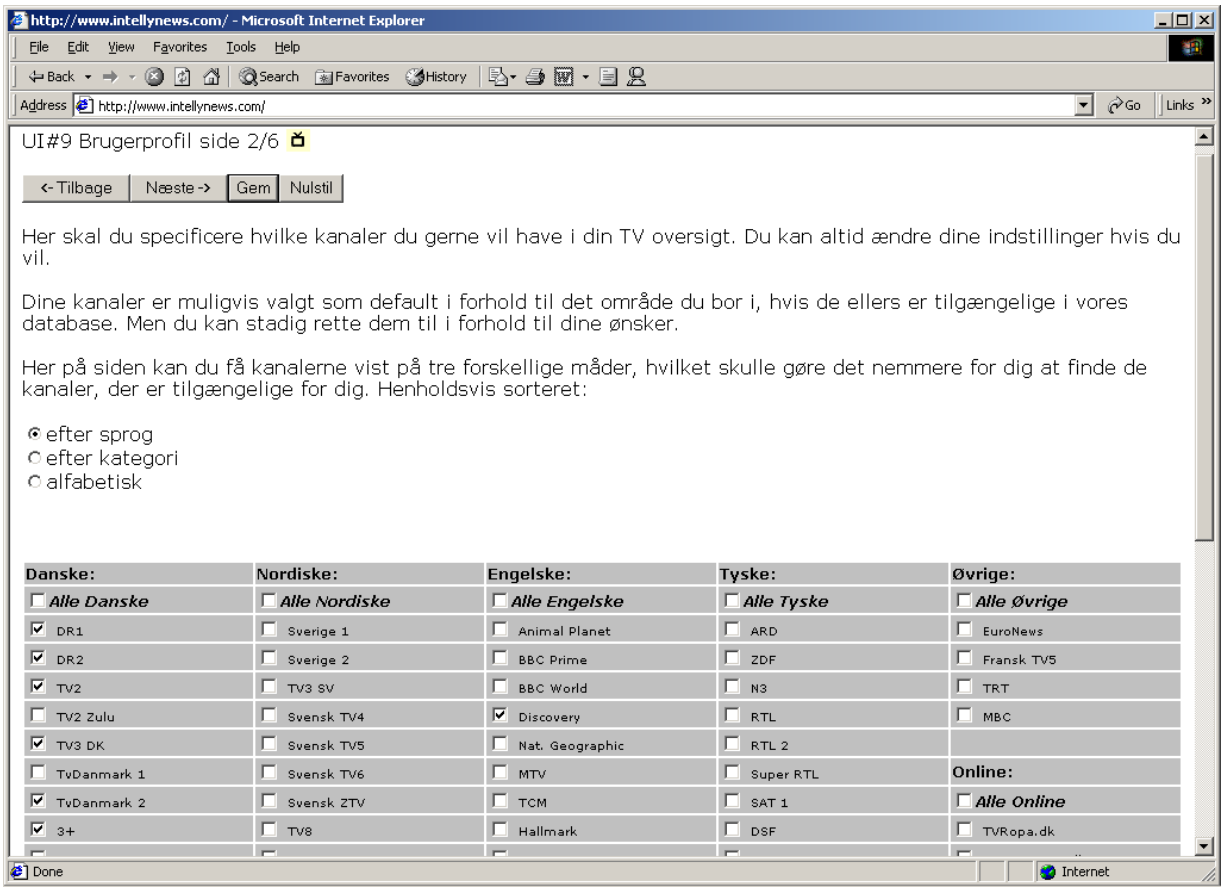

Figure 26 : The user profile page where the channels have to be set.

The next page in the user profile is the categories page (see Figure 27) where the categories that interest the user can be chosen. It is here again possible to choose the categories one by one or all at once.

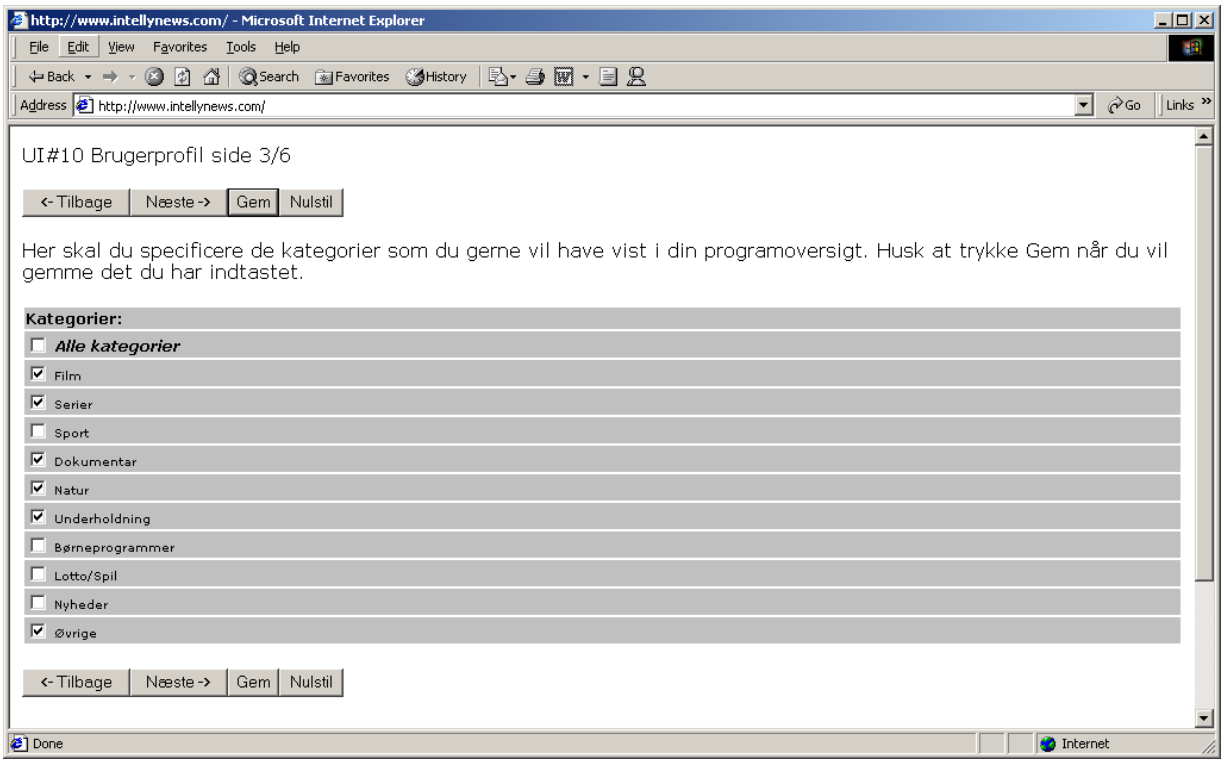

Figure 27 : The categories page in the user profile.

Next is the Repeaters page (see Figure 28) where the user gets the repeating programs presented. The user has here the possibility of selecting the programs that is interesting for him/her and there is also the possibility of deselecting the programs that are of no interest.

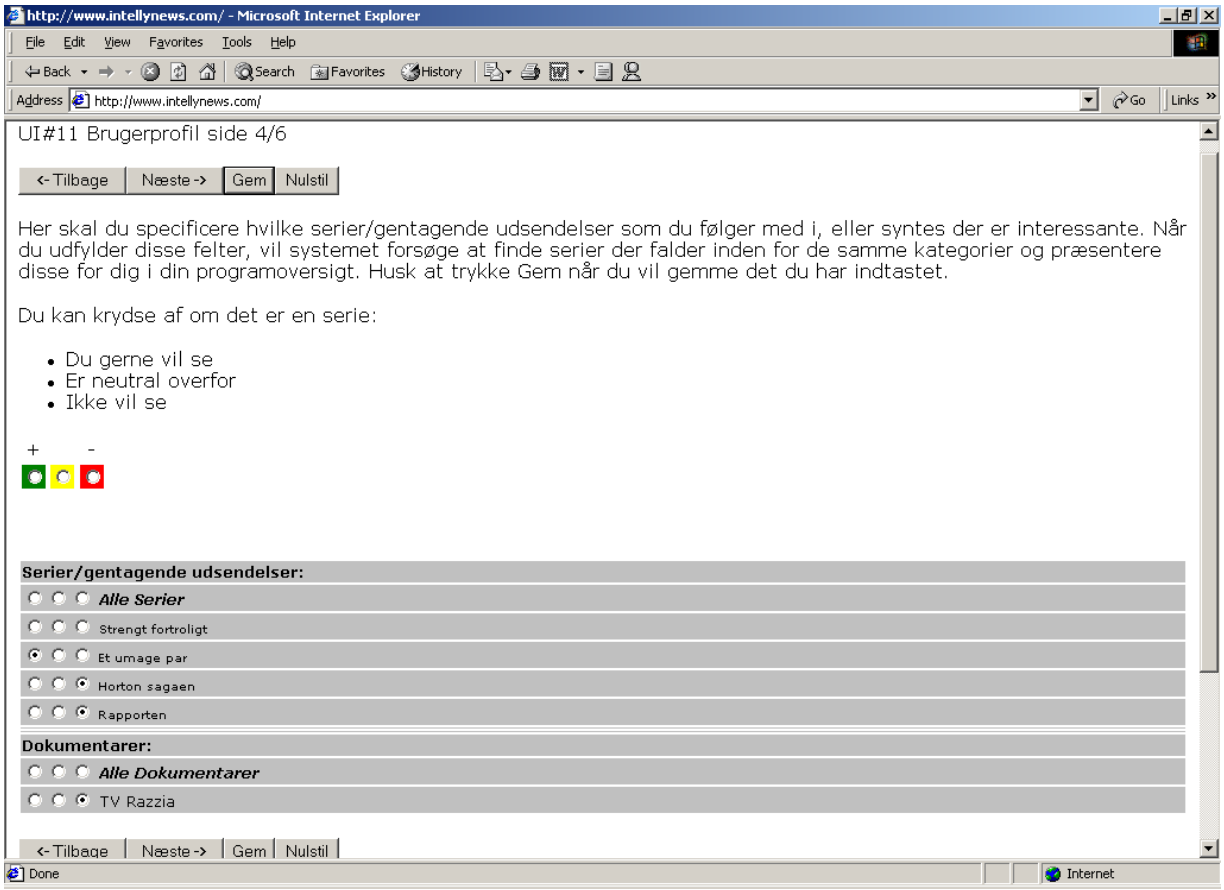

Figure 28 : The Repeaters page where the user can select/deselect programs that he/she wants to see or not.

The fifth page is the keyword page (see Figure 29) and here the user can type in keywords that he/she wishes the system to look for in the title or description of the program.

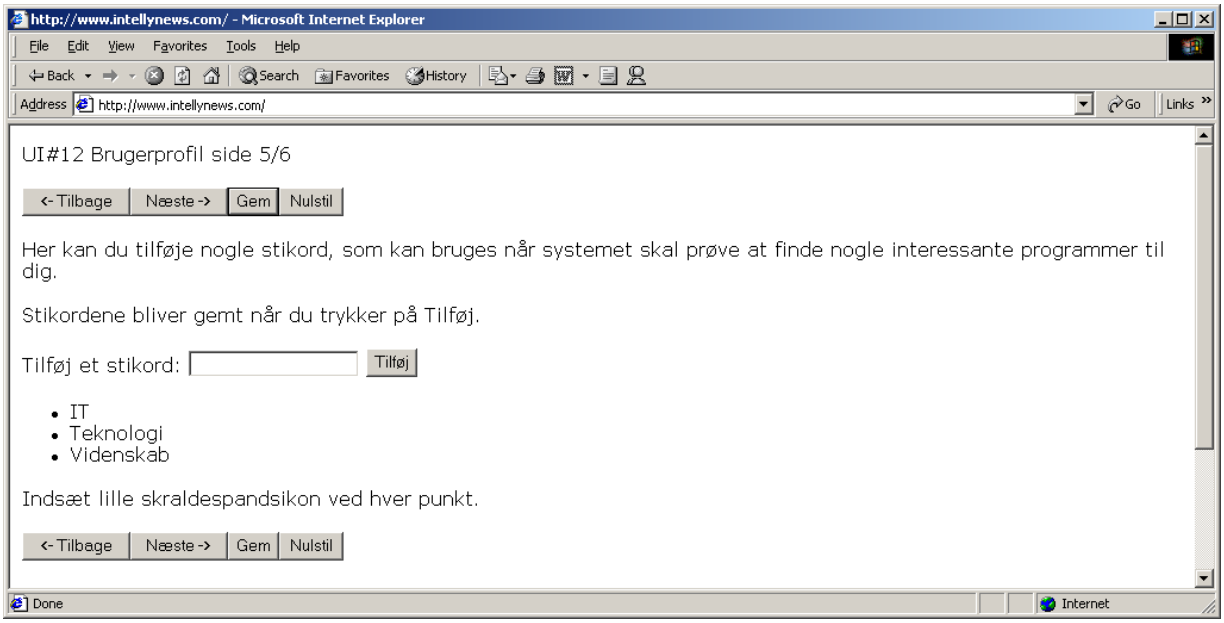

Figure 29 : The keyword page in the user profile.

The last page in the user profile is the set-up page (see Figure 30) where the users can set-up details of how the program list should be presented. It is possible to choose the columns that should be presented, which page of the TV-guide that should be the start page and how the date and time should be presented to the user.

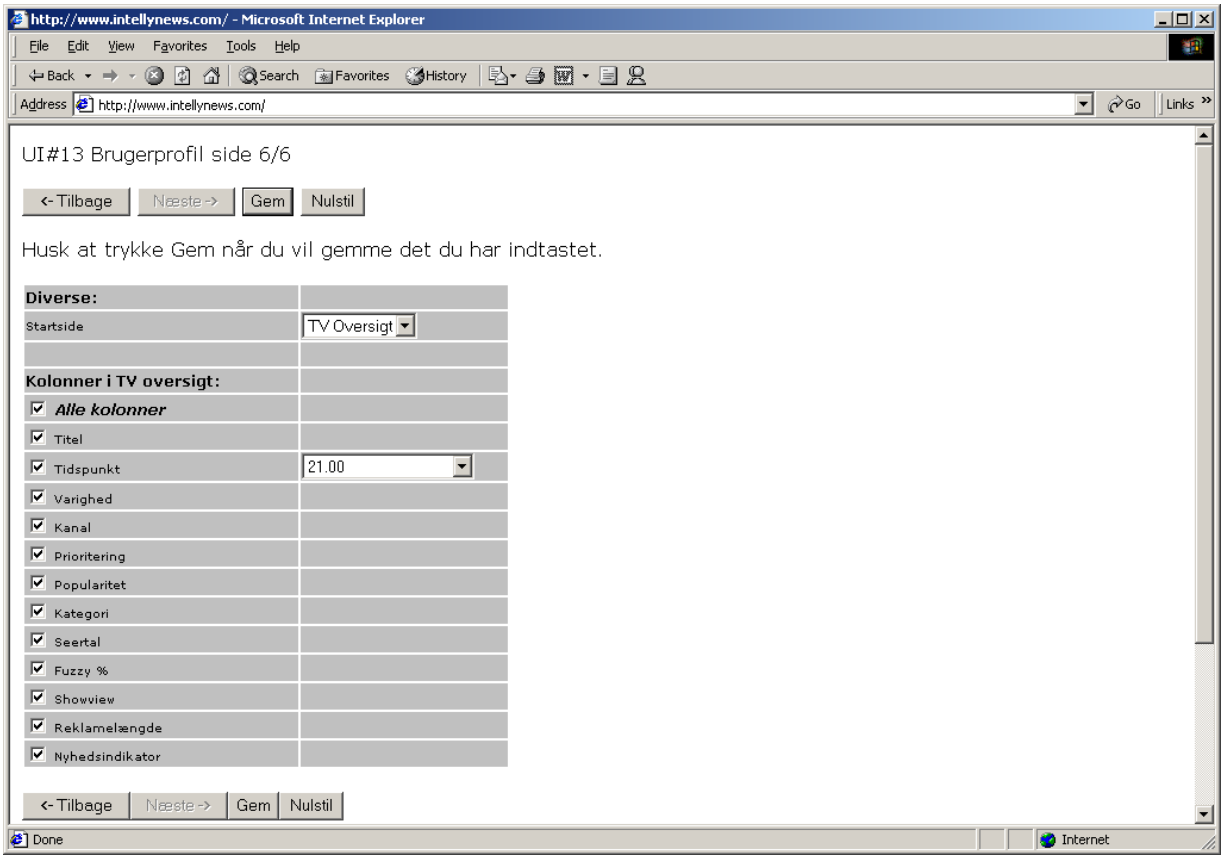

Figure 30 : Set-up page.

Programs List

This is the main page in the TV-guide. The user gets the TV-programs presented at this page. The user has the possibility of filtering the program list by use of the filter presented at the left side of the page. Additional functionality on the program list is the notepad and prioritising functionality. The notepad shall help the user to remember what to see and the prioritising functionality is used to personalise the TV-guide. This is also used to hide the programs that do not interest the user. There is a description of the program available but this varies from mock-up to mock-up and will be describe in connection with the description of the single mock-ups. At last there is a search feature that make it possible to search for programs.

Below there is presented different mock-ups of the program list. The first mock-up presented (see Figure 31) is where there is a short description available for all programs and where the user can prioritise the programs by the use of three radio buttons. The hidden programs are presented below the dotted line. These are hidden if the hidden select box (Danish: "Vis skjulte") is selected. A colouring of the radio buttons is tried out.

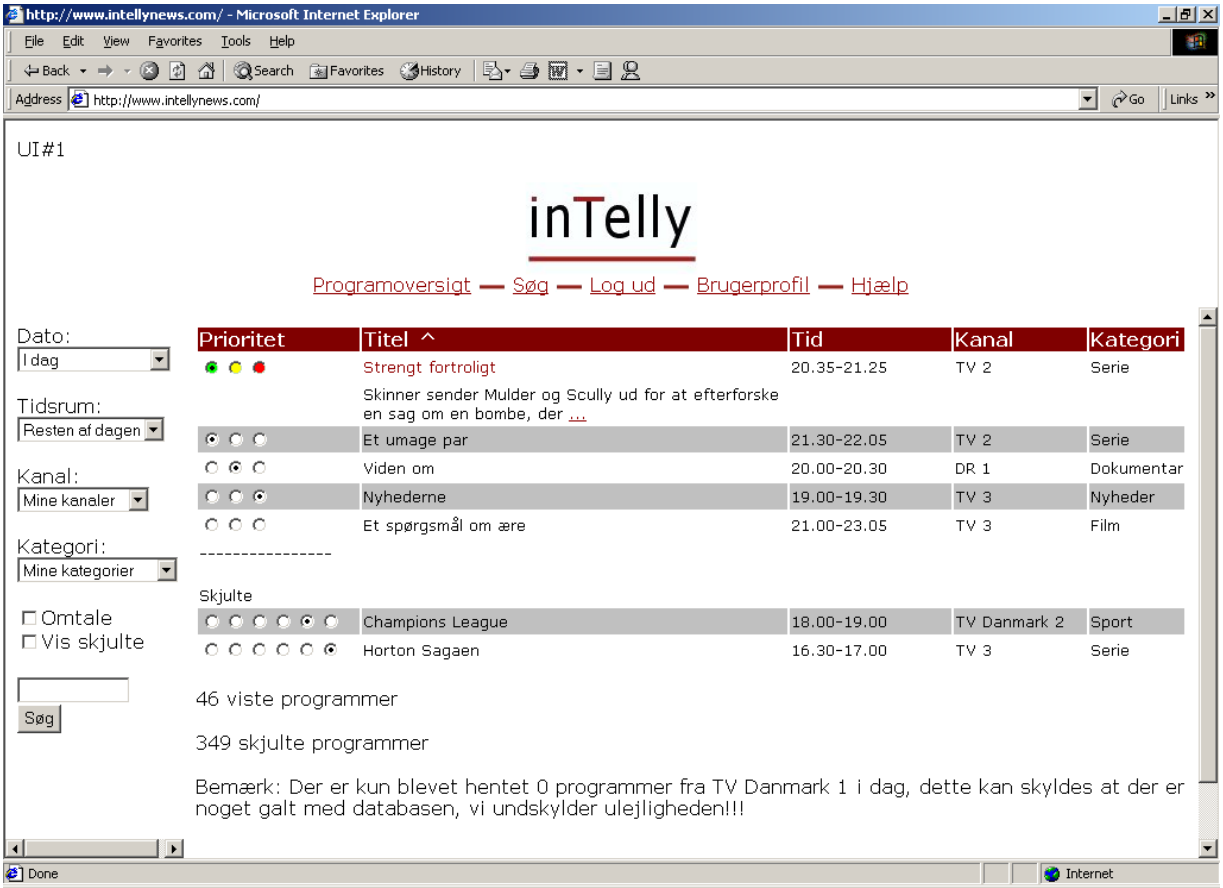

Figure 31 : A mock-up of the possible program list.

The next mock-up (see Figure 32) is almost the same as the interface presented above. The difference is the way of prioritising programs and the description. The way of prioritising has changed from the three radio buttons to two icons, where "√" indicates Want to see and "X" indicates Does not want to see. The programs that are evaluated "Want to see" are placed on a

separate notepad page (see Figure 34). At the right side of the program list there has now been added an area where the detailed description for a program is presented and the text presented is decided by the position of the mouse pointer.

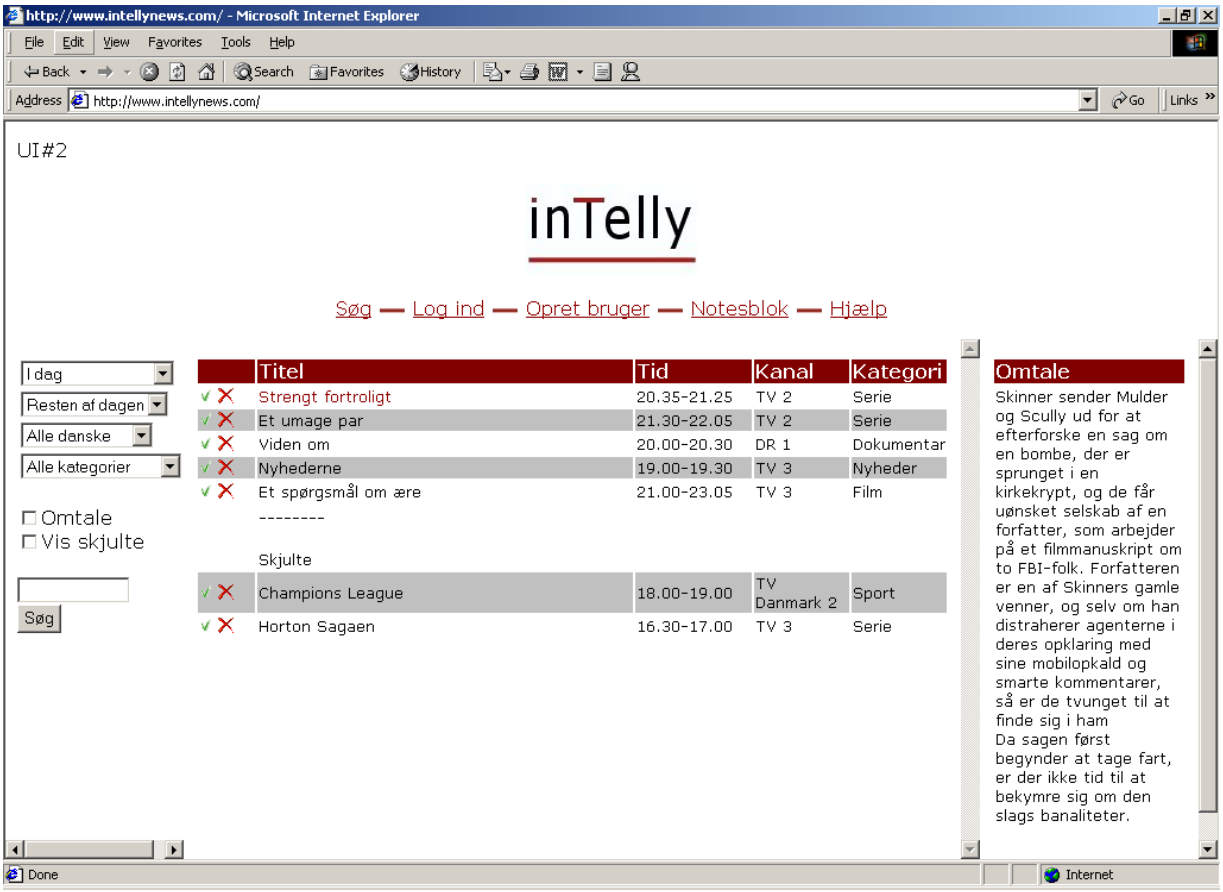

Figure 32 : Another program list mock-up with integrated detailed description.

Next is a mock-up that is very identical to the one above (see Figure 33). The difference in this mock-up is that the integrated notepad now has moved to the frame where the detailed description was earlier. The detailed description can be seen by clicking on the title of the program.

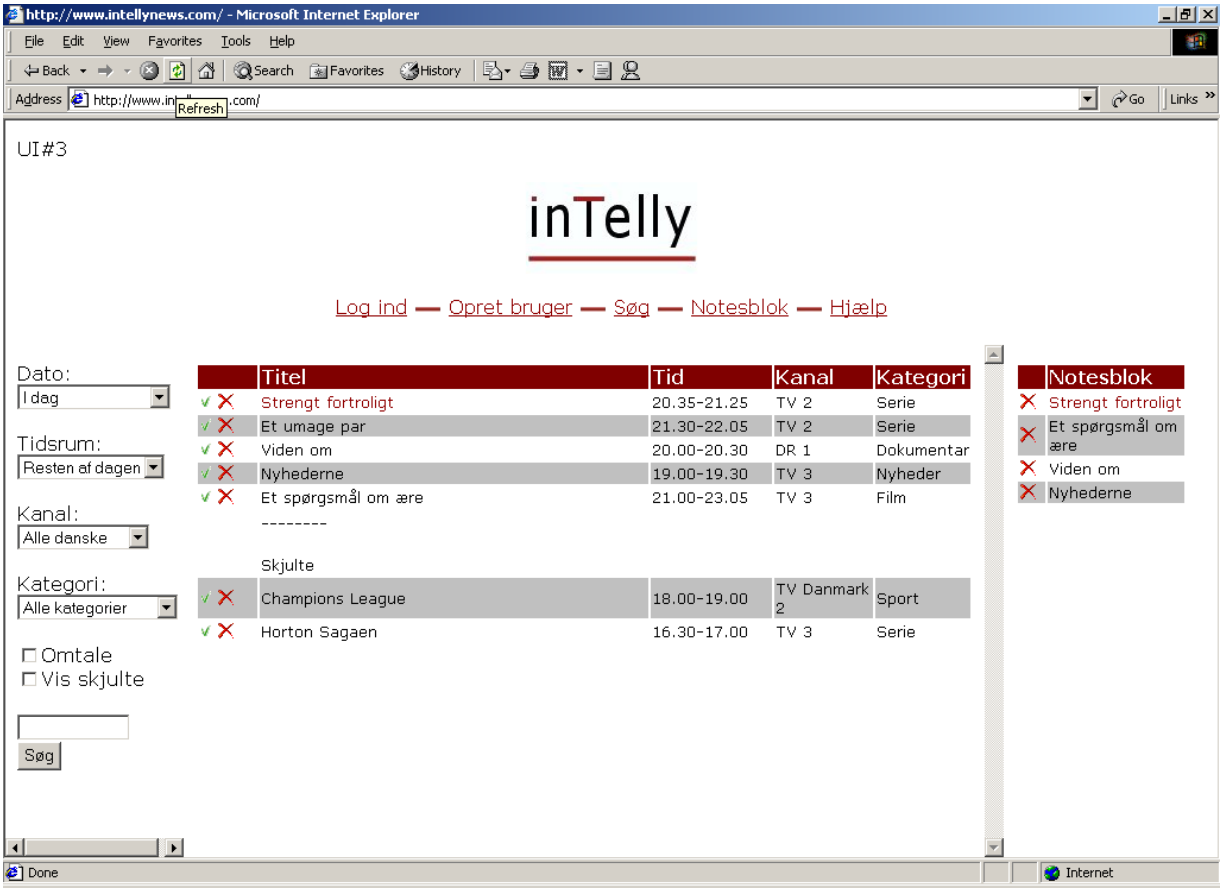

Figure 33 : Program list mock-up with integrated notepad.

