MASTER

On the construction of a personalized home media system using TV-Anytime packaging

Dekker, T.J.; Loef, E.

Award date:
2006

TECHNISCHE UNIVERSITEIT EINDHOVEN

Department of Mathematics and Computer Science

MASTER THESIS

On the Construction of a Personalized Home Media System using TV-Anytime packaging

by

T.J. Dekker & E. Loef

Supervisors:	Lora Aroyo (TU/e)
	Peter Hulsen (Philips)

Eindhoven, January 2006

Abstract

This thesis concludes the joined work of the authors in the ITEA Passepartout project. The ITEA Passepartout project envisions an ambient interactive personalized connected home environment. The project is executed by different companies and institutes all over Europe.

In this thesis we describe a concept of next generation television, by combining internet, optical discs and traditional broadcast. This concept is based on the TV-Anytime Phase 2 metadata specification, which describes of packages. Packages are the basis of the concept. A Package is a collection of content, which not actually contains the content, but has references to the content. The content can be on a broadcast network, ip-network or on an optical disc.

The concept is converted into an architecture for a system, that also performs personalization over the packaging concept. In the proposed system it is possible to have a personalized search for packages. This search extends the query with extra keywords, derived from a set of ontologies. The result of the query is filtered and ordered using a unique combination of personalization techniques. The combination of the techniques is chosen in a way that it uses the advantages of each techniques and covers the disadvantages of each personalization technique.
Based on the proposed architecture of the system a prototype is implemented. The intention of the prototype is to make a proof-of-concept, with the focus on personalization and packaged content.

Contents

Terms and Abbreviations

Blu-IS	Blu-Ray Information System
CRID	Content Reference Identifier
CWI	Centrum voor Wiskunde en Informatica, Amsterdam
DDL	Description Definition Language
DNS	Dynamic Name Server
DTD	Document Type Definition
DVD	Digital Versatile Disc
EPG	Electronic Program Guide
GSM	Global System Mobile
HDD	Hard Disk Drive
HDTV	High Definition