Notepad

Two proposals for the notepad interface are presented in Figure 34. This page contains the programs that have been evaluated "Want to see" by the user. On the first proposal it is possible to remove the selected program again by pressing "X" or the user has the possibility of choosing between see the program (the eye icon) or recording it (the red dot, as on an VCR that is recording). The second possibility has also the "X" that removes programs from the list again. This one also has two arrows so that the user can move around with the programs and place them in the three categories: Want to see, Maybe want to see and Record.

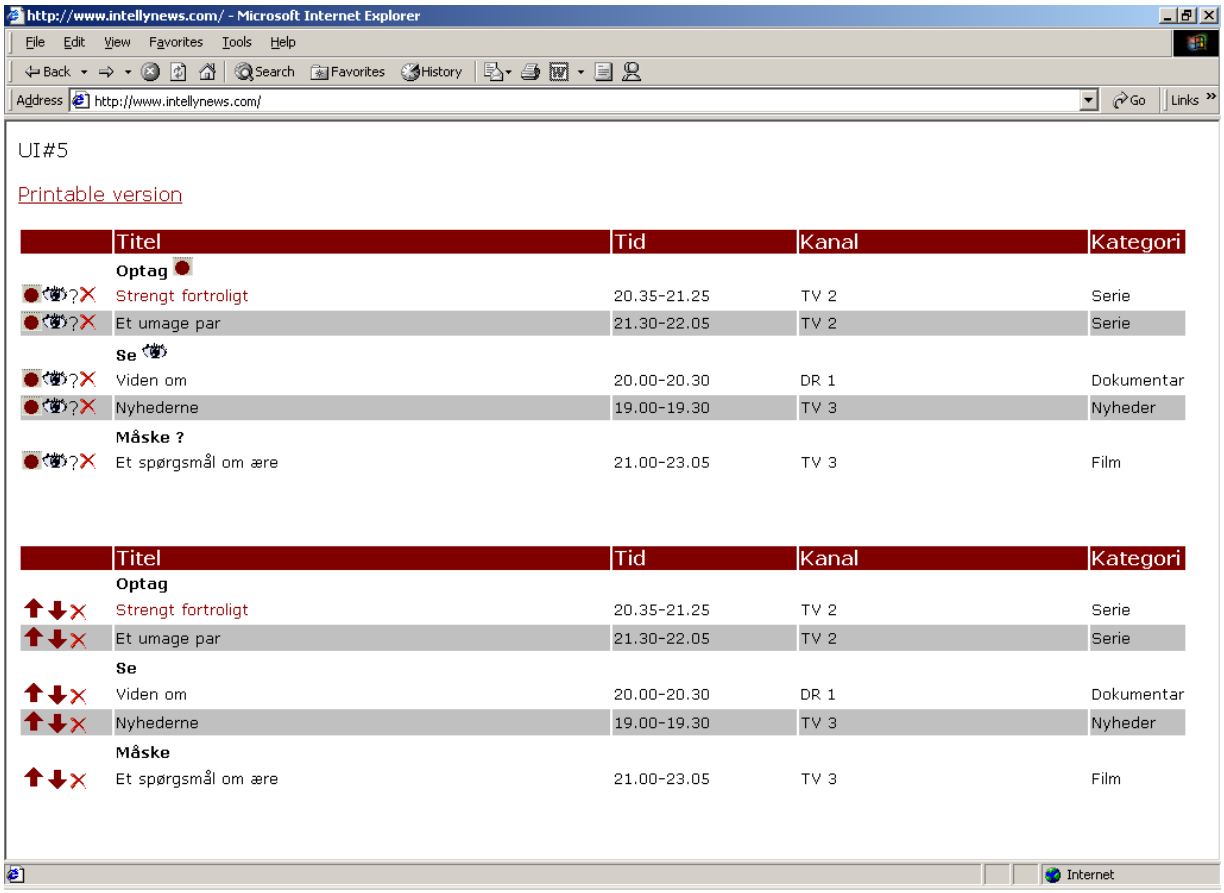

Figure 34 : The notepad for the program list.

Detailed Description

This proposal of the detailed description contains a text describing the program showed, program data like channel, start time etc., photo material relating to the program, news concerning the programs, websites about the program and newsgroups where the single program is discussed (see Figure 35).

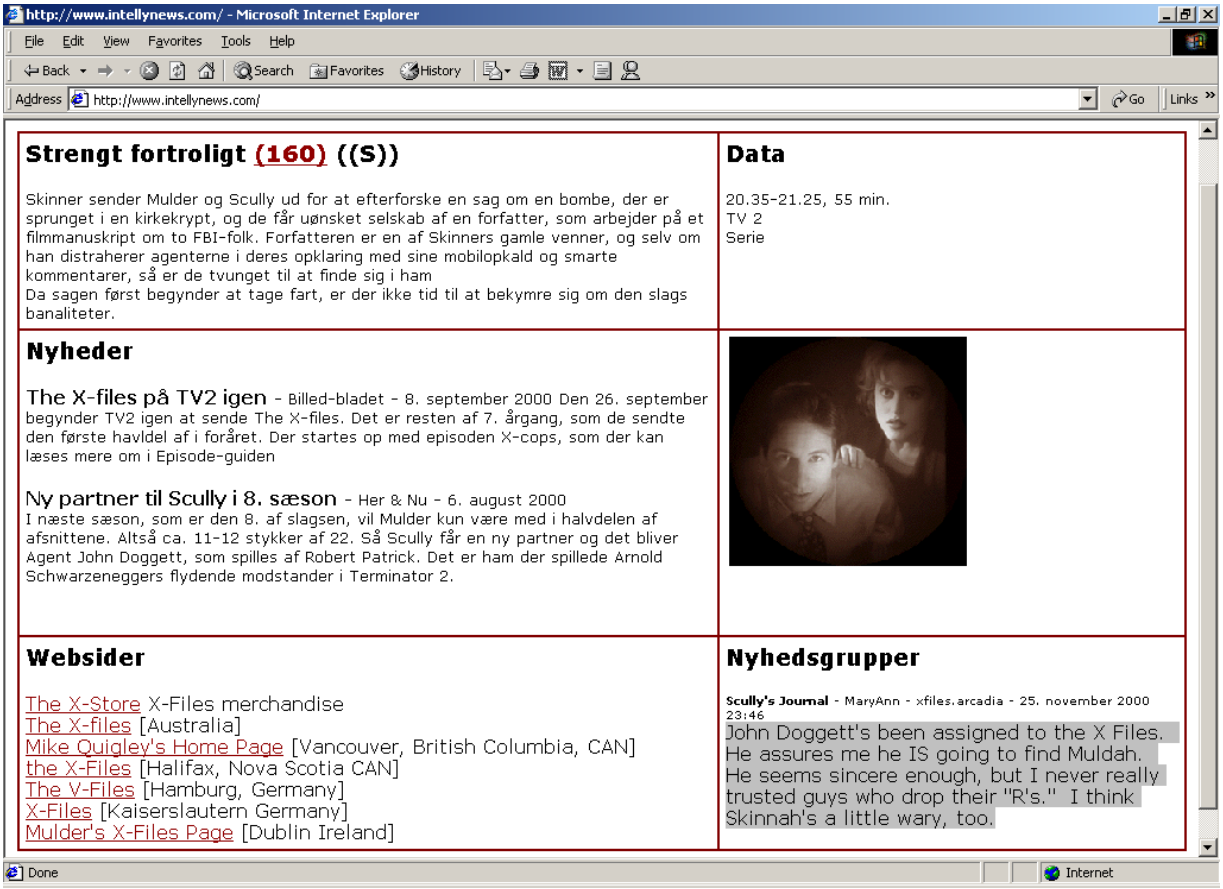

Figure 35 : The detailed description of a program.

Help

The help is an important part of the system and that is why there has already been made the first mock-ups. The first mock-up presented in Figure 36 shows a possible way of describing the way of prioritising the programs.

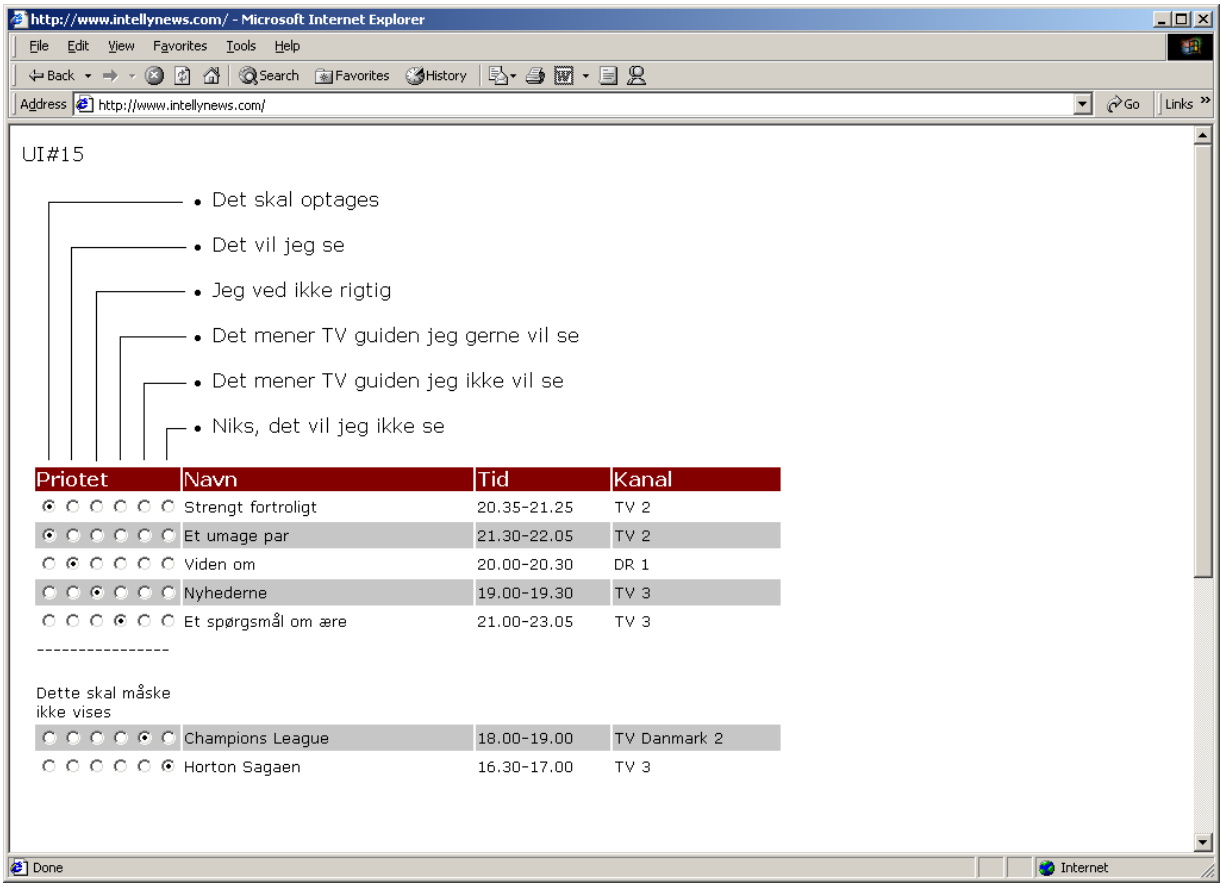

Figure 36 : Help mock-up for the use of the priority.

The next is the help for the program list (see Figure 37).

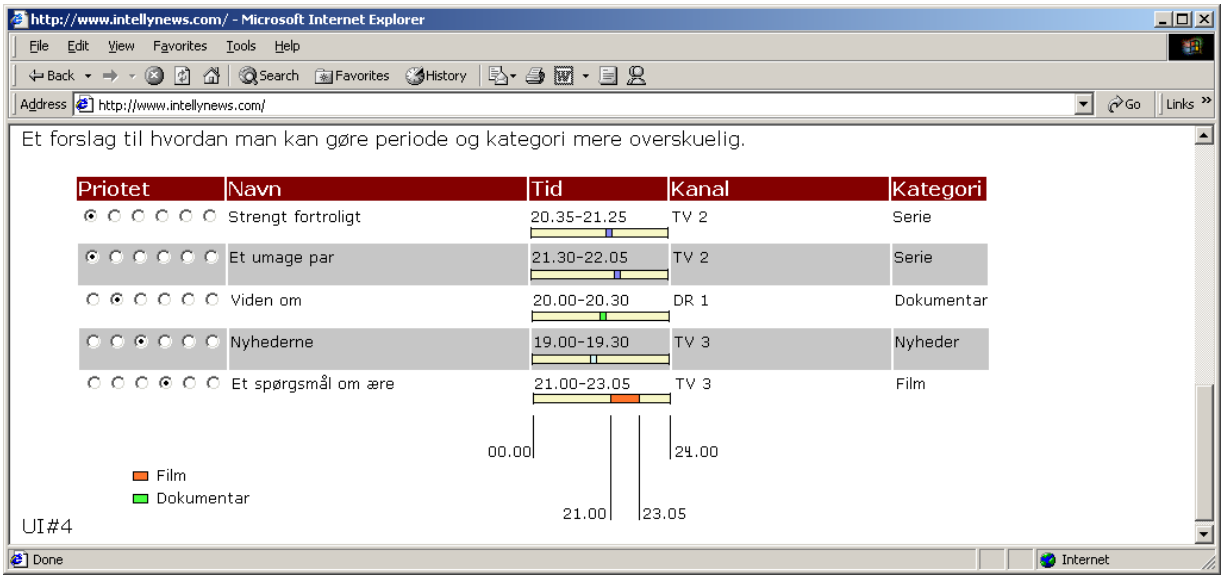

Figure 37 : Mock-up of the program list help.

Menu Bar and Logo

The menu bar and logo is seen in Figure 38 and this is an important part because it shall help the user not to feel trapped. The menu bar consists of links to the:

- Program list
- Search
- Logout
- User profile
- Help

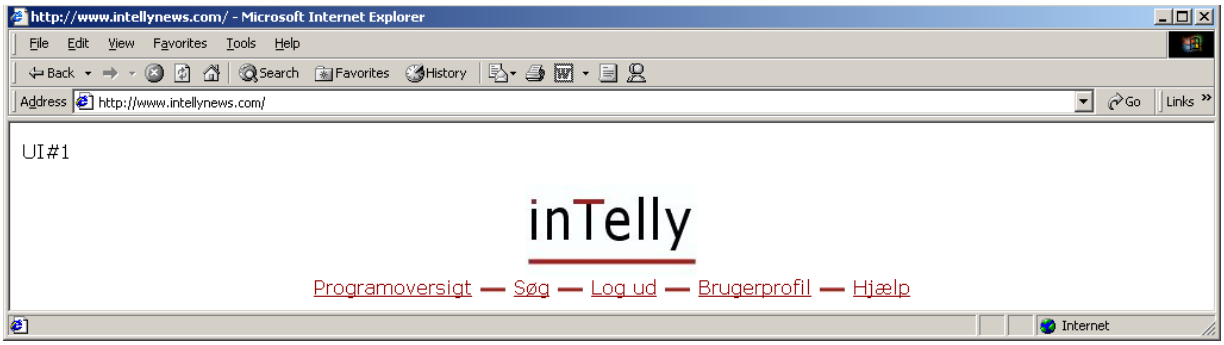

Figure 38 : The logo and menu bar of the inTelly system.

Messages

The messages are thought as feedback to and from the user. The messages shall inform the users about important things like programs that are about to start or maybe that a Repeater has been moved (see Figure 39). The messages can also be used to get feedback from the user by letting him/her prioritise the program presented in the message. The little “m” shown on the figure below is a button that redisplays the last message.

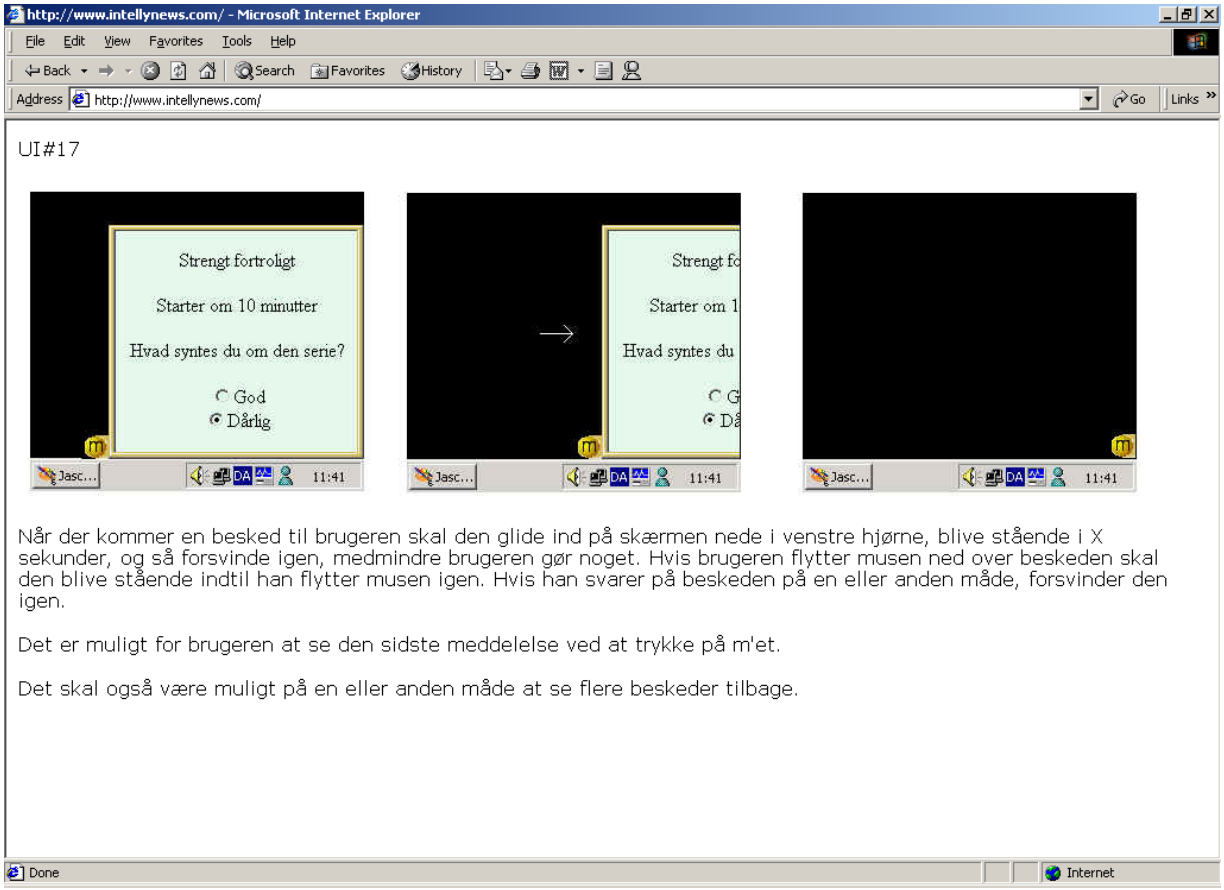

Figure 39 : A mock-up of a possible message telling the user that a program is about to begin.

N. User Interface Version 0.2

This supplement will present the user interfaces that where designed for version 0.2 of the inTelly system. The arguments for the design are presented in the *Main report – 17. inTelly version 0.2*.

Create User

The create user page used in the version 0.2 is seen in Figure 40. There are no differences in the way this page works from the one presented in version 0.1. There has here been added some text describing how to create the user profile and what characters to use and not. There has also been added an "Afbryd" button, which gives the user the possibility of stopping the creation of the user profile.

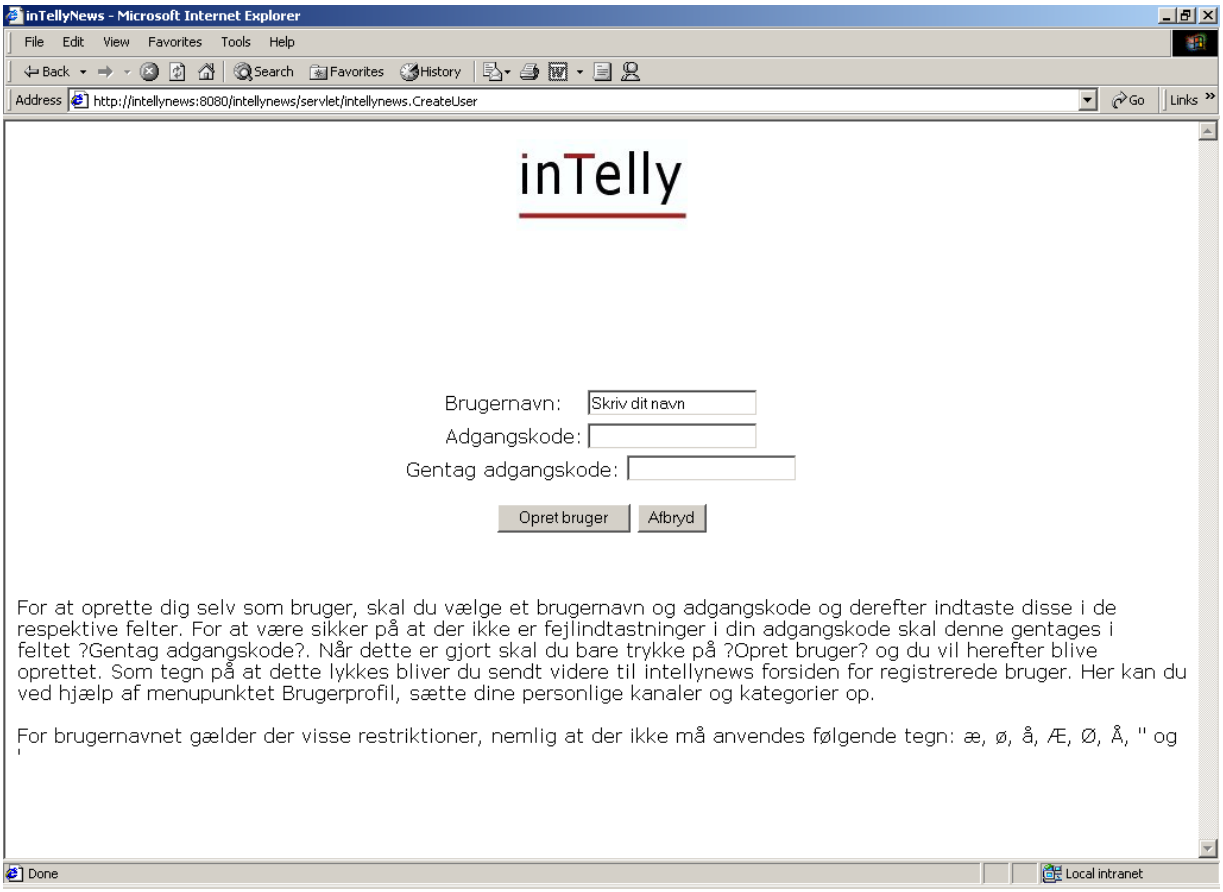

Figure 40 : The version 0.2 of the create user interface.

Login

In the version 0.1 there where no login page but this has changed in the version 0.2. The login page can be seen in Figure 41. At this page the user has to type in the username and password and then press the "Log in" button to get into the system. If he/she regrets there is also the possibility of using the "Afbryd" button that sends the user to the front page for non-registered users.

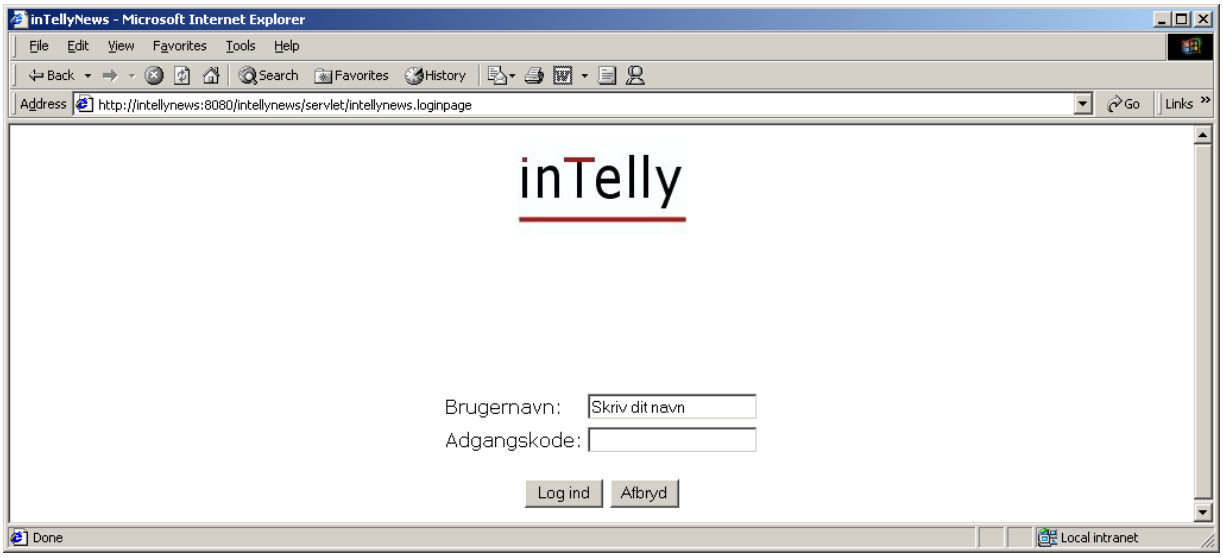

Figure 41 : Login page for inTelly.dk version 0.2.

User Profile

The version 0.2 also contains a user profile. This version only contains a front page, a channels page and a category page. The front page is seen on Figure 42 and contains as earlier a description of what the user profile is used for. It also consists of a menu at the left side of the page. This menu is present at all pages of the user profile to make navigation easier for the user. The menu also contains to buttons for saving and retrieving the user profile from the database.

Figure 42 : User profile front page.

The next page presents the channels page of the user profile (see Figure 43). Here the user has the possibility of choosing the channels that are of interest. There are here the same possibilities of selecting channels as in version 0.1.

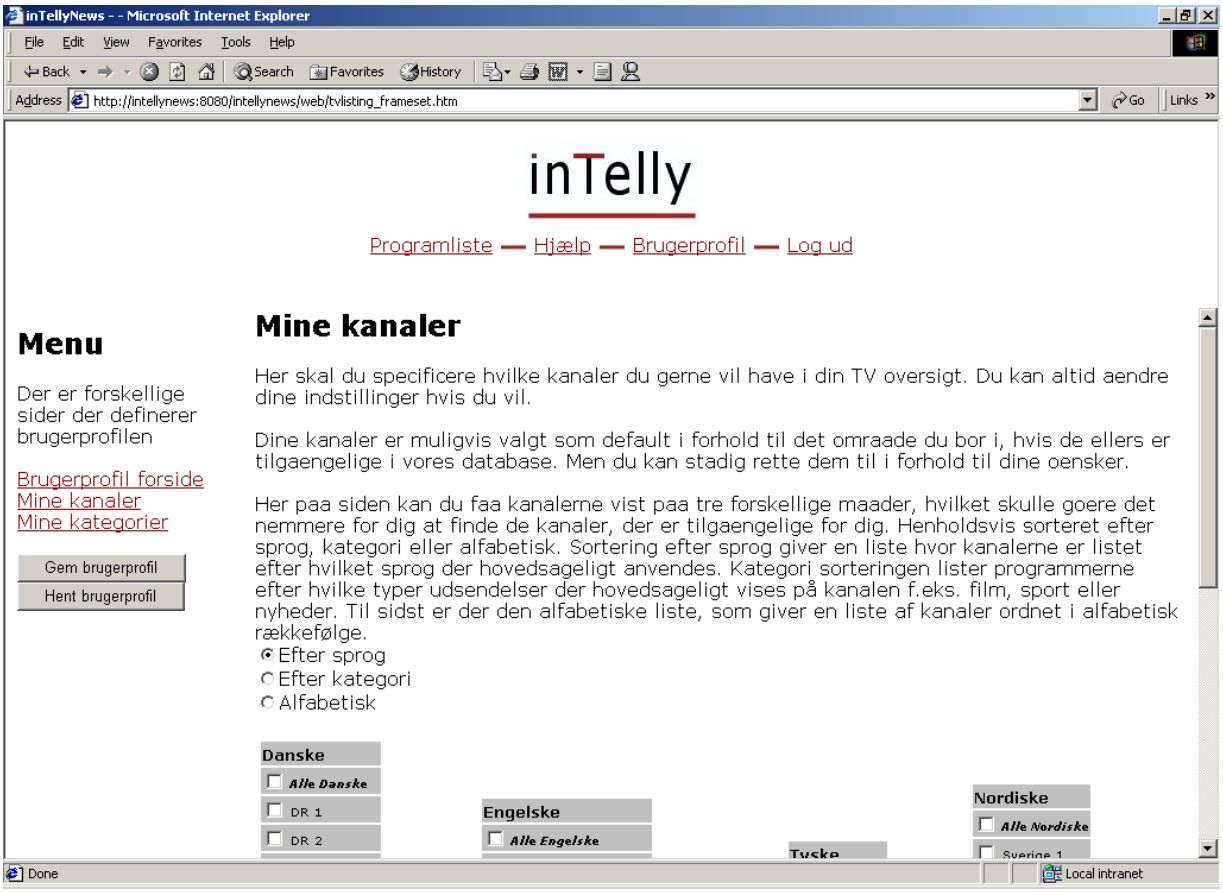

Figure 43 : The channels selection page of the user profile.

The last page of the user profile in this version 0.2 is the categories page and it is presented on Figure 44. There are the same features as on the version 0.1 page.

Figure 44 : The categories page in the user profile.

Programs List

There have been a number of changes to the program list from the version 0.1 to version 0.2. First of all there has been created a user interface for non-registered users of the system (see Figure 45). This page has all the facilities of a normal TV-guide like the filter and so on. An extra feature is the possibility of sorting the single columns ascending or descending.

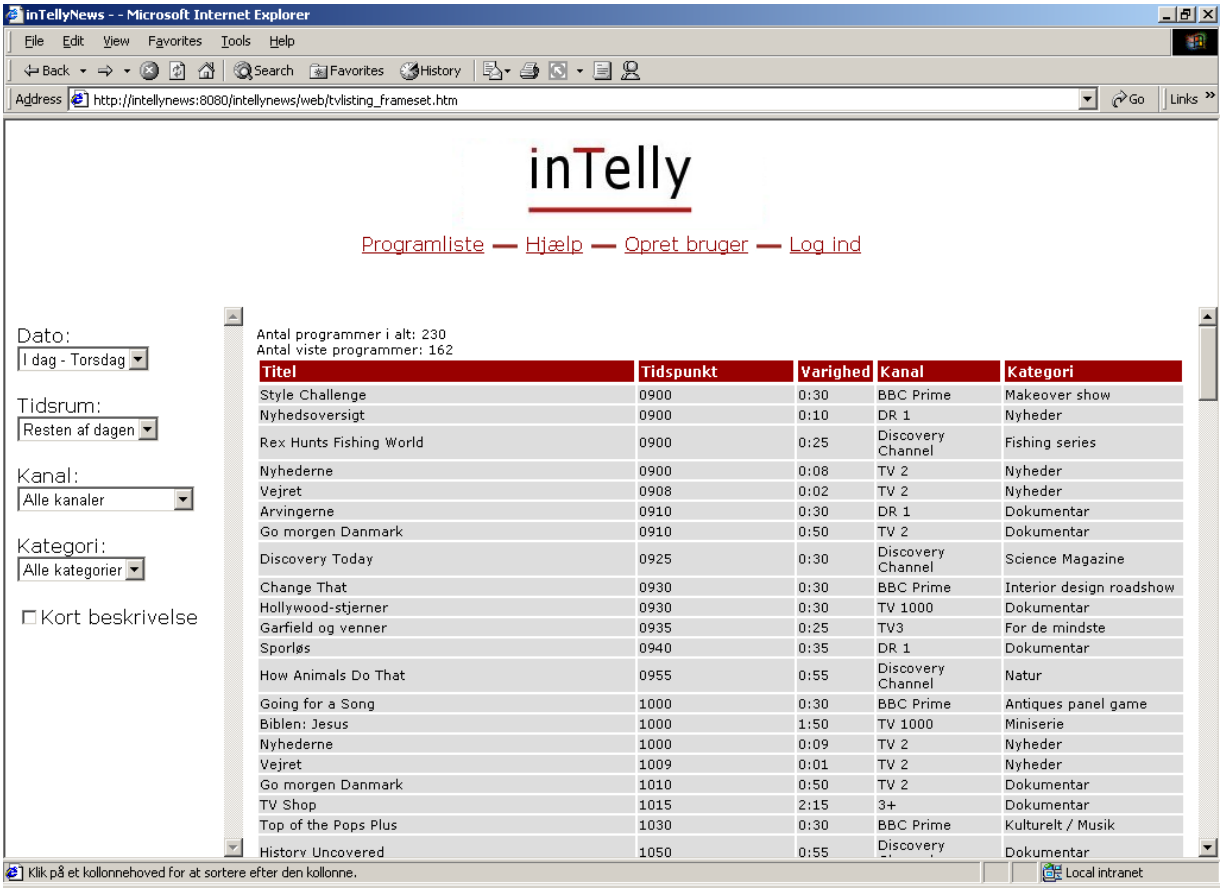

Figure 45 : The user interface for non-registered users.

On Figure 46,Figure 47, Figure 48 the new user interface for registered user can be seen. The functionality of the interface has only changed a little from the earlier version. The program list interface consists of the filter menu at the left side of the screen and the table of programs. In this version it has been chosen to use the radio buttons for prioritising and adding programs to the notepad, which in this version is integrated in the table of programs. The colour indication on the page shows how the user has prioritised the programs and green indicates, "Want to see", yellow is "Maybe want to see" and red is "Do not want to see". The grey and dark grey colours are the systems indication of "Want to see" and "Do not want to see". This user interface also provides the possibility of sorting the single columns ascending or descending. The programs with the colour red and dark grey are hidden if the check "Vis skjulte" at the left side is checked.

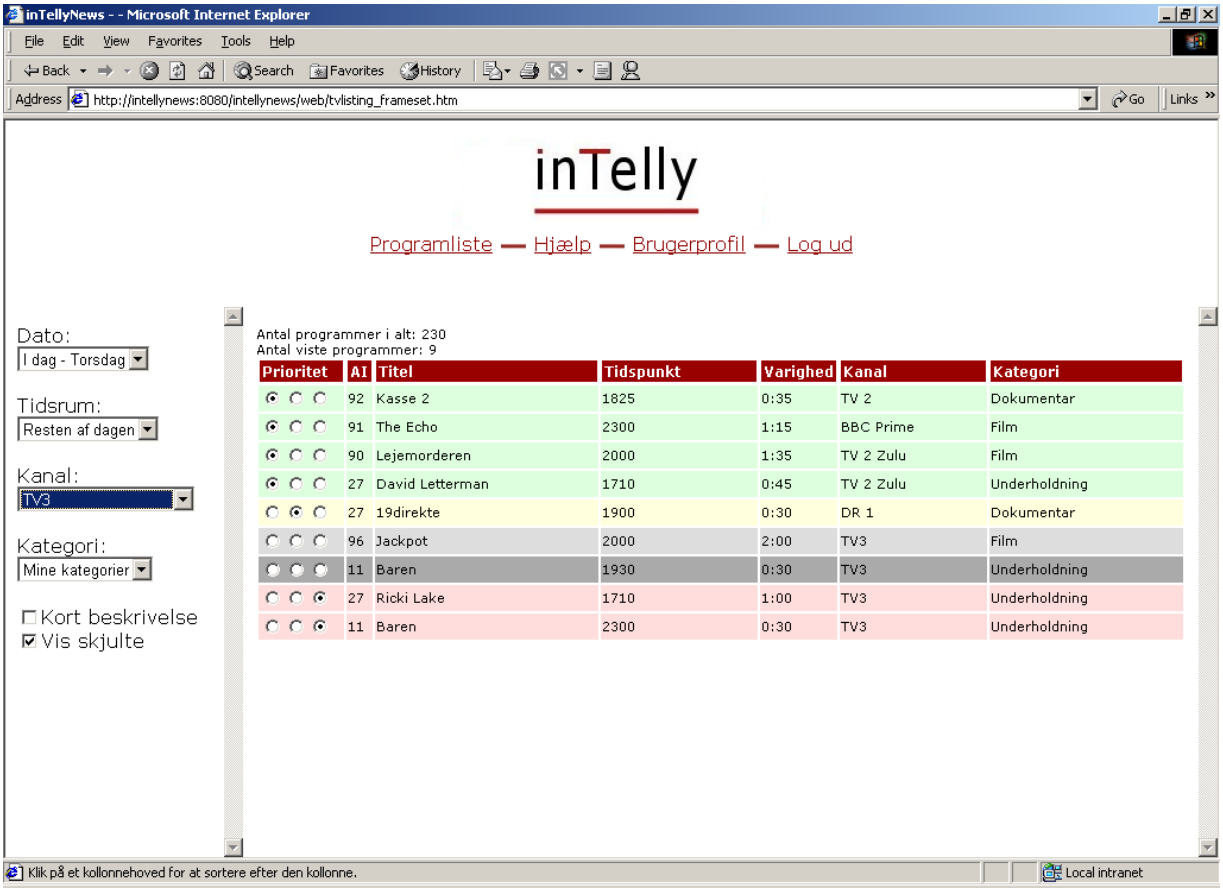

Figure 46 : The program list user interface.

Figure 47 shows what happens when the short description check box is checked. In that case there will be presented a short description for each of the programs in the program list.

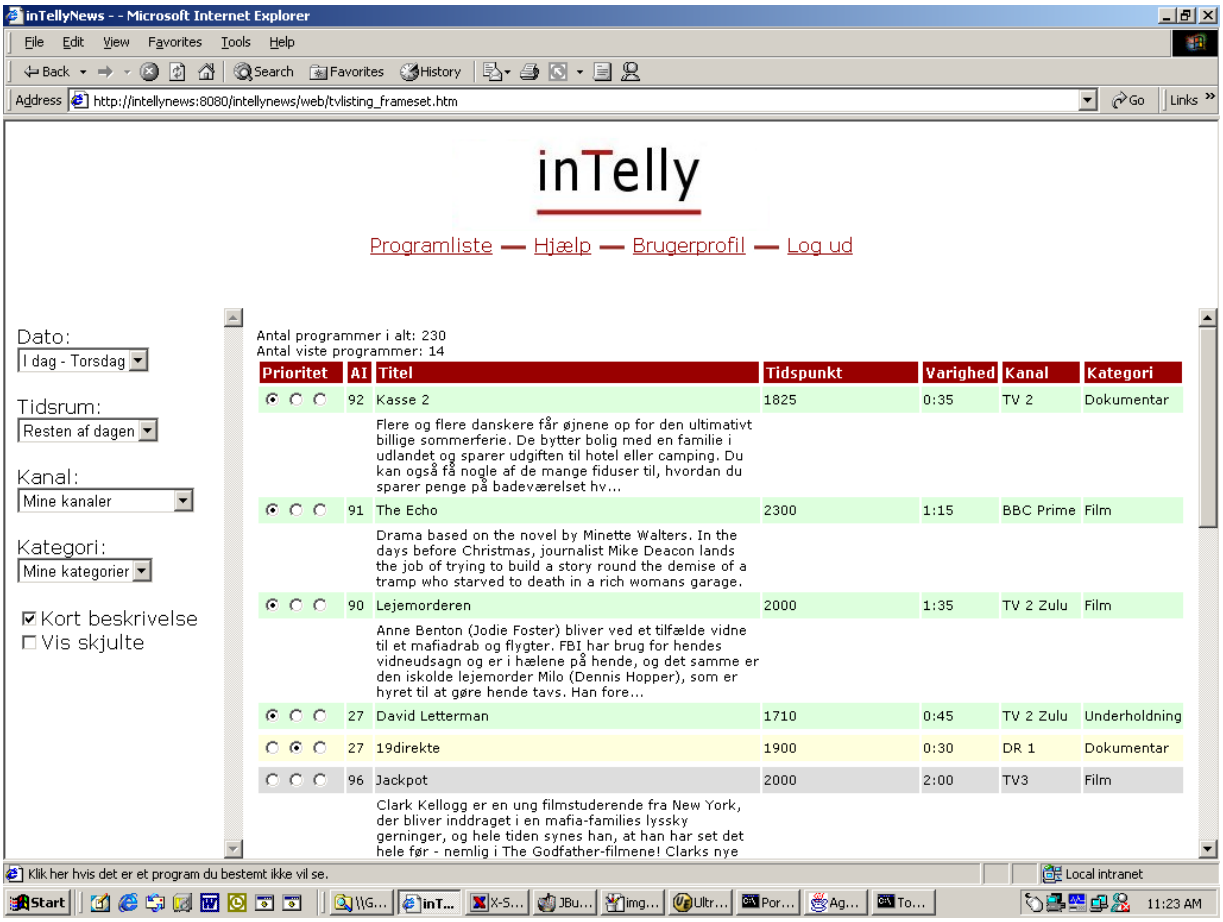

Figure 47 : The program list with a short description for each of the programs.

The next shown is the program list with a built in pop-up functionality for the description (see Figure 48). Besides the short description shown in Figure 47 there has also been created a pop-up functionality that shows a description of the program when the mouse pointer is pointing at the title of the program.

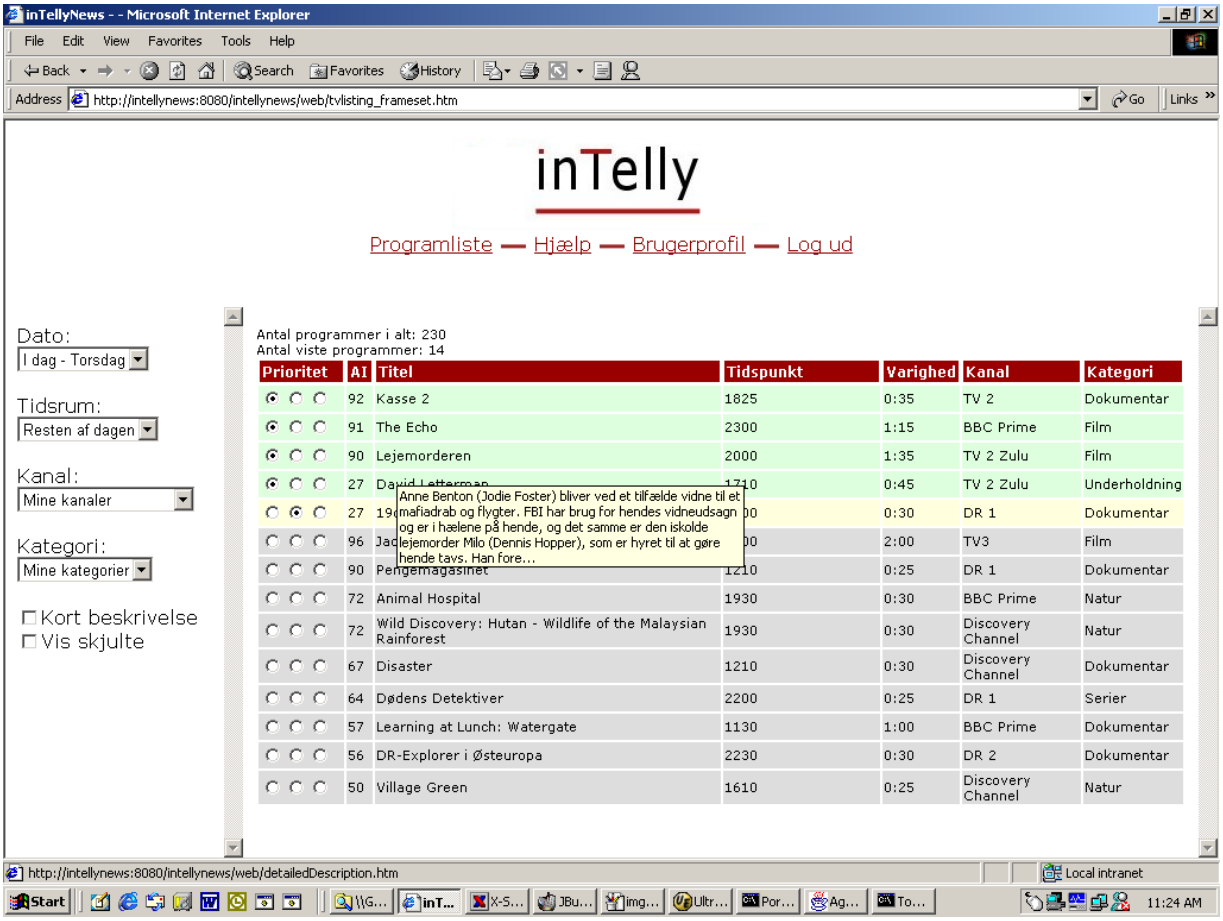

Figure 48 : A screendump of the program list where the pop-up description is showed.

Detailed Description

The detailed description has not changed since version 0.1 (see Figure 49).

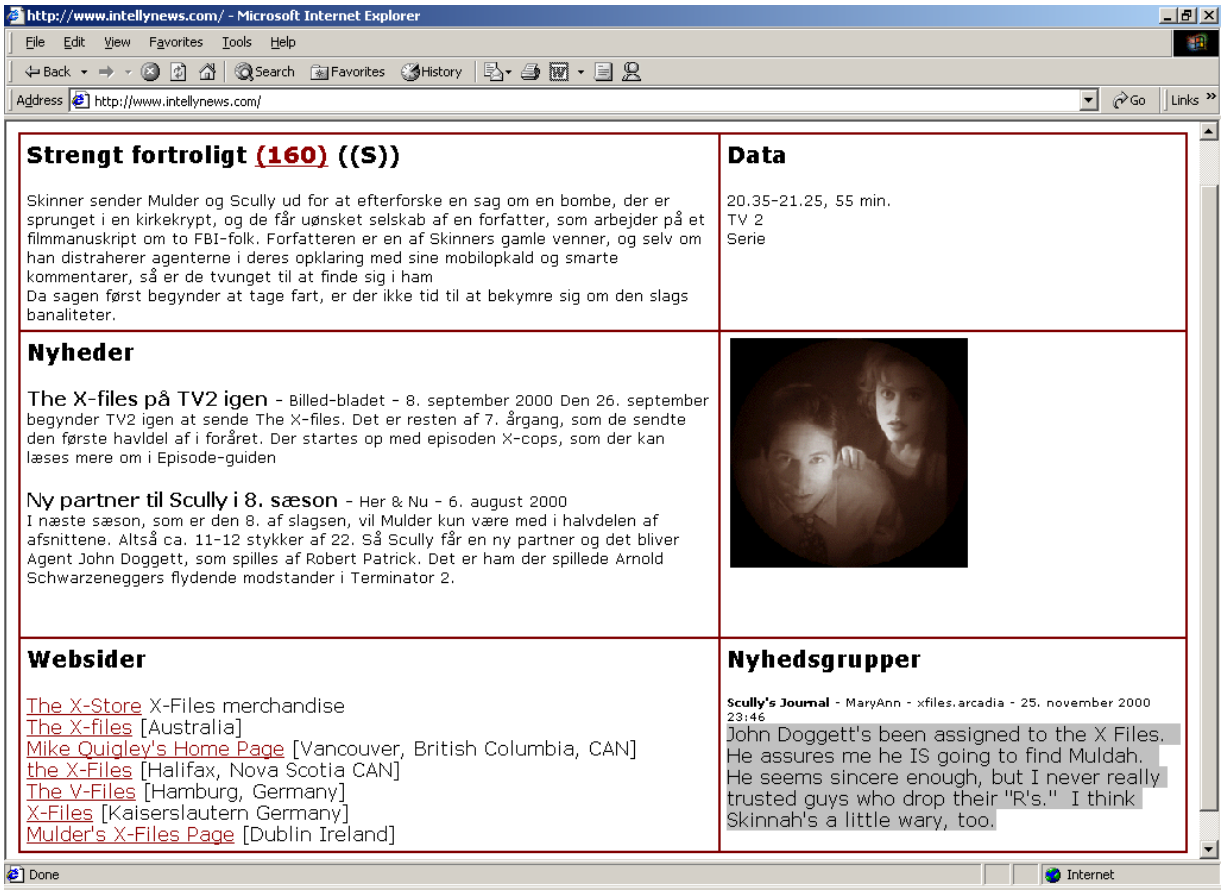

Figure 49 : The detailed description.

Help

The help in version 0.2 has been change so that it is similar to the user profile. It consists of a front page and a number of pages that describes the main pages in the system. As well as in the user profile there has been created a menu at the left side of the page. The front page (see Figure 50) gives an introduction to the help. On the help pages there is placed a navigation button that sends the user to the next page within the help.

Figure 50 : The help front page.

The second page in the help is an introduction the to TV-guide itself. The page on Figure 51 tells the user what possibilities there are at this TV-guide and how it differs from the other normal TV-guides.

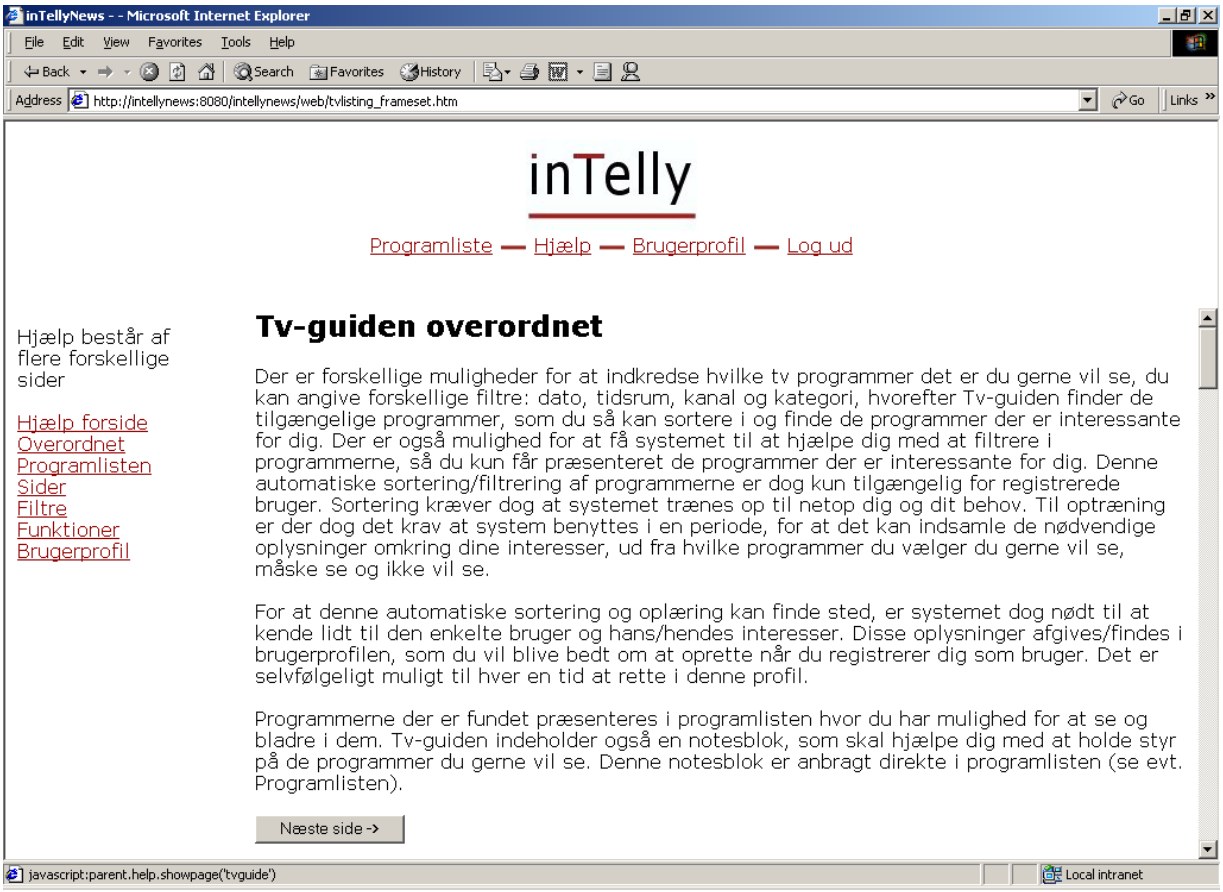

Figure 51 : Introduction to the TV-guide itself.

Figure 52 presents the help for the program list. It describes the notepad function and the prioritising of programs.

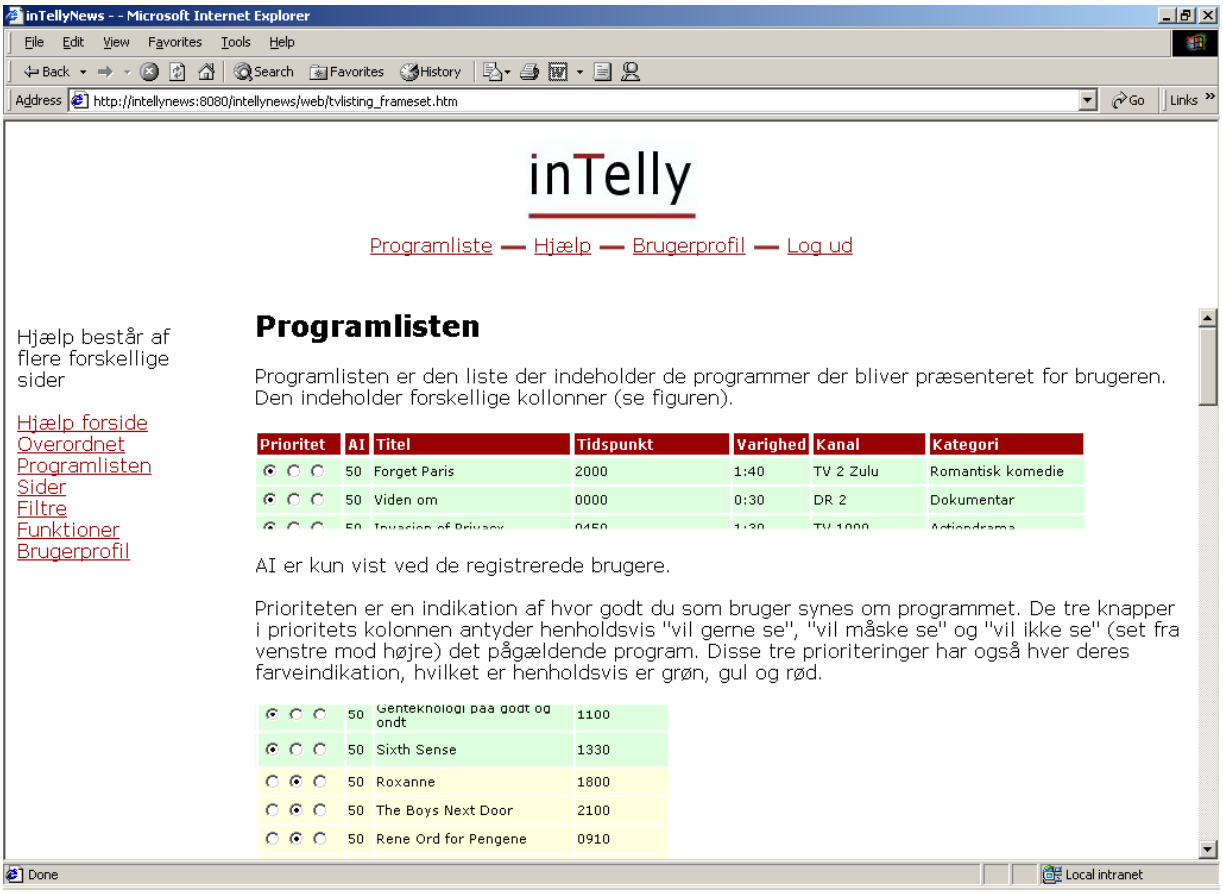

Figure 52 : The help for the program list.

Next is a presentation of the different pages in the inTelly system and the help page for this can be seen on Figure 53.

Figure 53 : The presentation of the different pages in the system.

There is also a page containing help for use of the filters (see Figure 54).

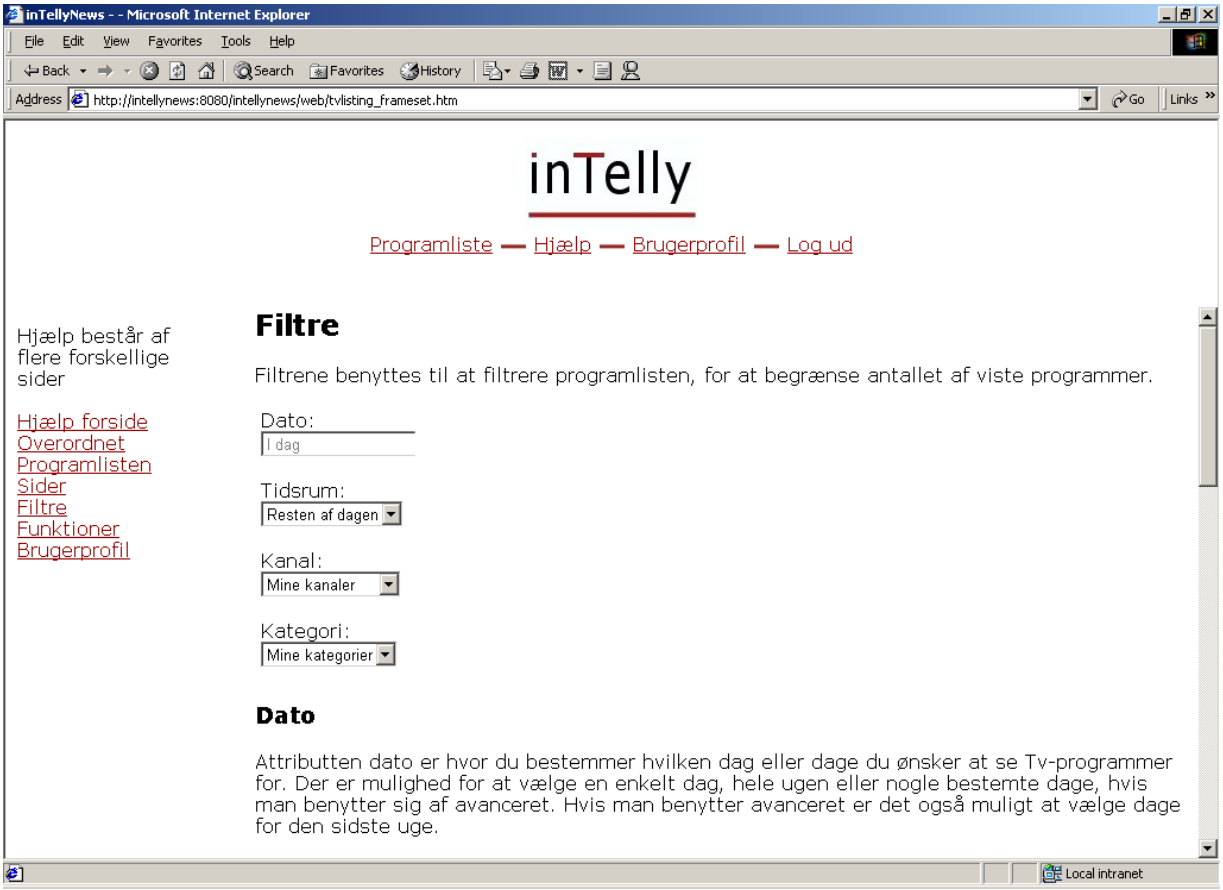

Figure 54 : The filters help page.

The additional functionality like show hidden (Danish: “Vis skjulte”) and short description (Danish: “Kort beskrivelse”) is also described. This is done on the functions page (Danish: “Funktioner”) shown in Figure 55.

Figure 55 : The page that describes the different functions in the system.

The last page in the help is the help for the user profile (see Figure 56). Here it is described what the user profile is used for and how inTelly.dk treats user data etc.

Figure 56 : The help page for the user profile.

Menu Bar and Logo

The only change from Version 0.1 is that the search link (Danish: “søg”) has been removed as seen on Figure 57. The rest is like in version 0.1 this also goes for the functionality of the different links.

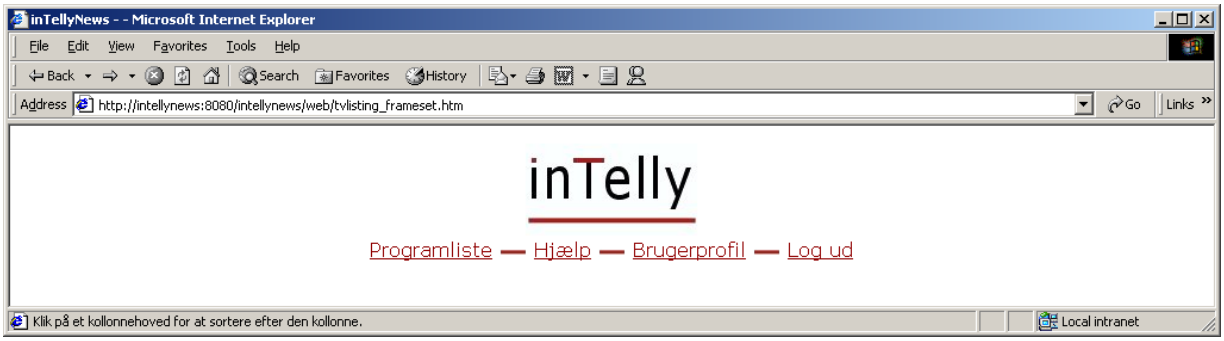

Figure 57 : The version 0.2 logo and menu bar.

O. User Interface Version 0.3

This supplement will present version 0.3 of the user interfaces. This design has to be seen as a somewhat finished design. The arguments for the design are placed in the *Main report – 18. inTelly version 0.3*.

Create User

The only change to the create user interface since version 0.2 is the messages that has been added (see Figure 58). This message slides in from the right top corner and gives the user an indication of what to do to become a registered user.

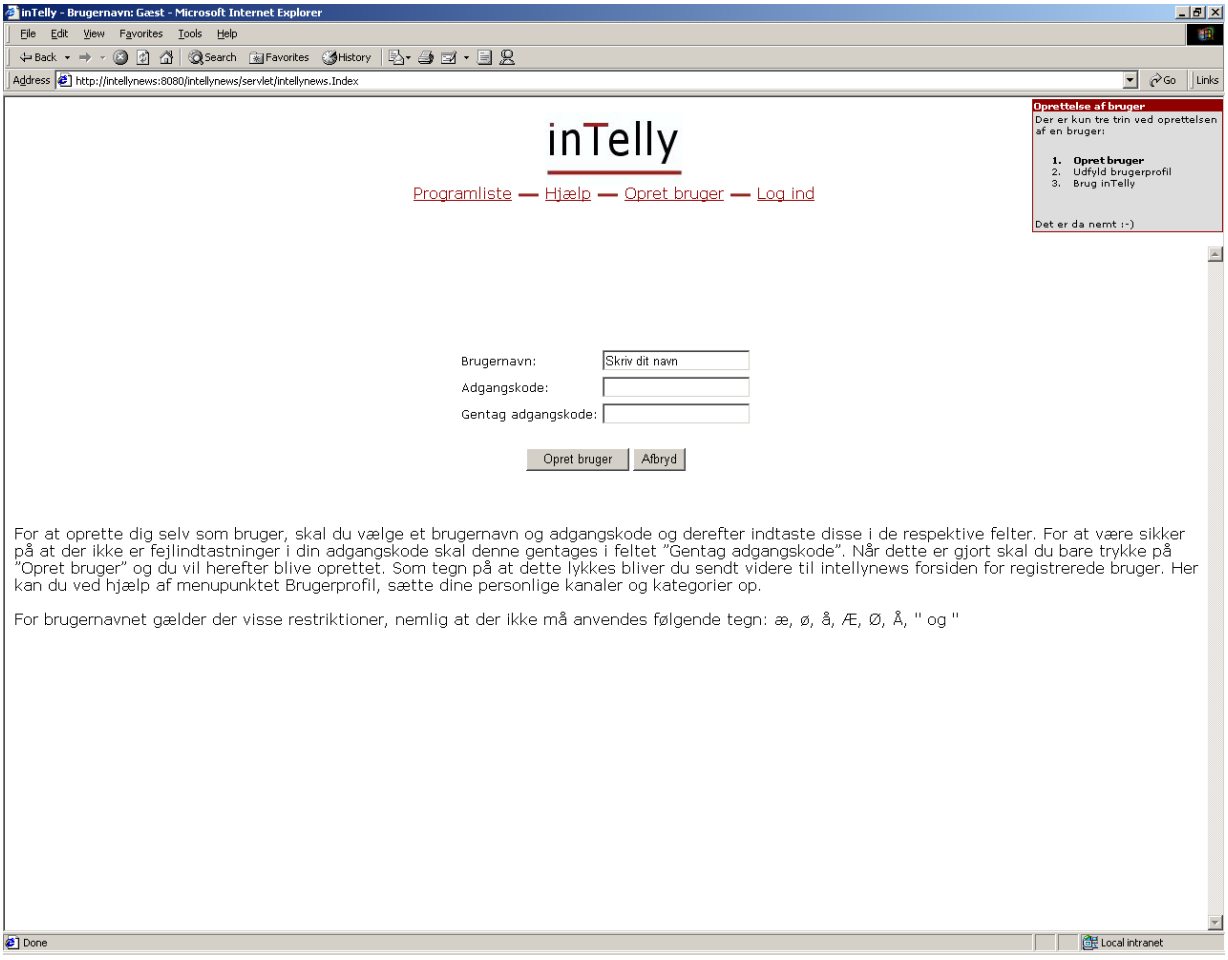

Figure 58 : Screendump of the create user screen version 0.3.

Login

The login page seen on Figure 59 has not changed since version 0.2.

Figure 59 : This screendump presents the Login screen

User Profile

The user profile has not changed much since version 0.2. The main changes in this version are the new pages that are added to the user profile. The new pages are keywords and repeaters pages. The front page of the user profile has also changed a bit (see Figure 60). The difference is that the user now is able to see which user's profile is showing. There has also been added the possibility of entering the users email address so that he or she can receive an email each morning containing the TV-programs for the day. It is also possible to see when the user profile last was saved. There has also been added a navigation button to each page that brings the user to the next page.

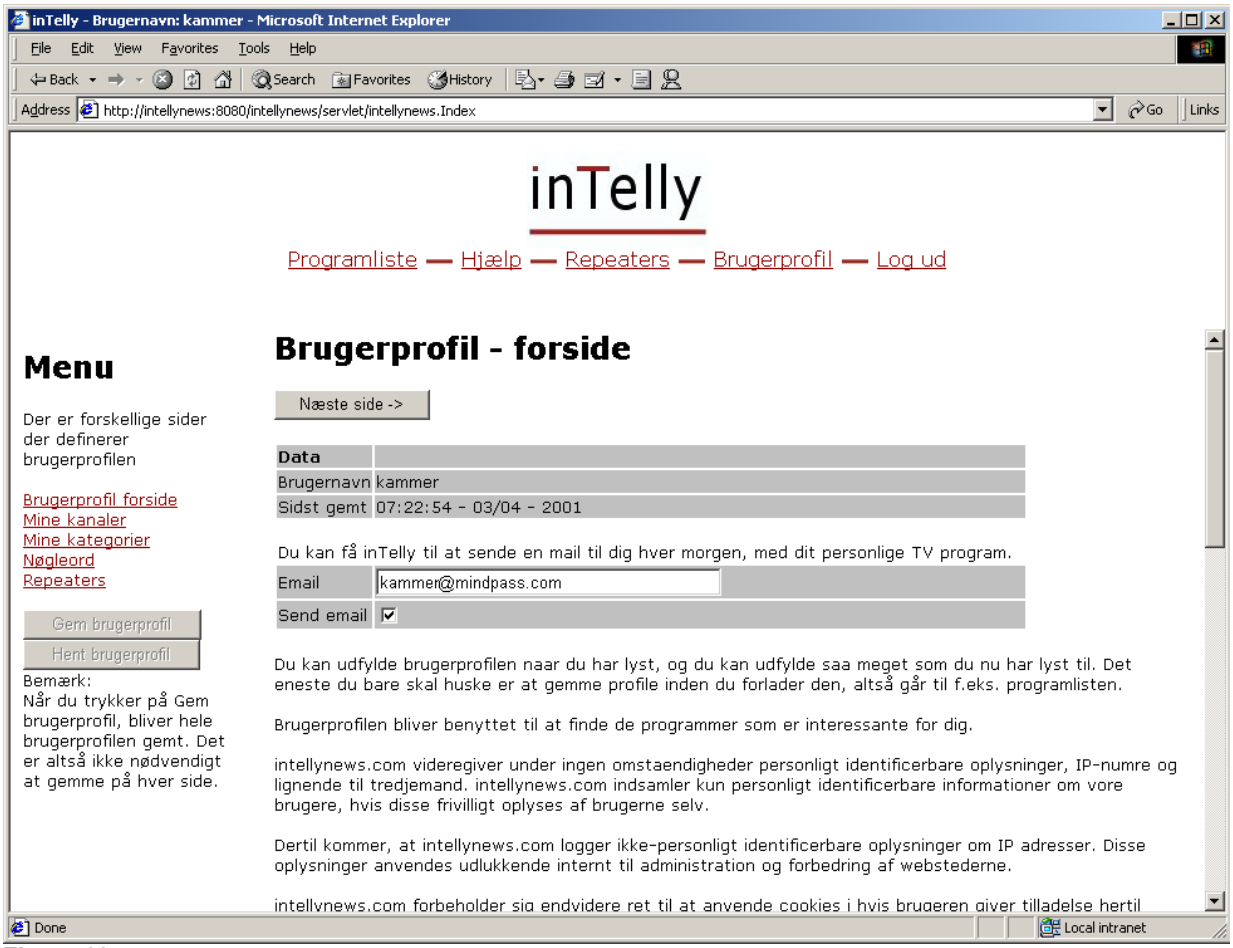

Figure 60 : Here is the first page of the user profile presented.

The next page in the user profile (see Figure 61) is the channel selection page where the user can select the wanted channels. The only difference from version 0.2 is the amount of text presented on the page is reduced.

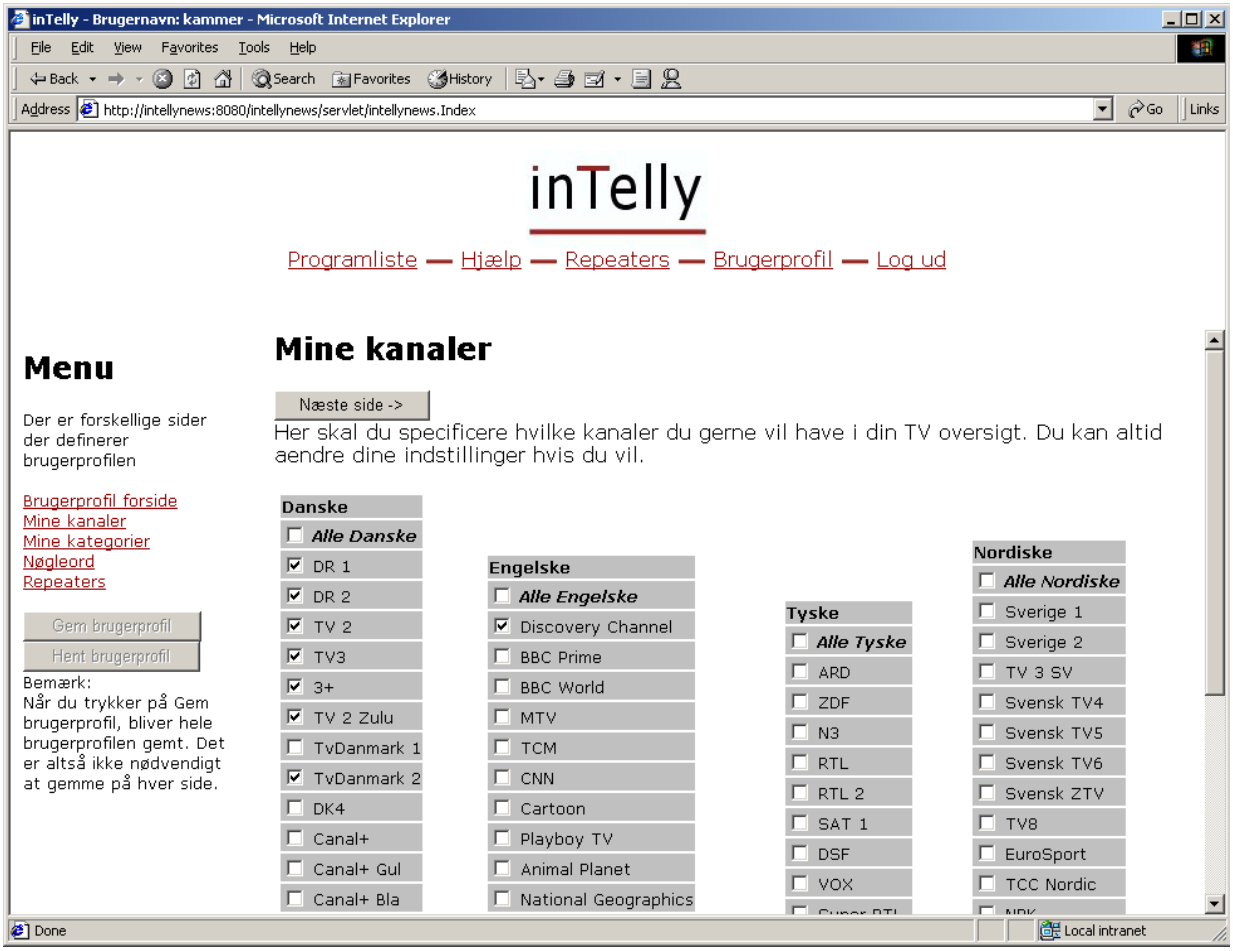

Figure 61 : Second page of the user profile presents the possible channels, which are in the system.

As in version 0.2 the next interface is the categories page where the user can select the wanted categories he or she is interested in. This page has not changed from version 0.2 (see Figure 62).

Figure 62 : On the third page of the user profile the possible categories are presented.

Next is the new keyword interface (see Figure 63) where the user has the possibility to type in keywords that that he or she thinks is interesting. The keywords are typed in separated by spaces.

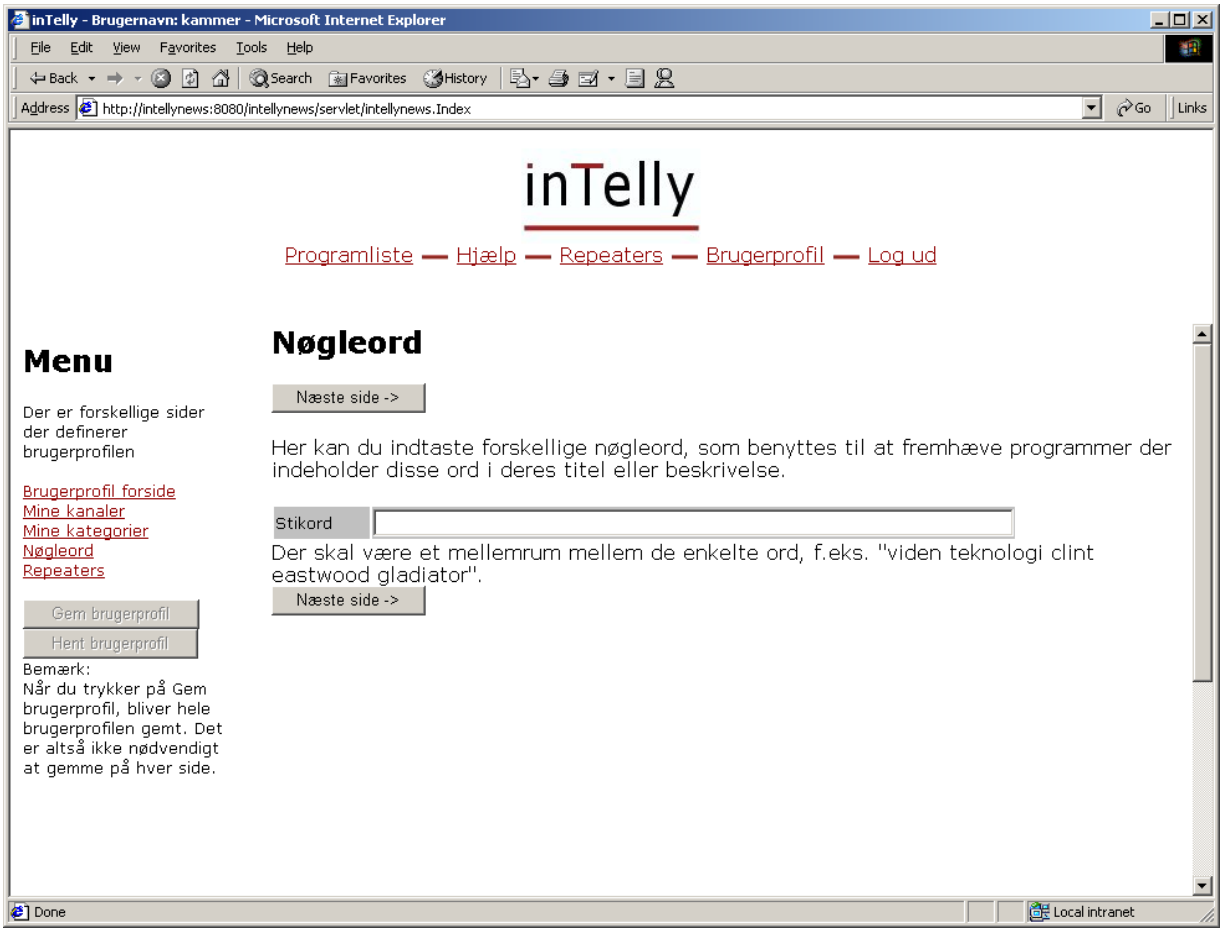

Figure 63 : The page where the user can type in some keywords.

The last page in the user profile is the Repeaters page. The actual Repeater list is present in its own page (see Figure 65) and the design is like that of the program list but the introduction to the feature repeaters is done in the user profile and can be seen in Figure 64.

Figure 64 : The last page of the user profile contains the introduction to the repeaters list.

The repeaters list shown in Figure 65 is almost the same as in version 0.2. This page is corresponding to the Repeaters page in version 0.1 although the design has been changed. The major change is the update (Danish: "Opdater") button, which first lets the users update the page when the button is pressed instead of updating it continuously like in version 0.2. The prioritising of the repeaters is similar to the prioritising of the programs in the program list.

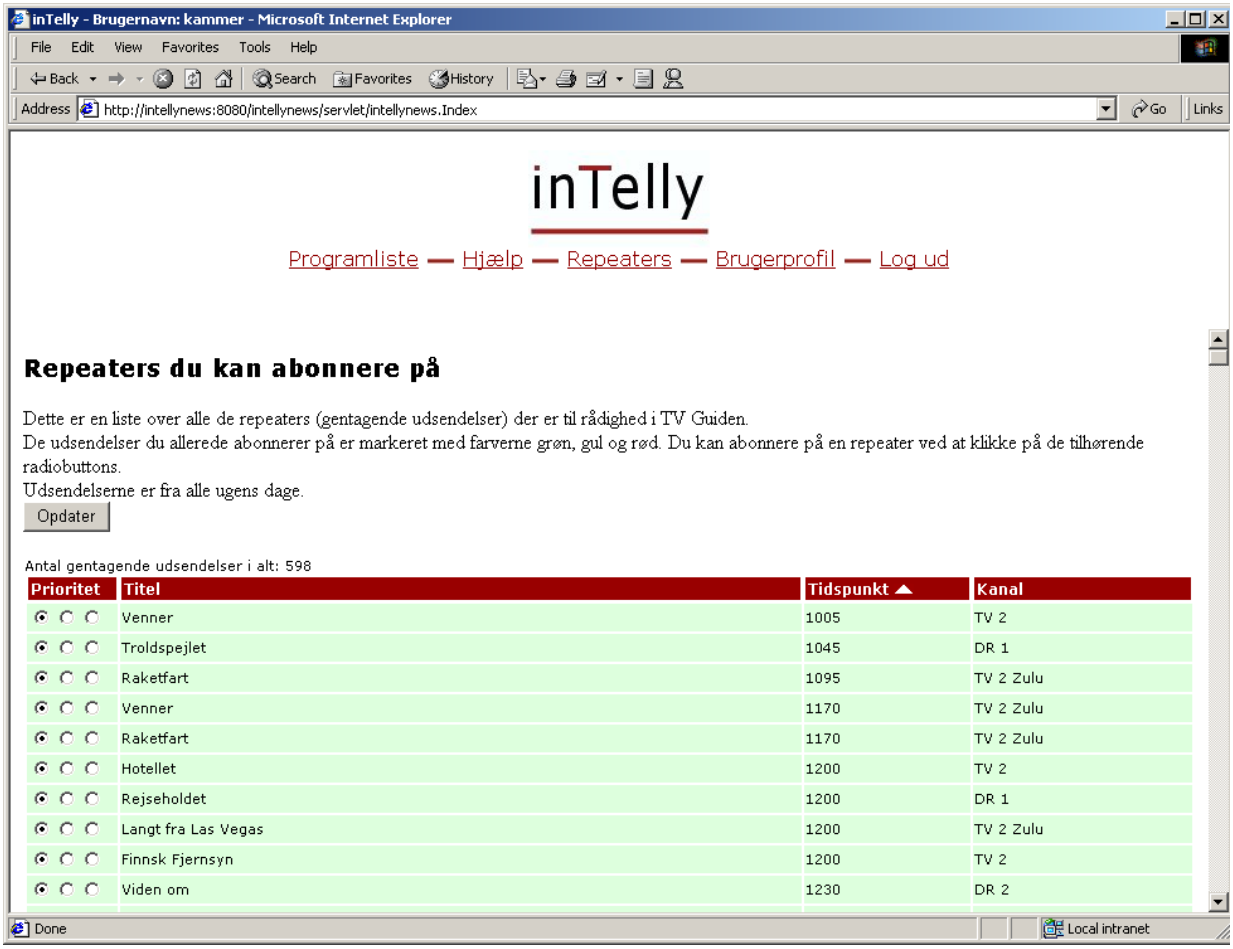

Figure 65 : This page contains the repeaters list where the user can choose which repeating programs he/she wants to see or not.

Programs List

The program list has not been changed drastically from the previous version. There have only been a few minor graphical changes to make it easier to understand for the user.

Figure 66 shows the program list for non-registered users. The main difference from version 0.2 is that there has been added a box called “Filter” showing the solidarity of the four select boxes. Another box with the title additional functionality (Danish: “Supplerende funktioner”) has also been created. In this box the short description check box is located together with the search facility and the print facility. There is also an install link (Danish: “Installer XML”), which sends a new user to download page where he/she can download the necessary installations for XML.

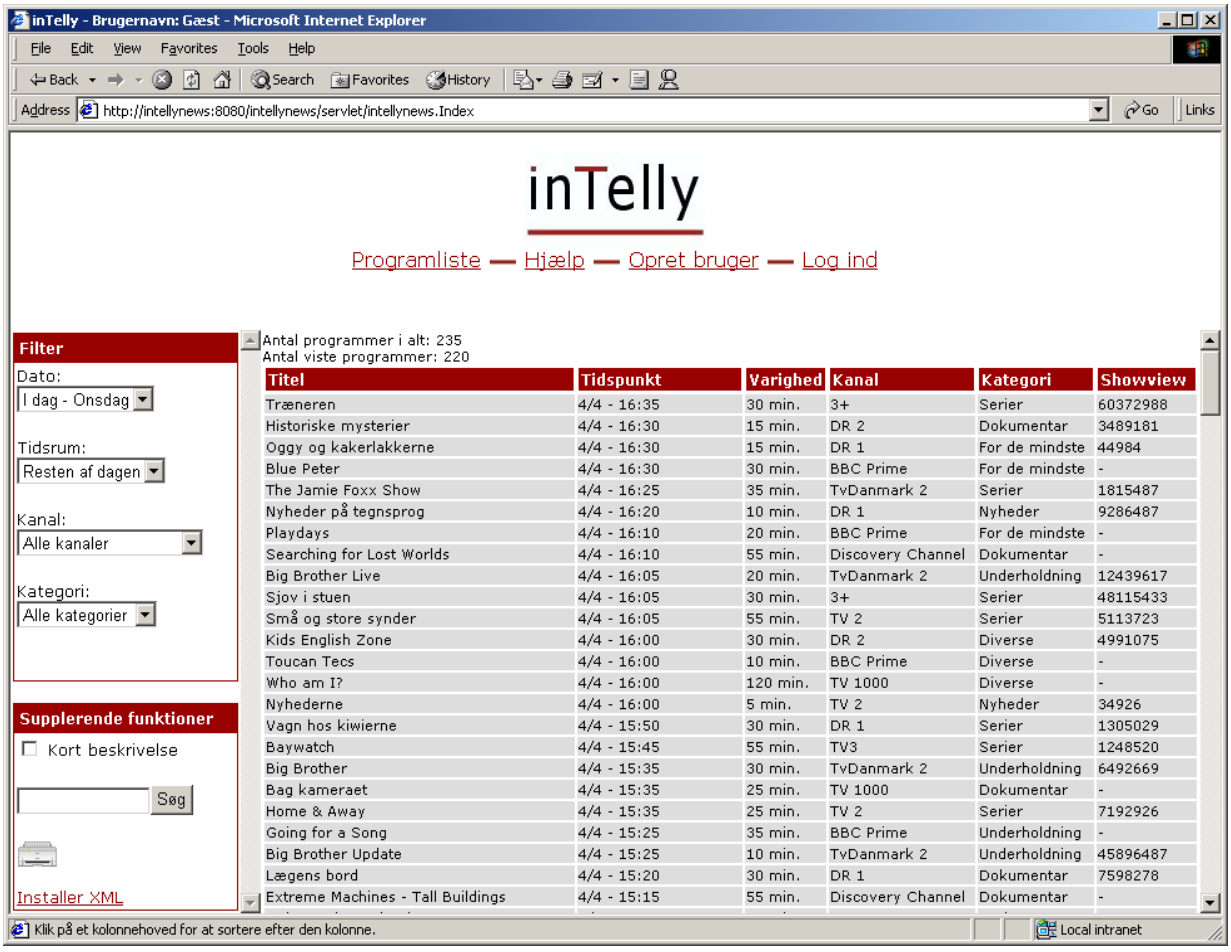

Figure 66 : The program list for the users that are NOT logged in to the system.

As already mentioned the changes of the program list user interface are only cosmetic. The changes made to the interface for users that is logged in to the system is the same as just mentioned above. The only difference is the show hidden (Danish: “Vis skjulte”) check box, which has been placed in the filter box (see Figure 67).

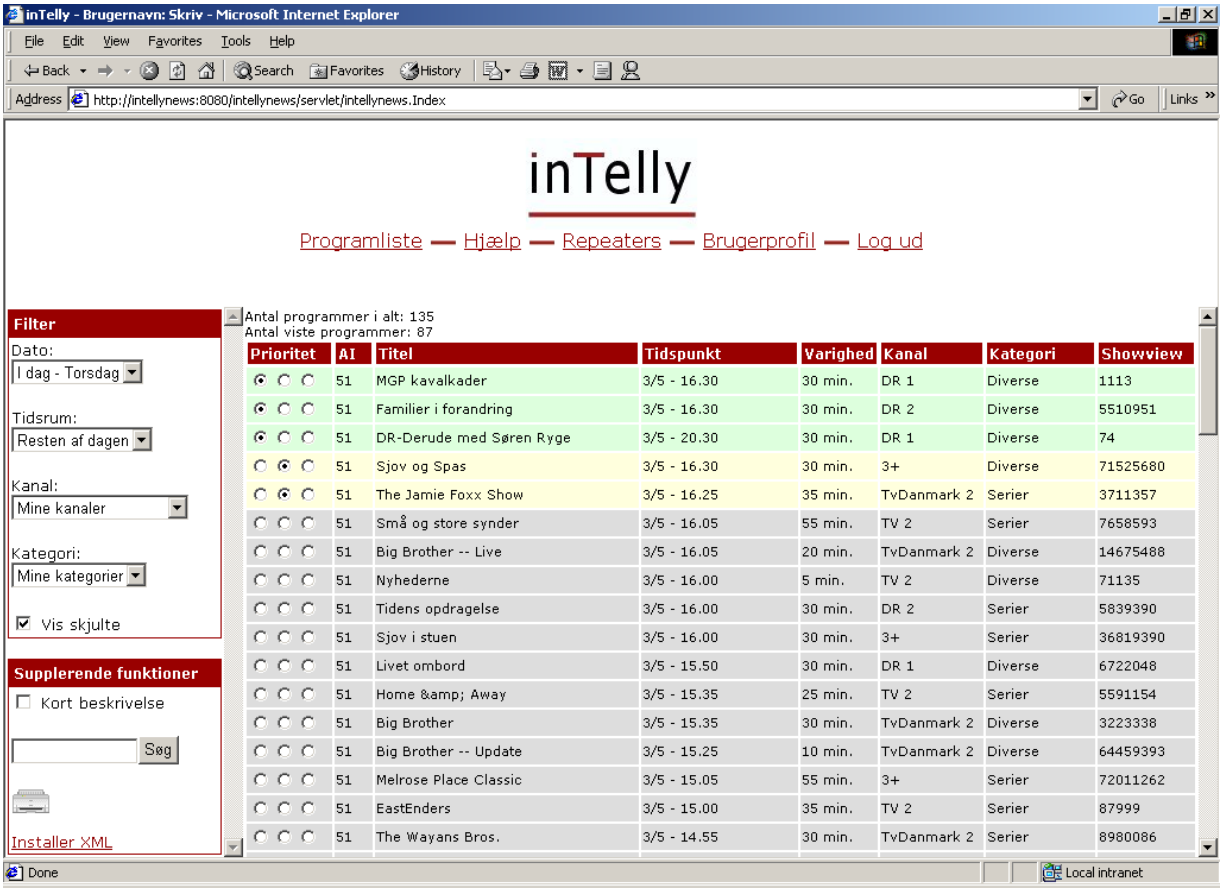

Figure 67 : The program list for logged in users

Figure 68 shows the program list where a pop-up description of a program is showed. The pop-up description of the program is shown as soon as the mouse pointer points at the program title.

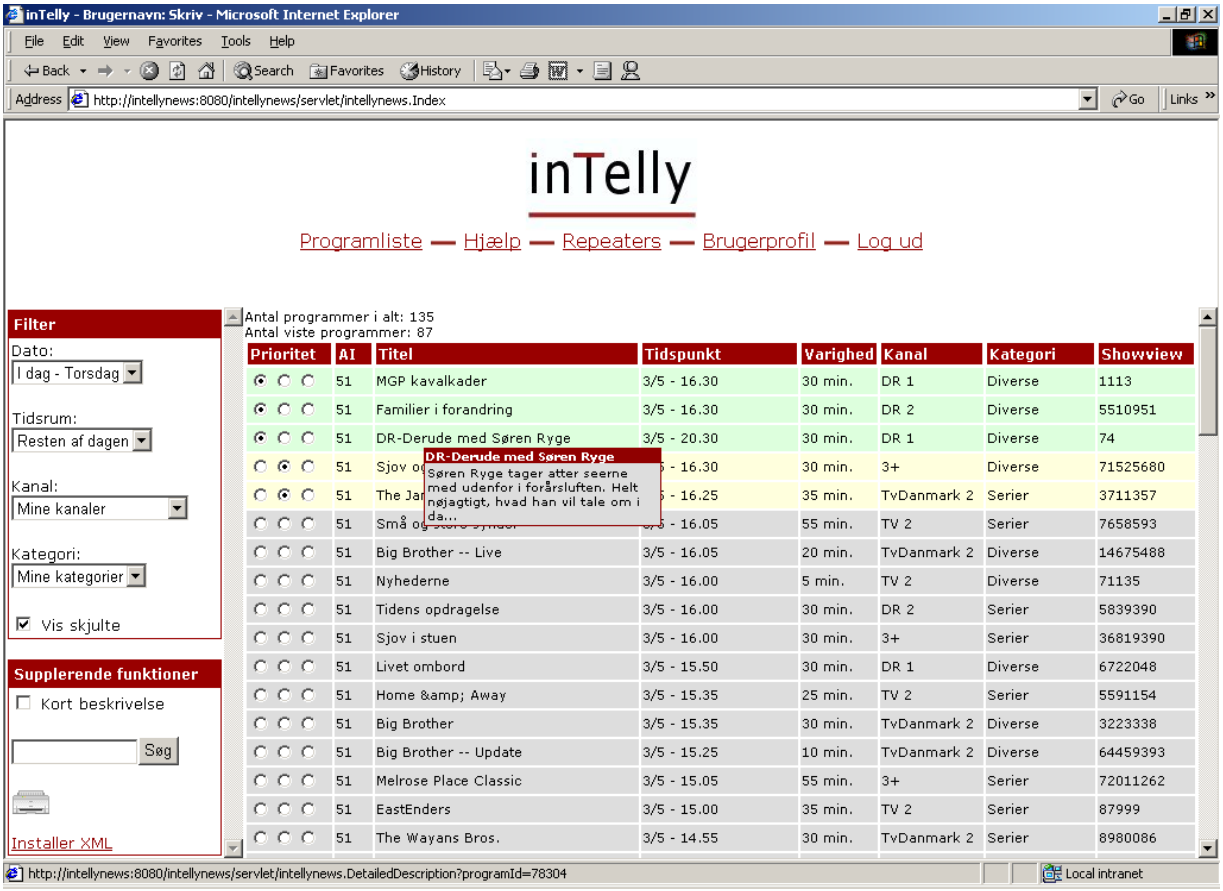

Figure 68 : The program list where a pop-up description of a program is shown.

Next is the program list with activated short description shown (see Figure 69). Only minor modifications has been made since the version 0.2. The description is now accompanied by a graphical representation of the start and stop time for the program.

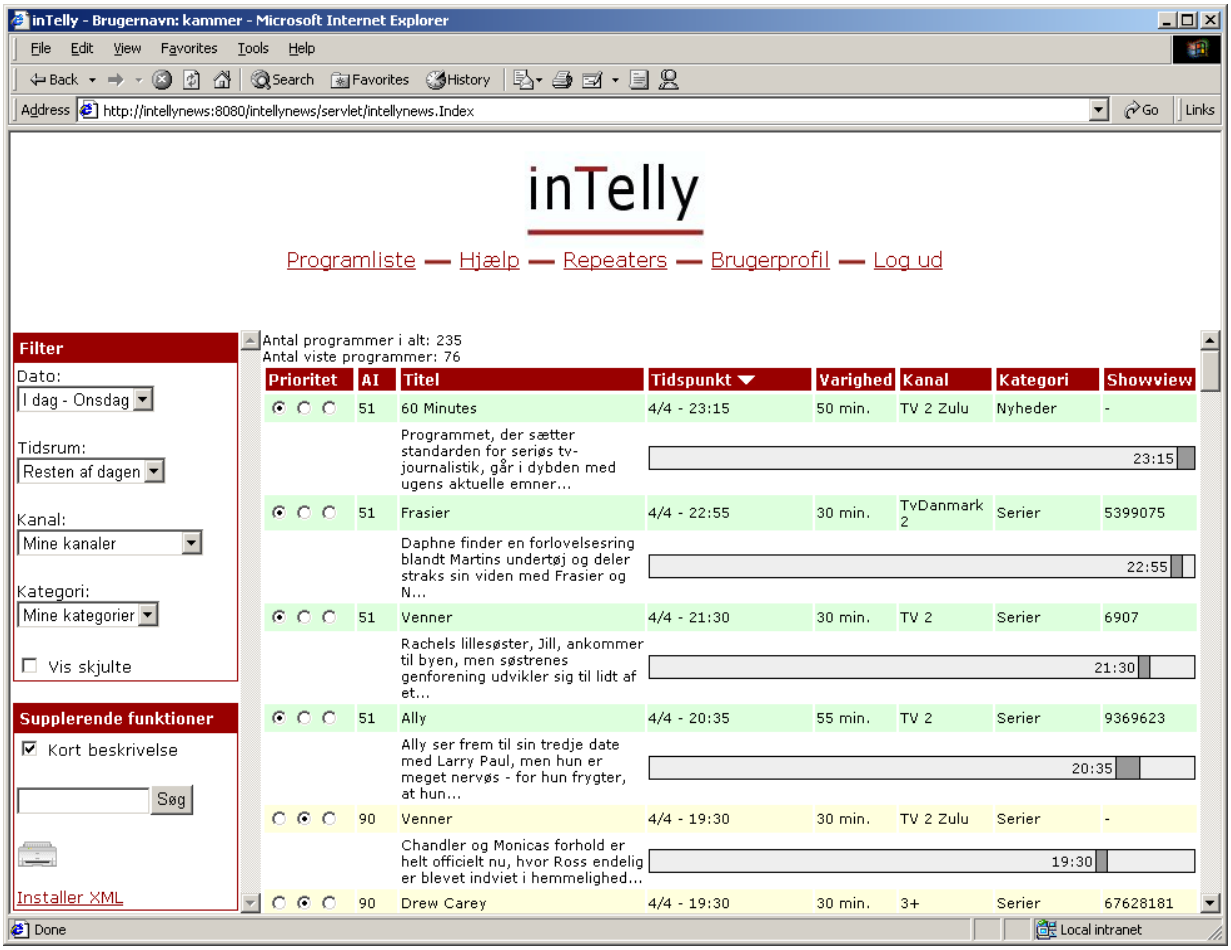

Figure 69 : This is the program list for the users that are logged in to the system. Here there have chosen to get a short description presented together with the program listening.

Detailed Description

The detailed description has changed massively since version 0.2 (see Figure 70). The new design of this page only present data concerning the TV-program and not like before a lot of additional information. It is also possible to prioritise the program at this page and if the program is a Repeater it can also be added with the desired priority to the repeaters list. The information showed here on the detailed description is now the same as the data shown in the program list.

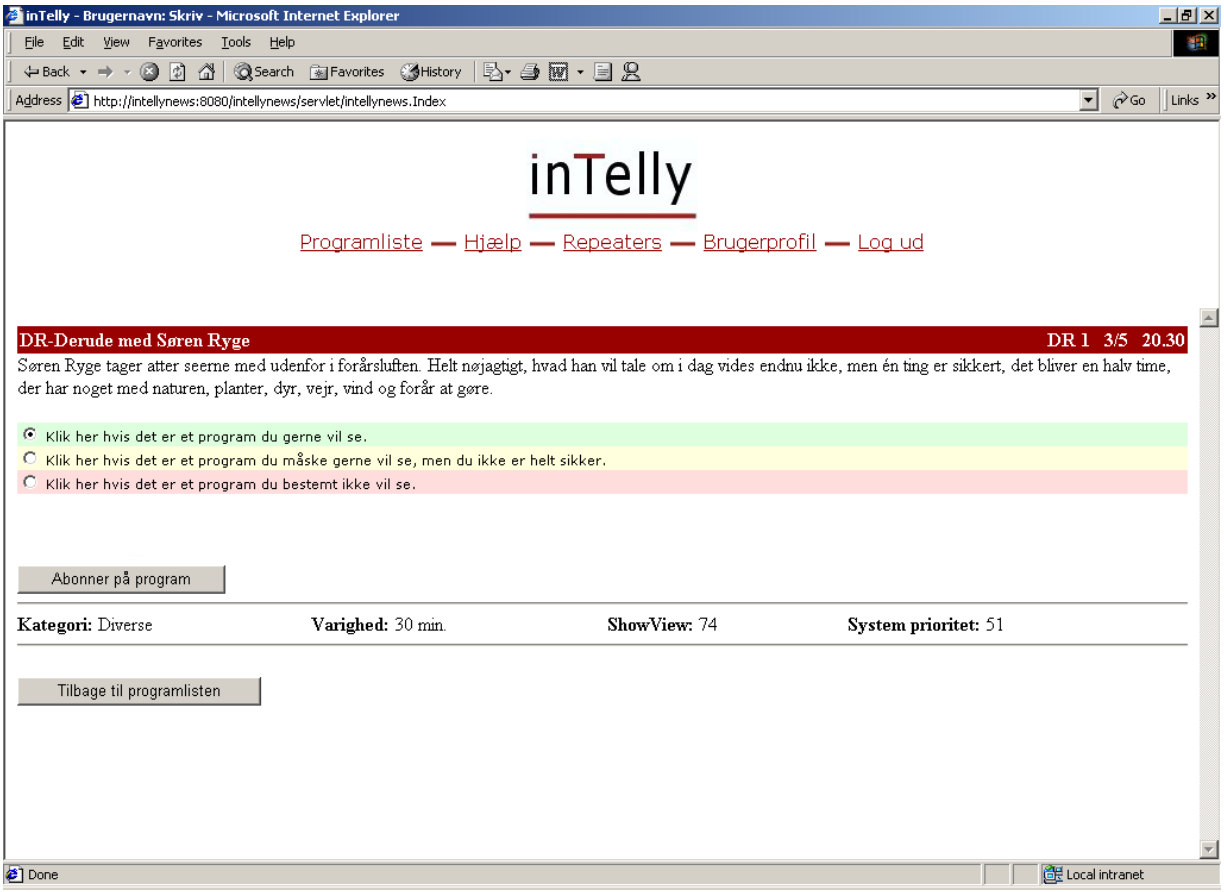

Figure 70 : This is the detailed description where the user can read more about the program.

Help

The inTelly system help has also changed a bit in this version. The major changes are the new page that has been added and the fast help facility that has been added. The extra page is concerning the messages in the system. The fast help facility is the picture located below the menu at the left side. When clicking e.g. the filters of the picture the users will be sent to the help page concerning the filters. There has also been made some cosmetic changes to the help like changes in the names of the pages.

The first page is the help front page (see Figure 71) that gives an introduction to the help facilities.

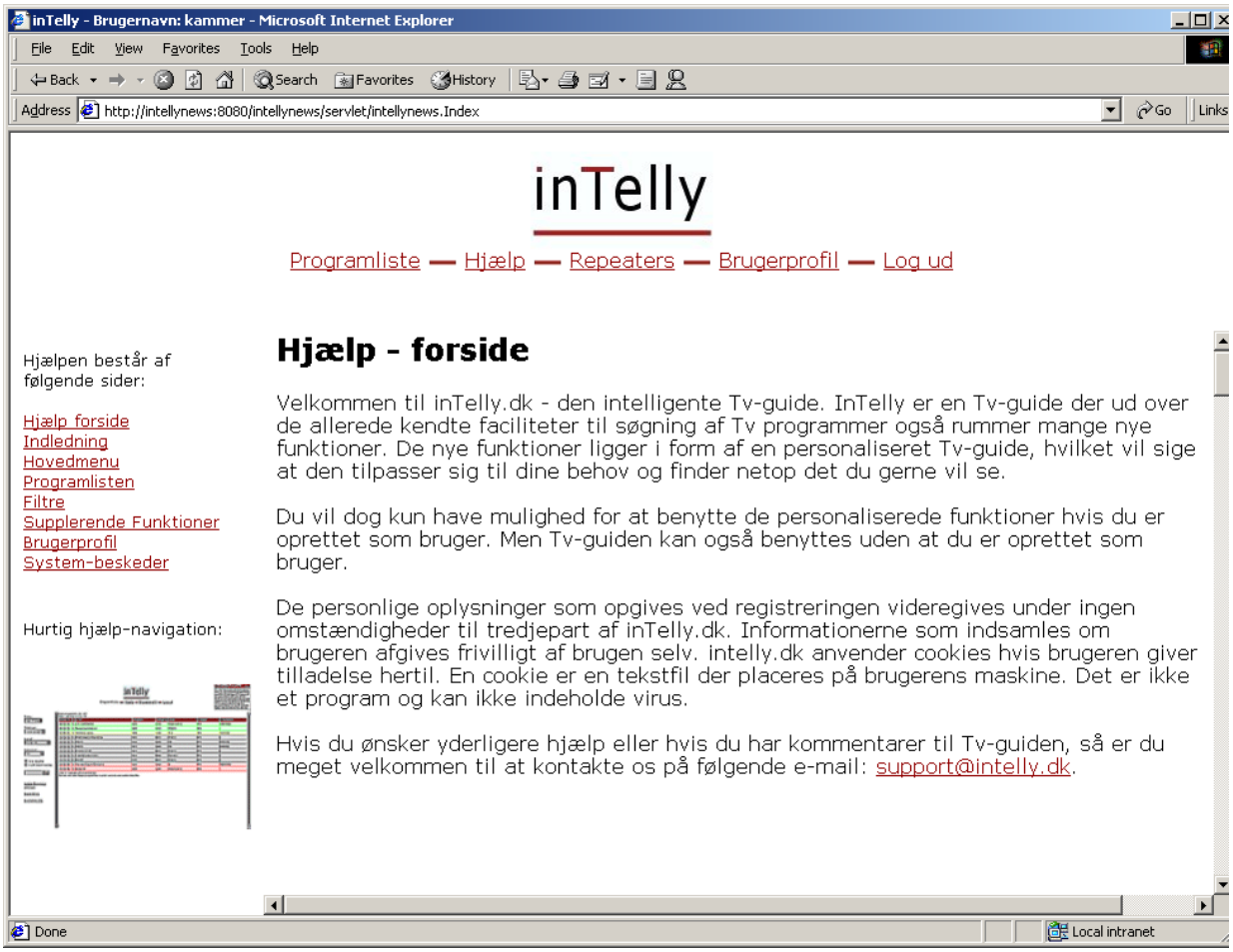

Figure 71 : Ths is the help front page from where it is possible to navigate deeper into the help facilities.

The next page is the introduction (Danish: “Indledning”) to the inTelly system, which also existed in the last version. See the new design on Figure 72.

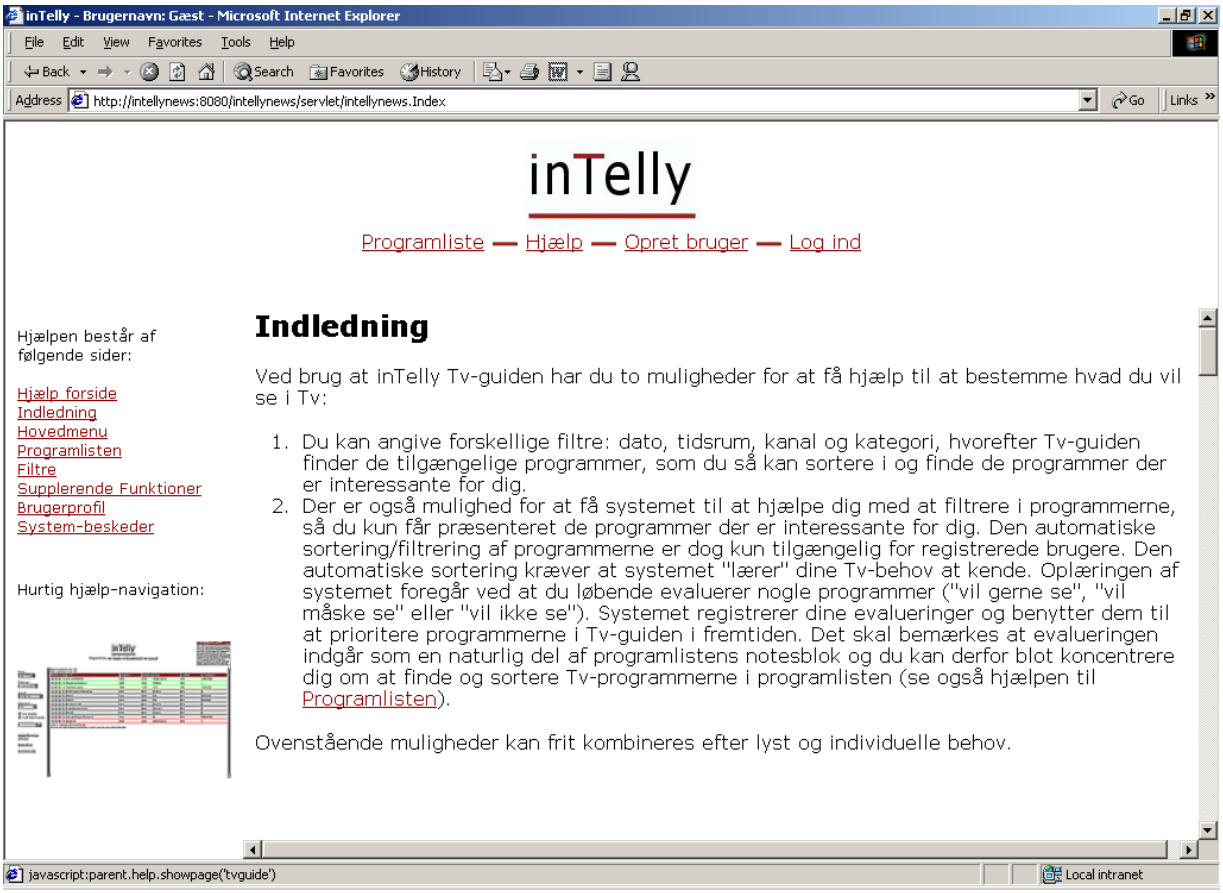

Figure 72 : The introduction page within help.

Next is a presentation of the main menu that is present on all pages. There is also a description of the different links (see Figure 73).

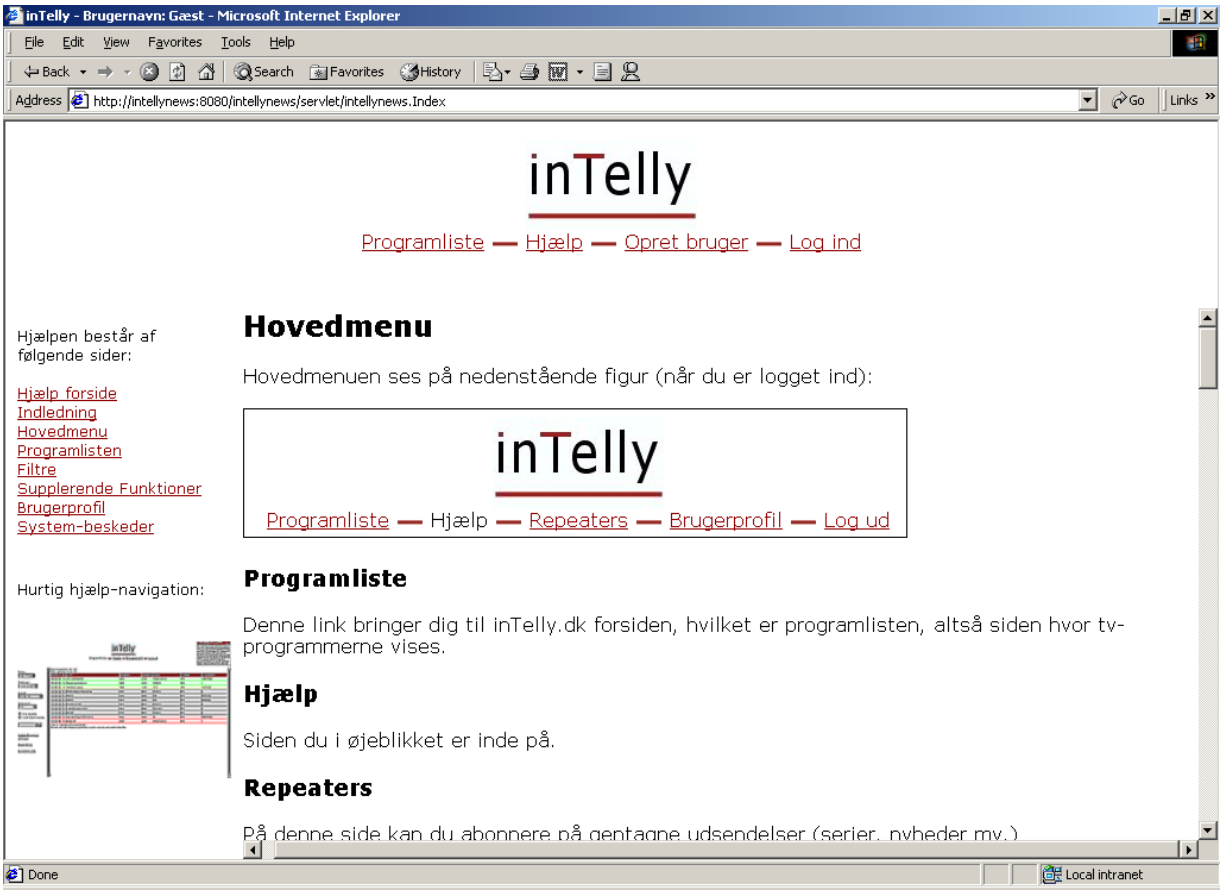

Figure 73 : The help page explaining the main menu.

Next is the description of the program list, how it should be used and how it is working (see Figure 74).

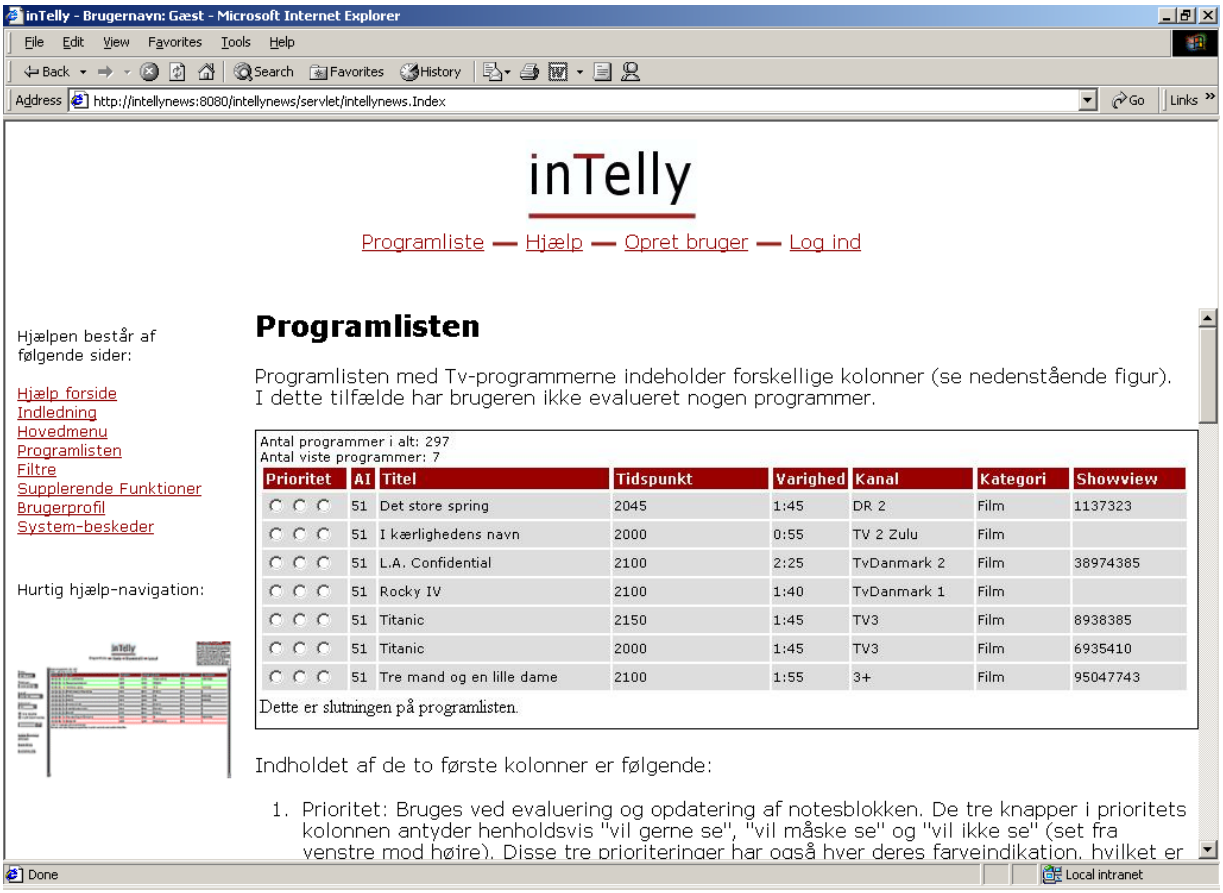

Figure 74 : The help for the program list.

Figure 75 contains the help for the use of the filters on the program list.

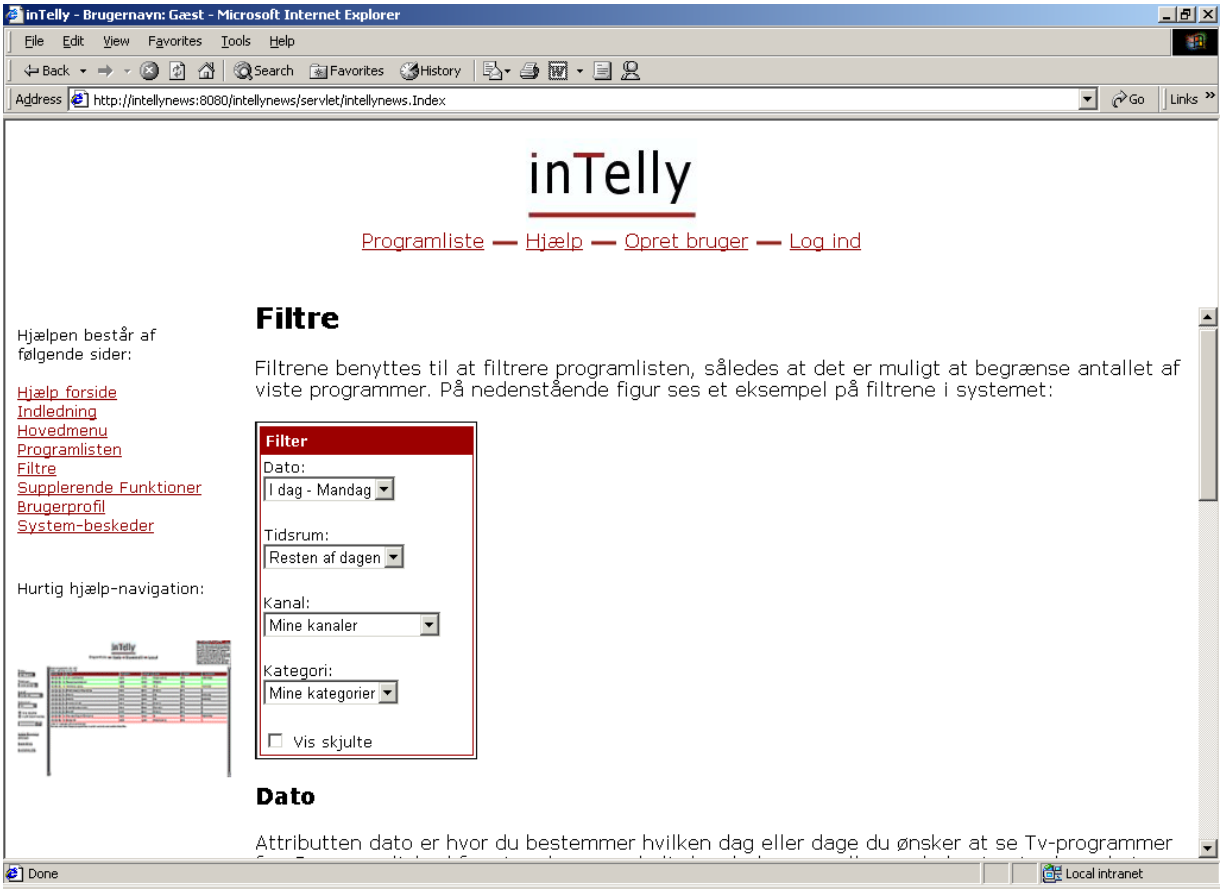

Figure 75 : Filters help page.

After the filters page the page with help for the additional functions (Danish: “Supplerende funktioner”) is placed. This page can be seen on Figure 76.

Figure 76 : The help page for the additional functions.

Figure 77 shows the help for the user profile. This page has like the other not much changed since last version.

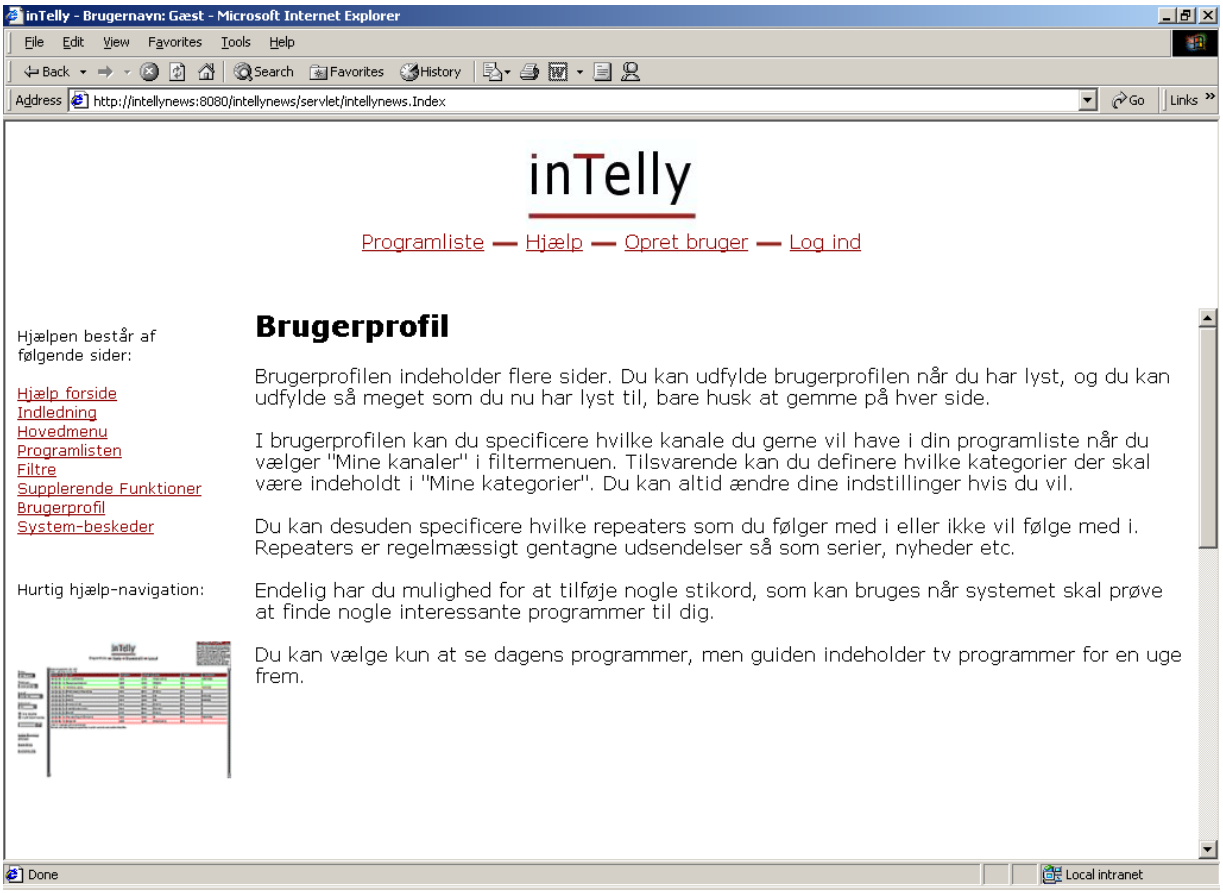

Figure 77 : The help concerning the user profile.

The last page in the system help is the help concerning the system messages. This page describes what the messages are and what they are used for (see Figure 78).

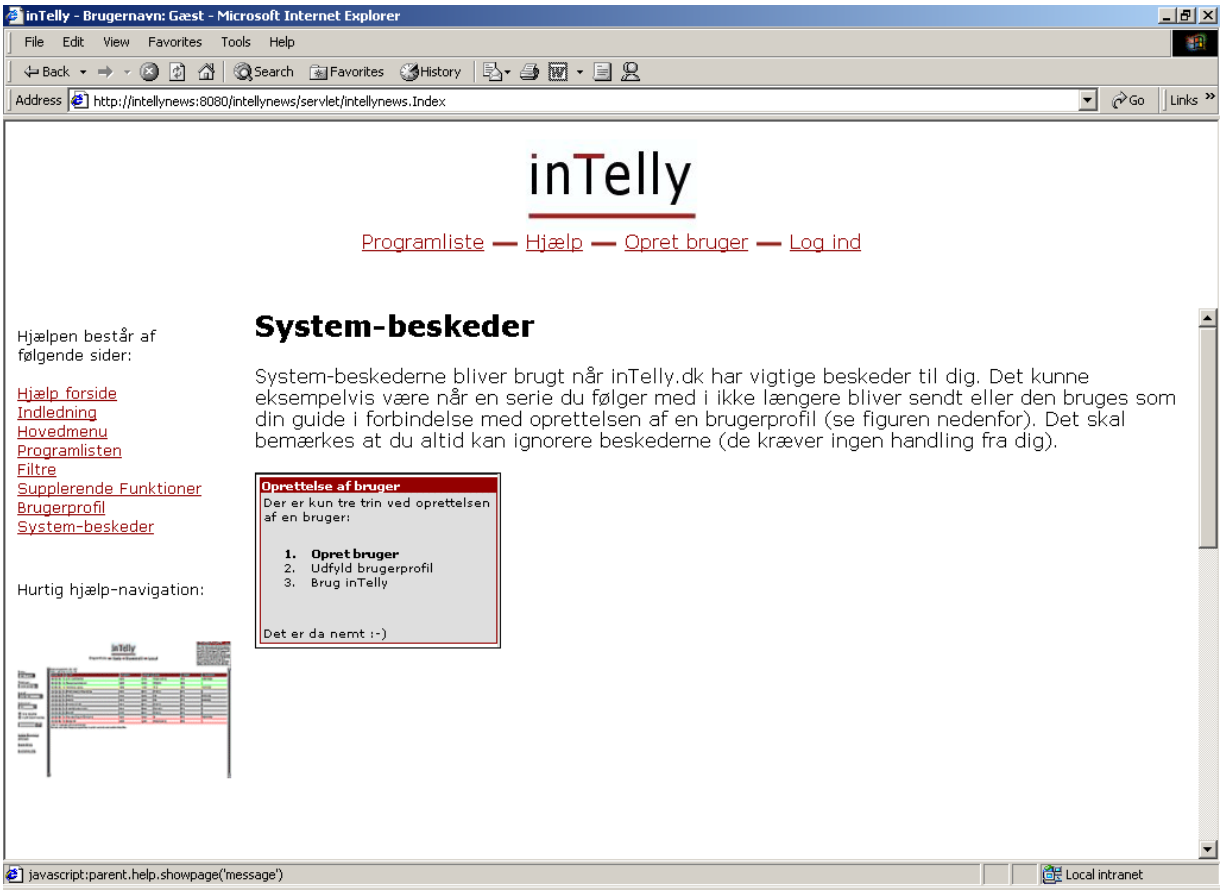

Figure 78 : The help about the messages in the system.

Menu Bar and Logo

The logo and menu bar has not changed in the last versions. The changes concerning a logged out user are now as seen on Figure 79.

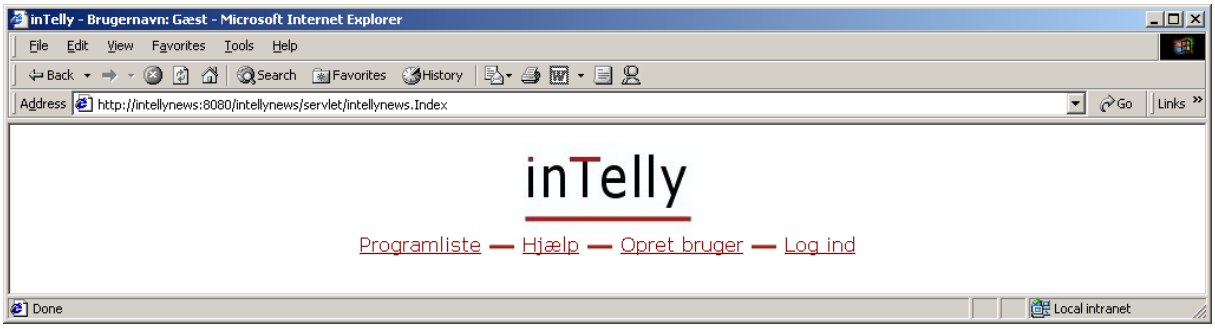

Figure 79 : The logo and menu bar for a logged OUT user

Contrary from the logged out menu bar the logged in menu bar has changed a bit since version 0.2. The difference here is the additional option that is on the latest version. This extra option is the link to the repeaters page (see Figure 80).

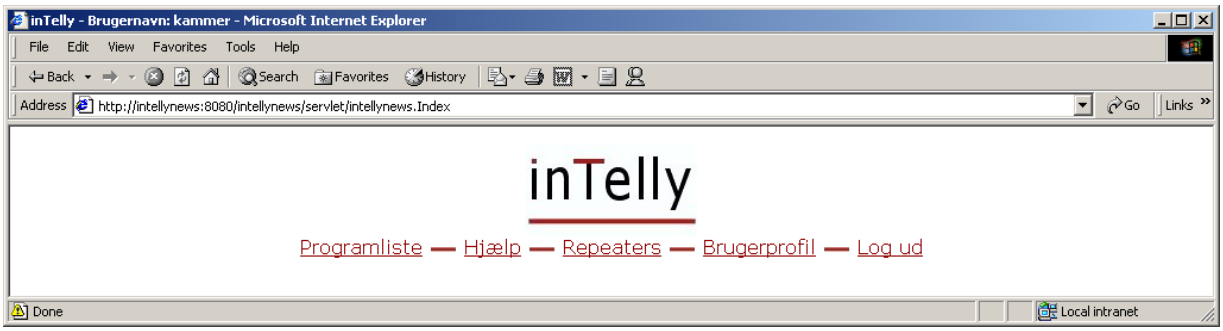

Figure 80 : The logo and menu bar for a logged in user

Messages

Figure 81 shows the new design of the system messages. As it can be seen it has changed since the first draft in version 0.1. The messages are now only used as feedback and information for the user.

Figure 81 : An example of an system message.

P. User Interface Version 1.0

This supplement contains all the user interfaces that are present in the version 1.0 of the inTelly system. There is no description of any of the user interfaces in this supplement. The description of the different interfaces can be found in the *Main report – 20. User Interface*.

Create User

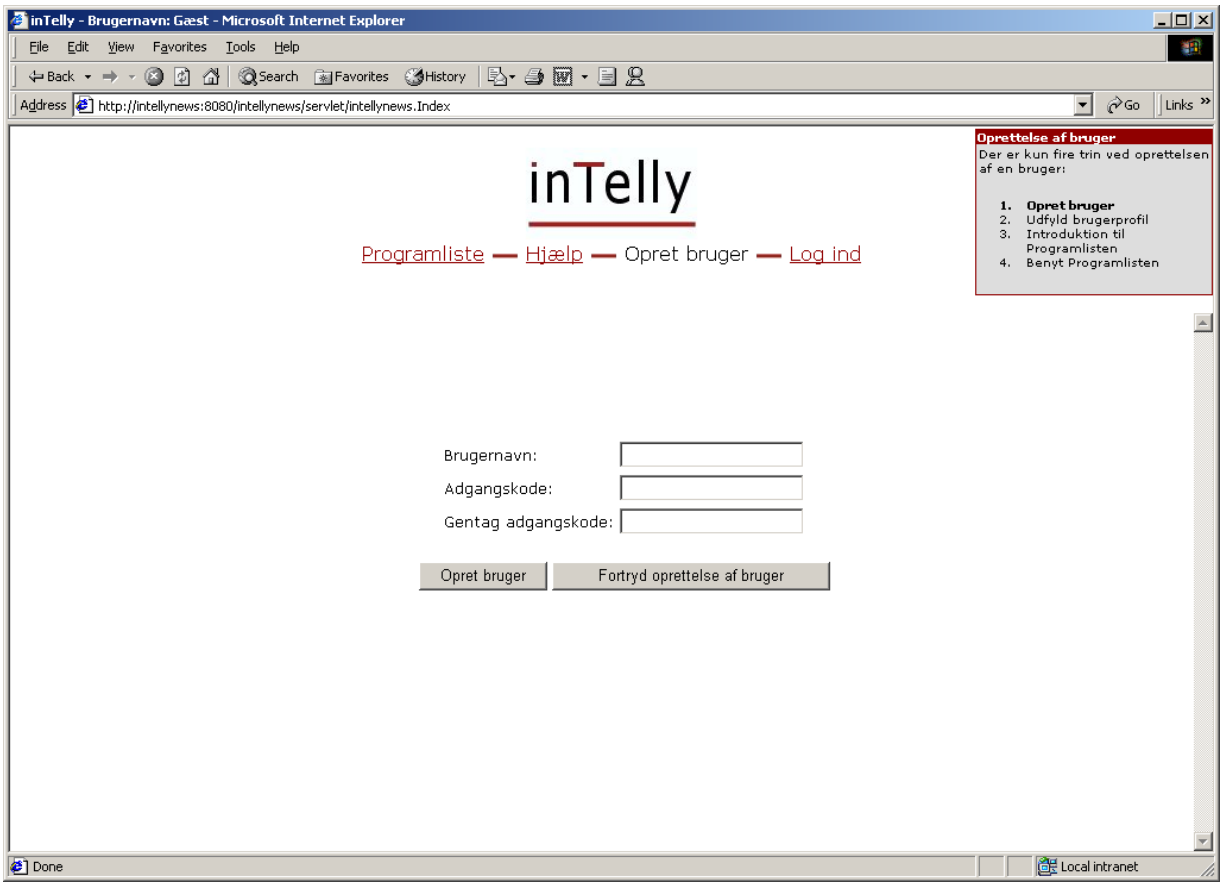

Figure 82 : The create user interface.

Login

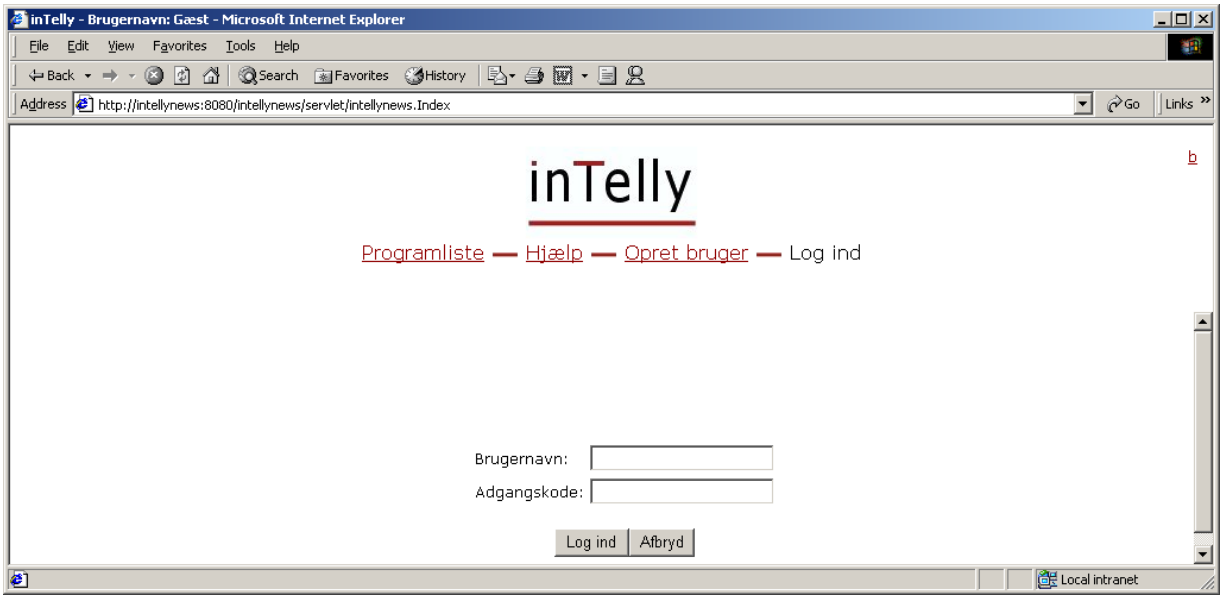

Figure 83 : Login interface for inTelly.dk

User Profile

Figure 84 : The front page of the user profile.

Figure 85 : The channel selection interface in the user profile.

Figure 86 : The interface for choosing the wanted categories in the user profile.

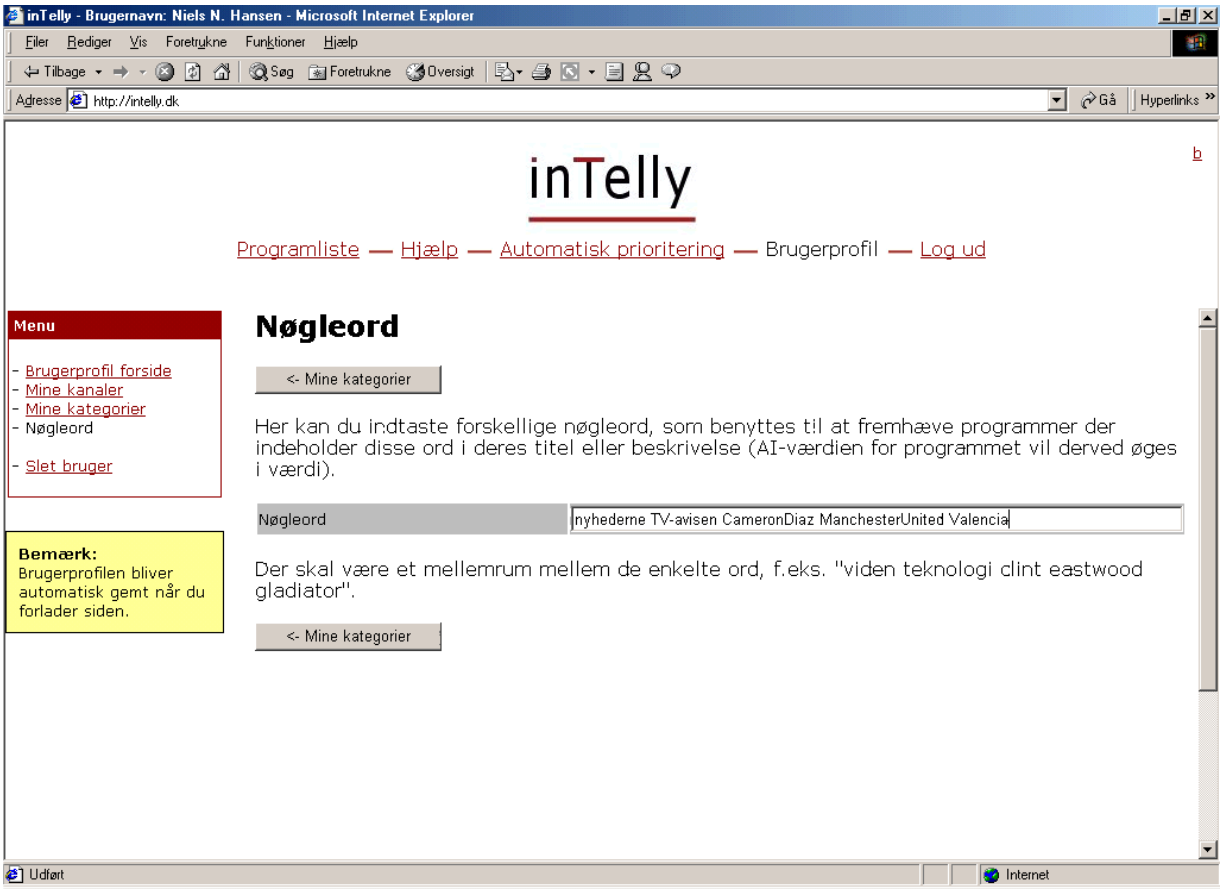

Figure 87 : The keyword page of the user profile.

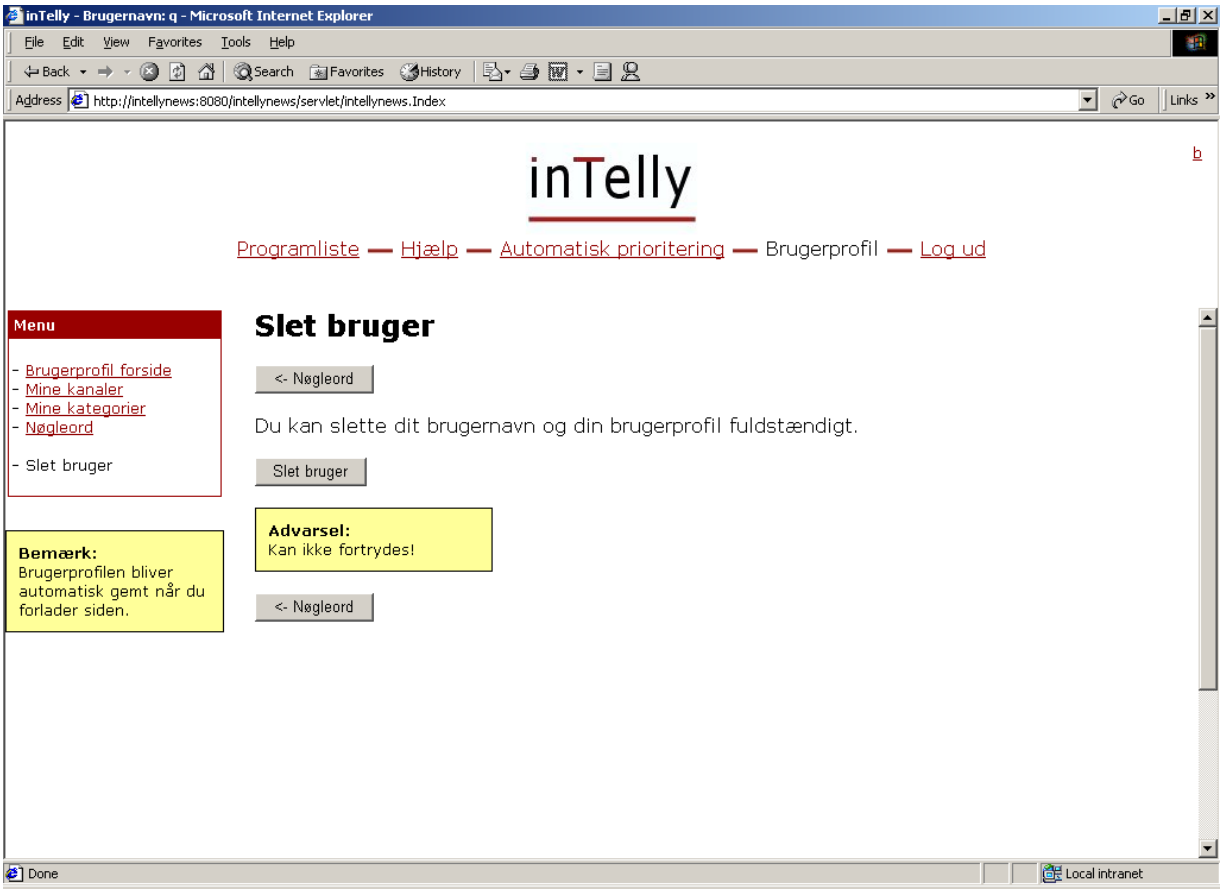

Figure 88 : The delete user interface in the user profile.

Program List

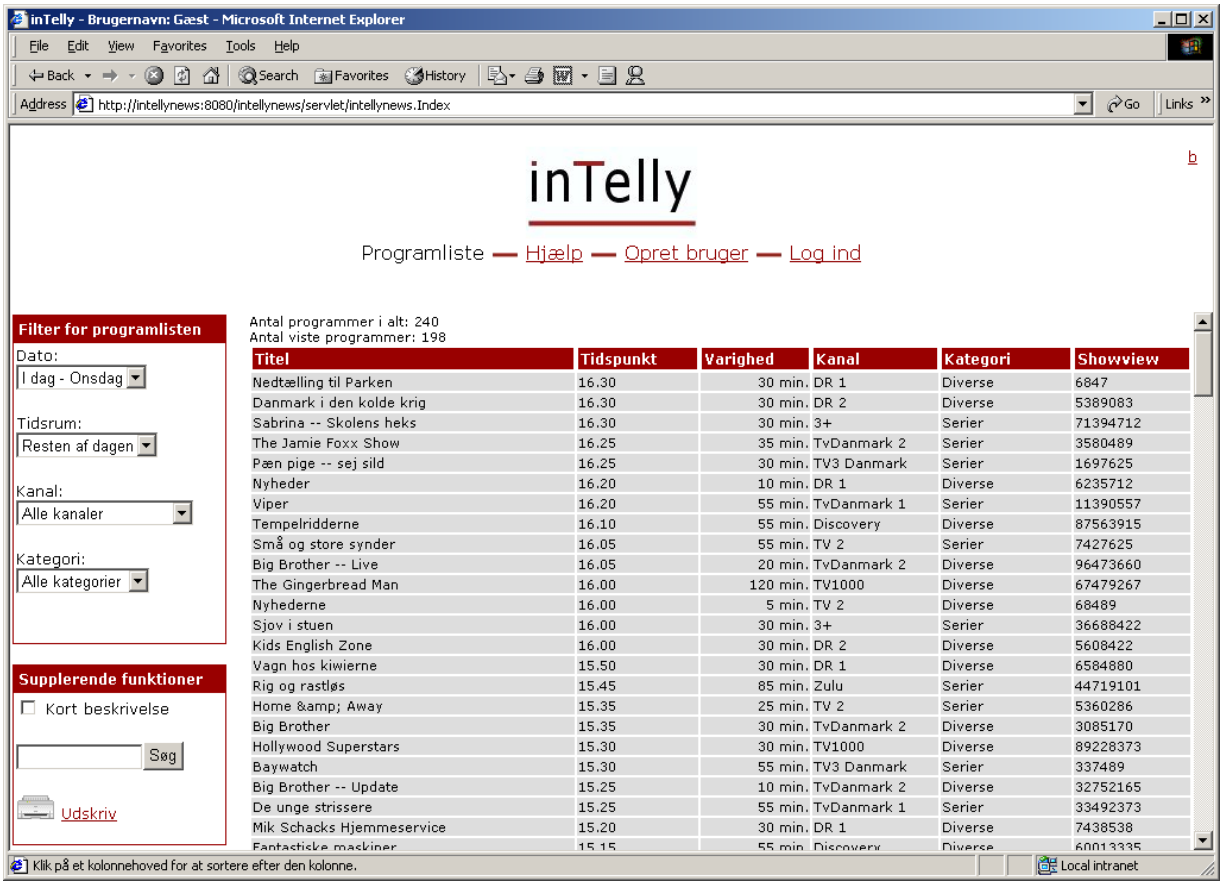

Figure 89 : The program list user interface for the non-registered users.

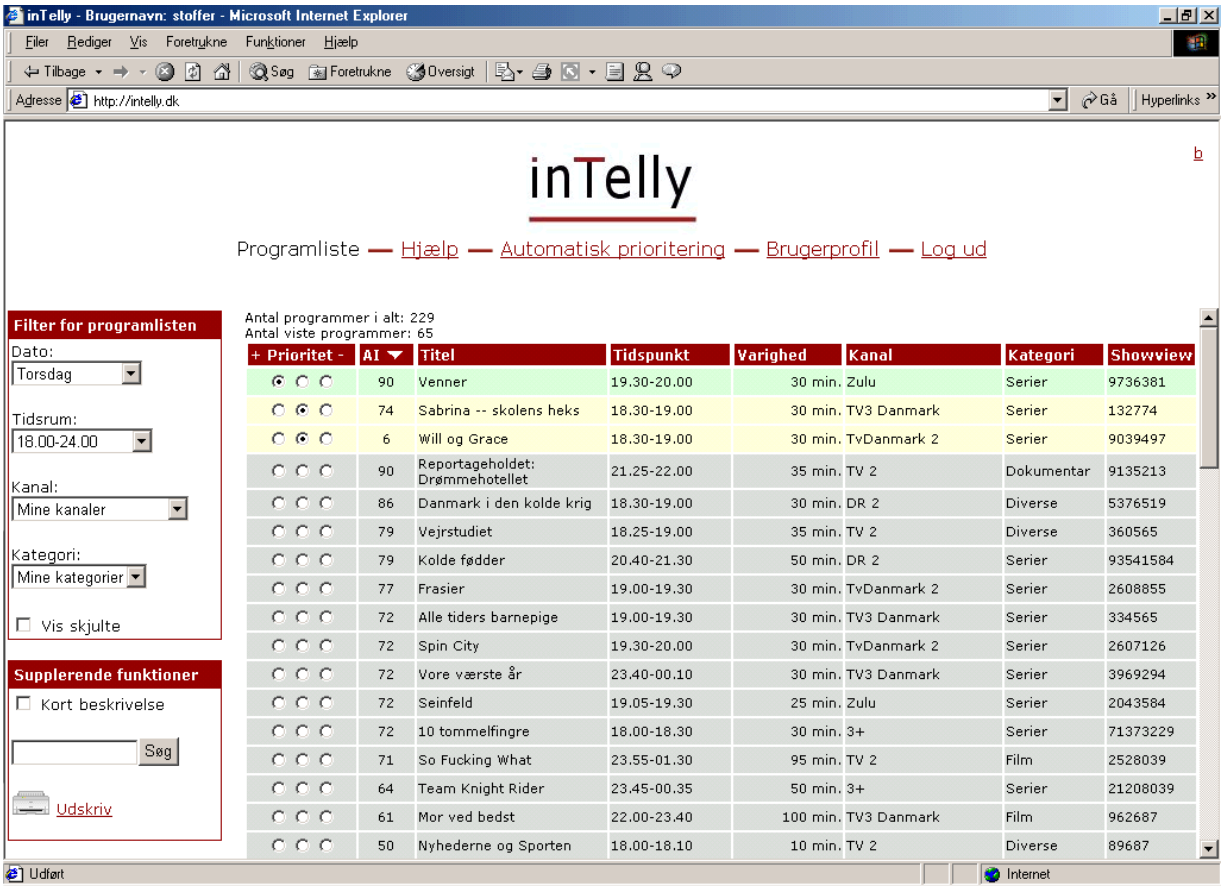

Figure 90 : The program list presented to the logged in users.

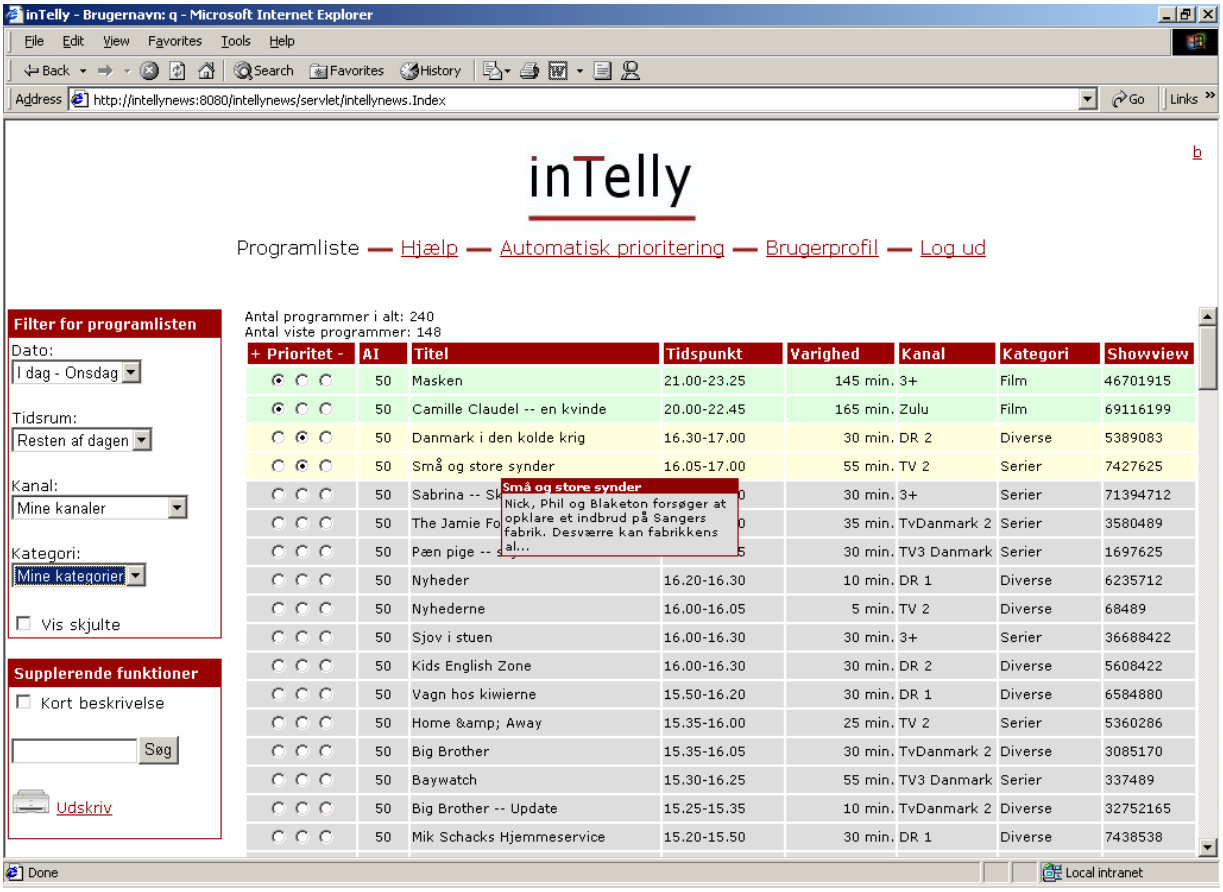

Figure 91 : The program list where a pop-up description on a program is shown.

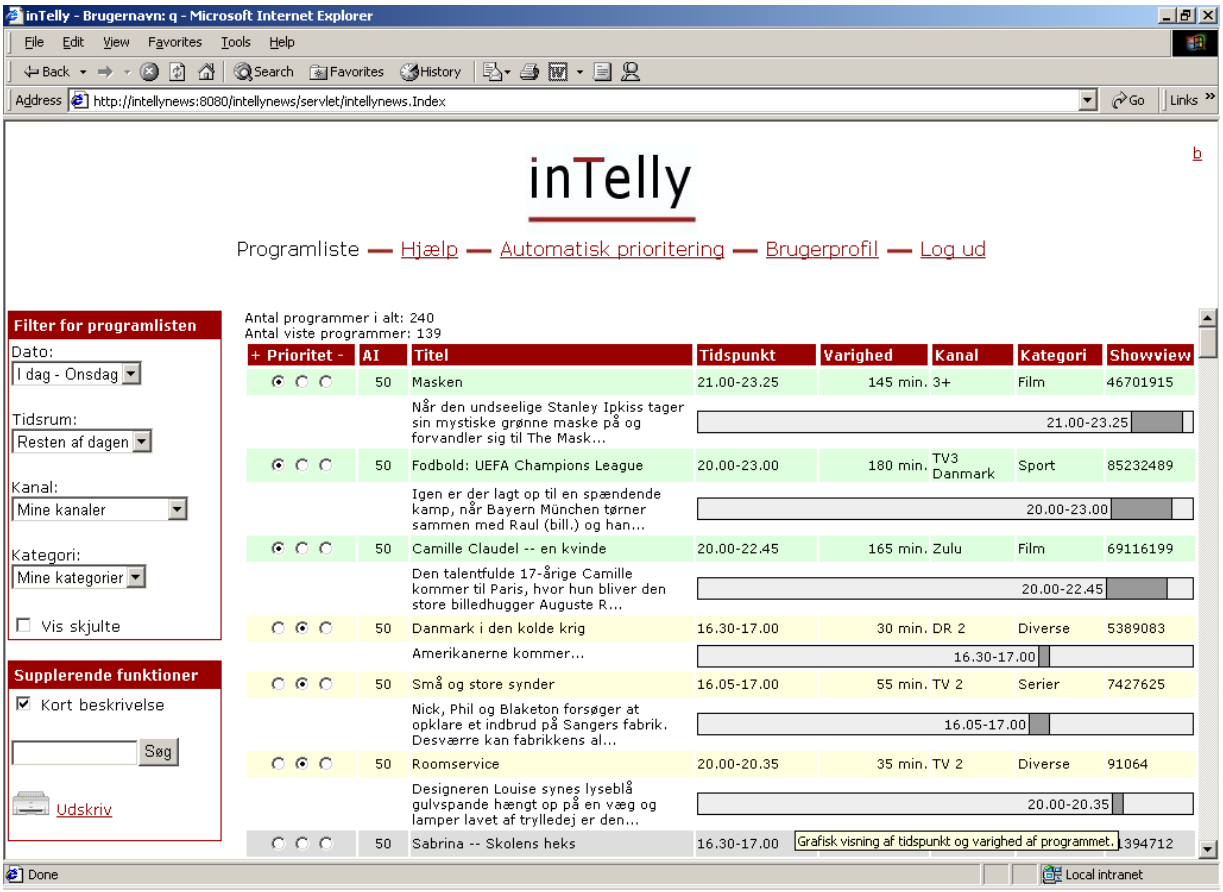

Figure 92 : The program list with the short description presented for each program.

Detailed Description

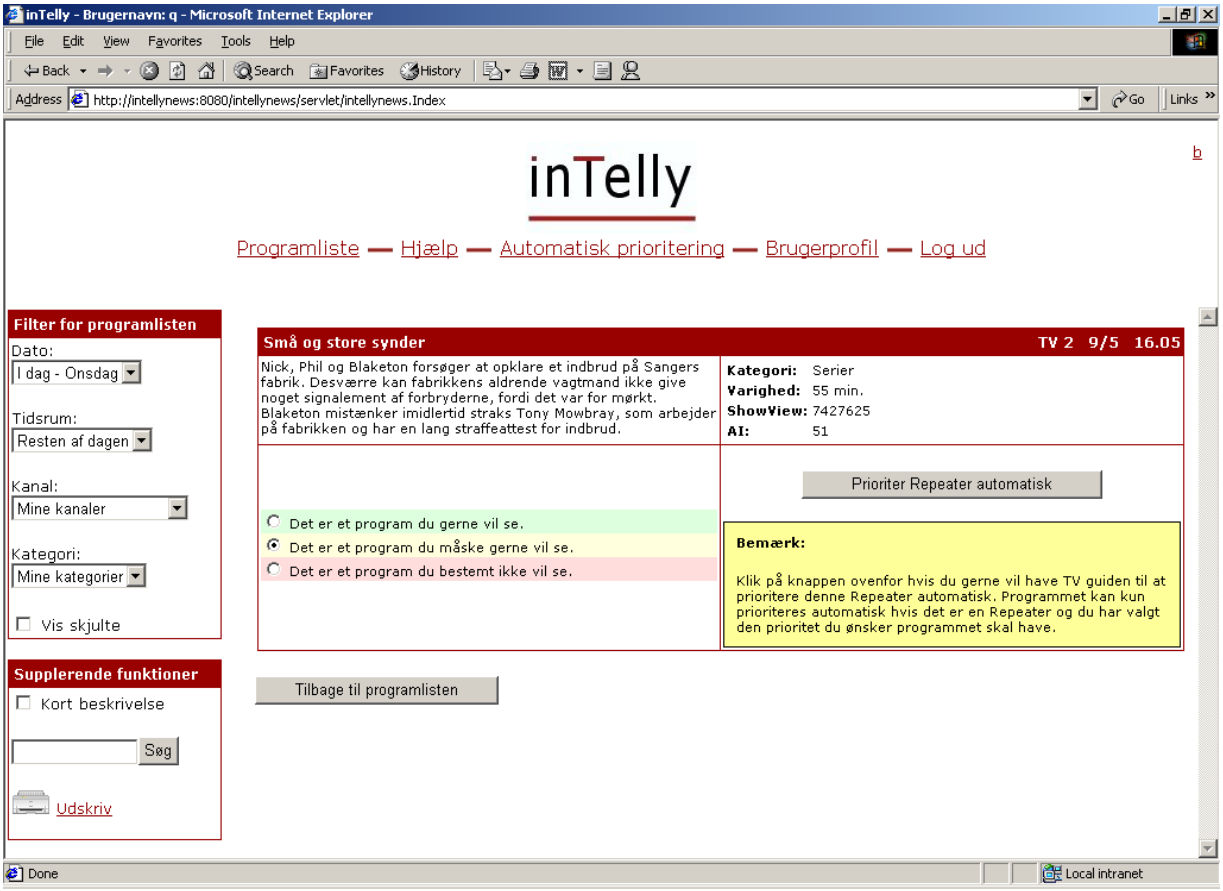

Figure 93 : The detailed description for a single program.

Repeaters

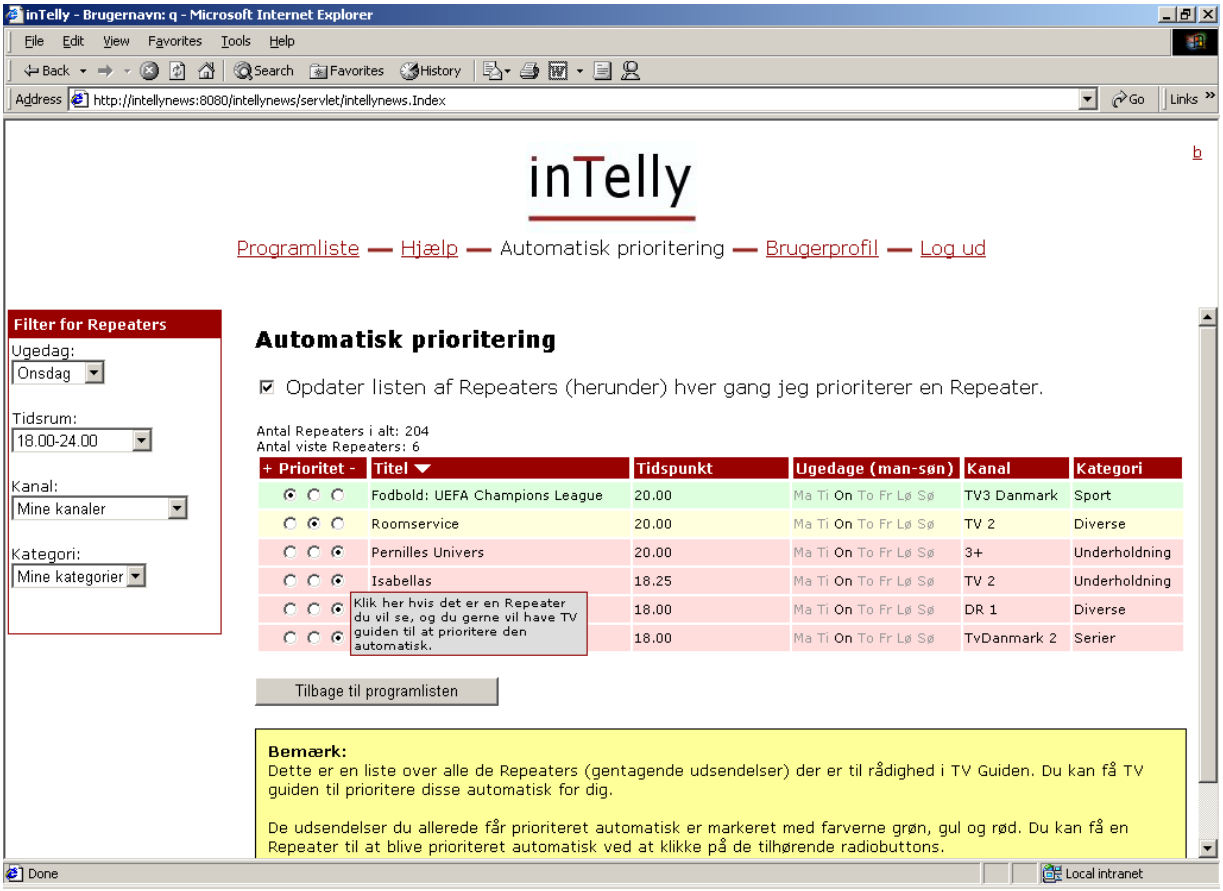

Figure 94 : The repeaters page, where the user can select/deselect repeating programs.

Help

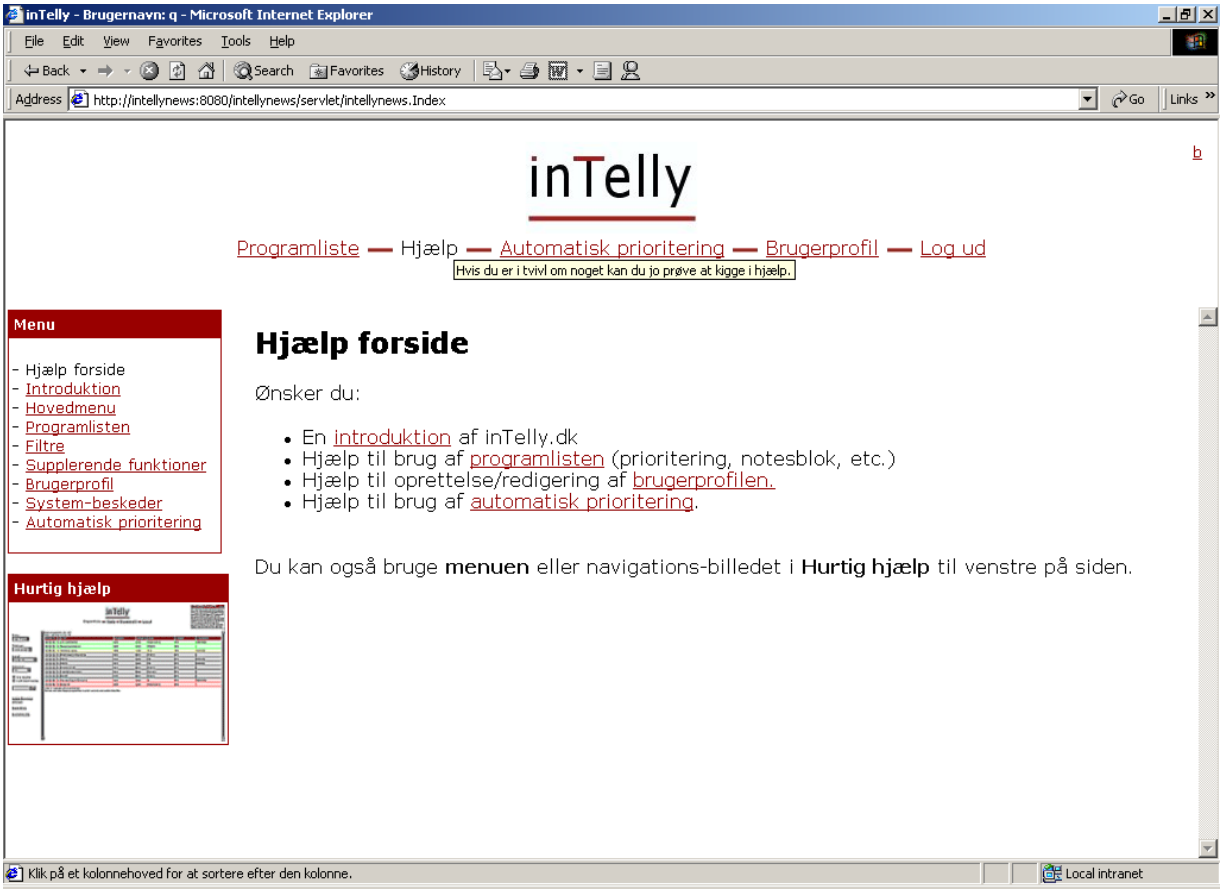

Figure 95 : The front page of the help pages.

Figure 96 : The introduction to the inTelly.dk TV-guide placed together with help.

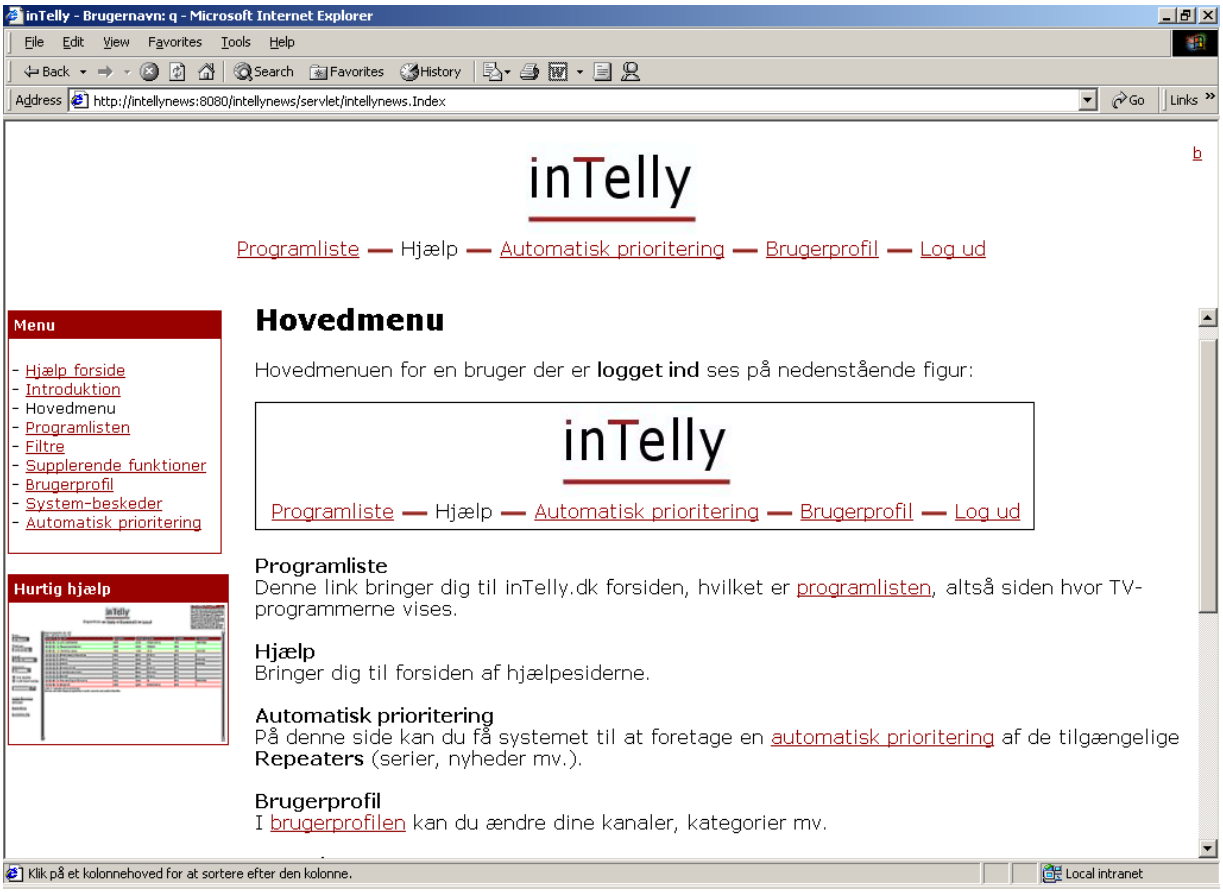

Figure 97 : The help for the menu bar at the top of all pages.

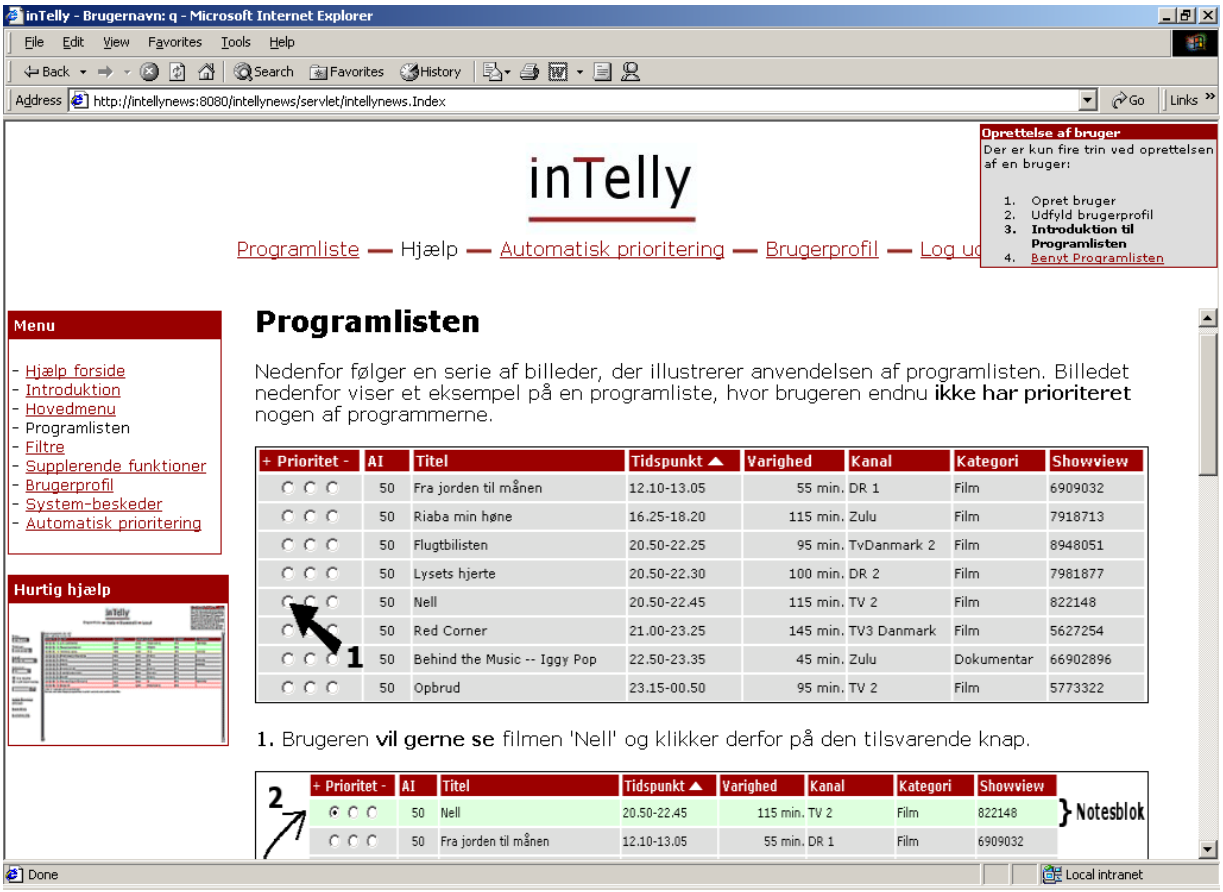

Figure 98 : The help cartoon for the program list.

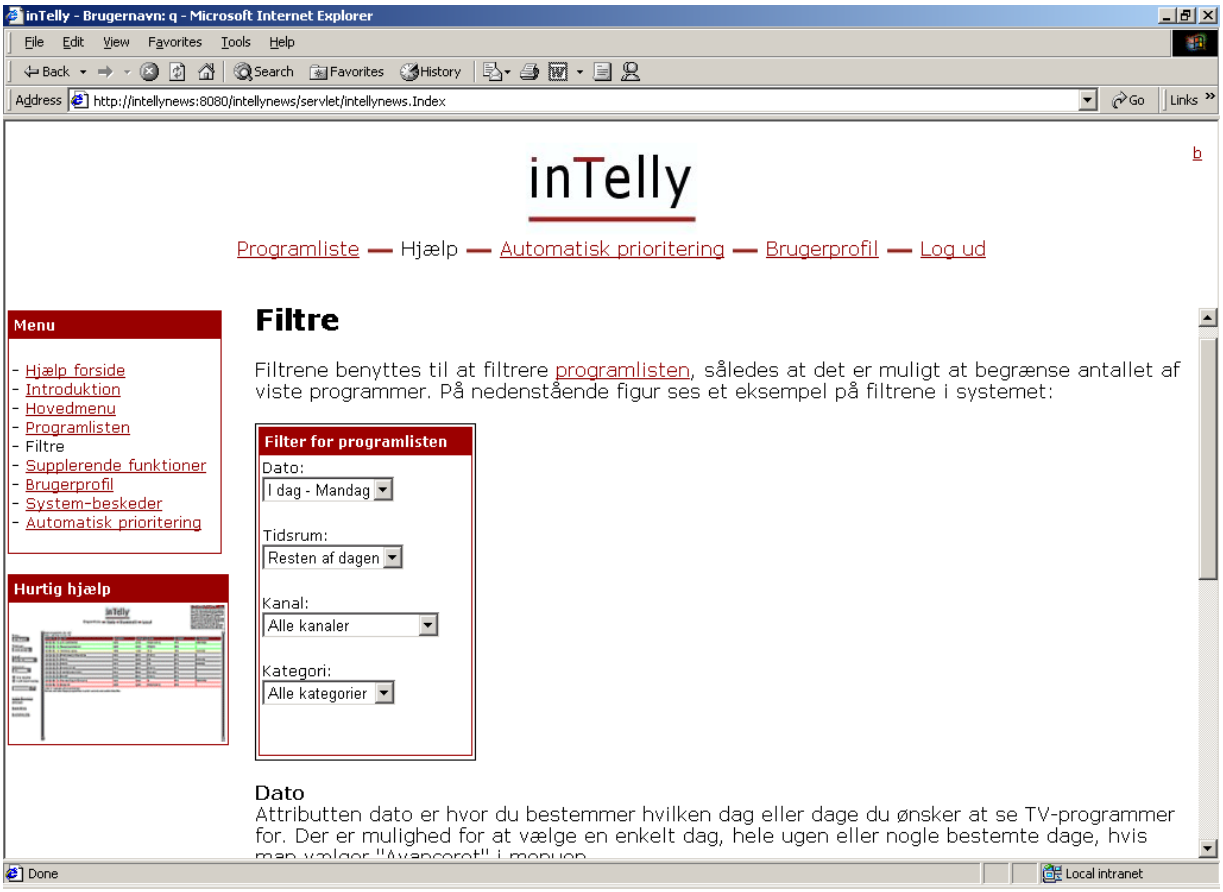

Figure 99 : The description of the filters.

Figure 100 : The help for the additional functionality.

Figure 101 : The presentation of the user profile and what it contains.

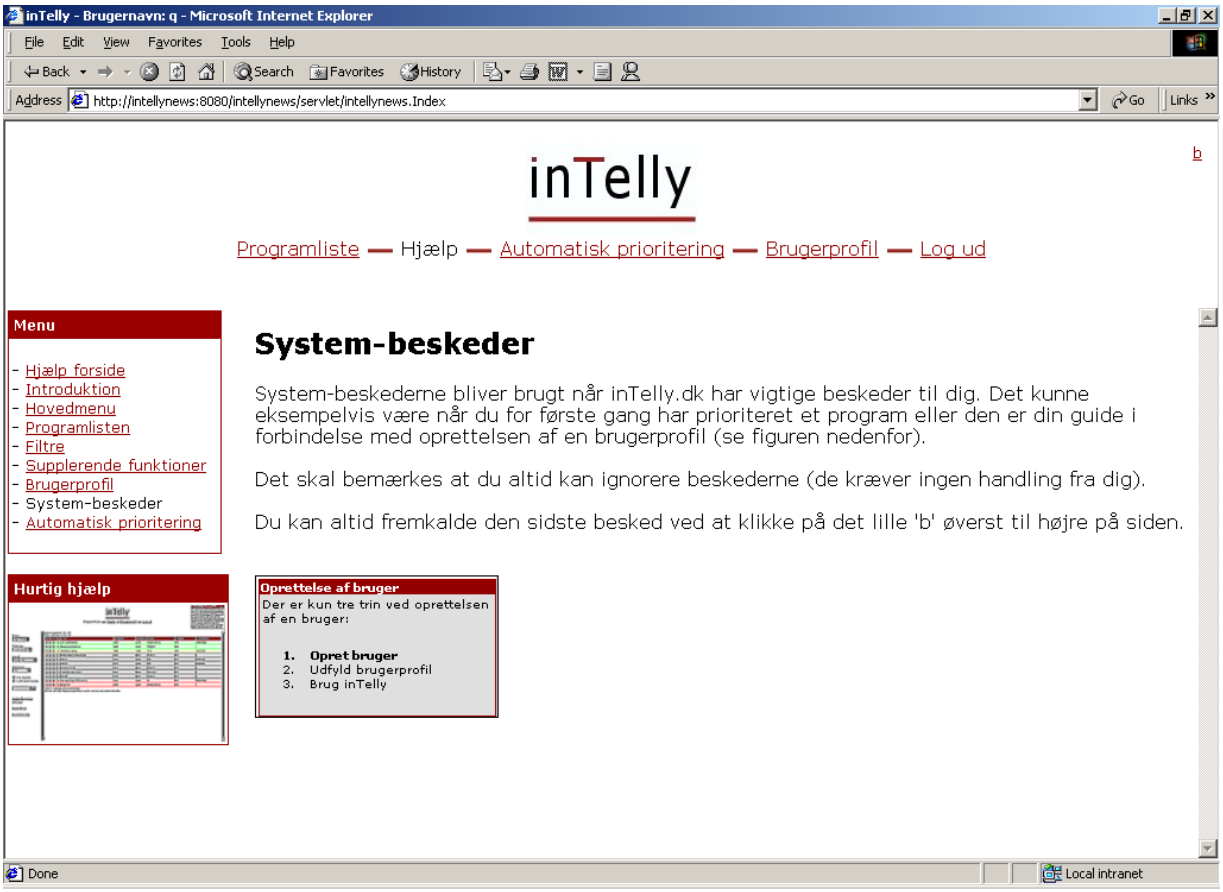

Figure 102 : A description of the system messages that pop-up when the user e.g. prioritises a program the first time.

Figure 103 : The help/description for the repeaters page.

Menu Bar and Logo

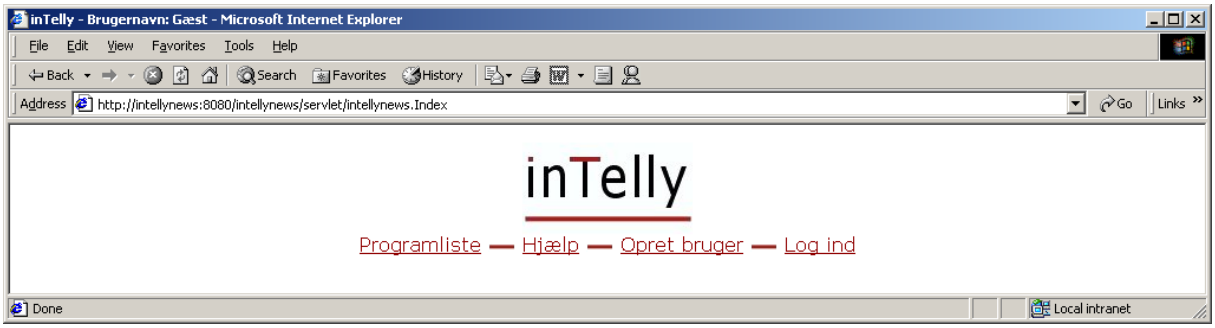

Figure 104 : The logo and menu bar for non-registered users.

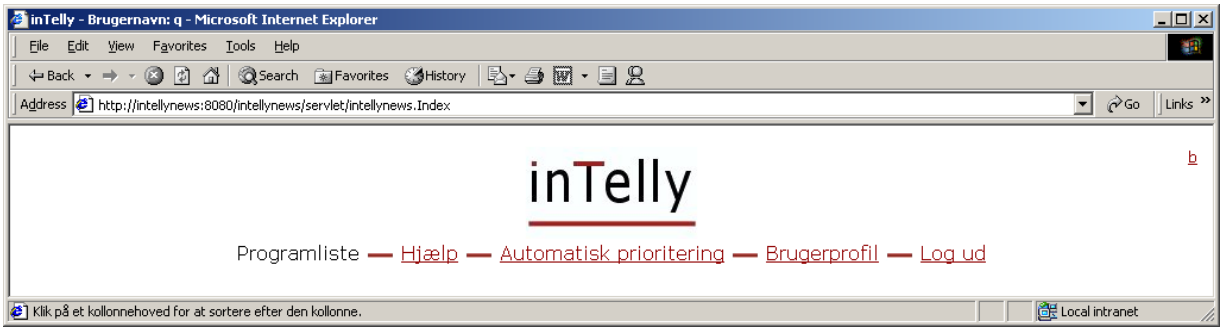

Figure 105 : The menu and logo presented to registered users.

Messages

Figure 106 : This is an example of the design of the messages in the system. The message here is presented to the when he/she creates a new user.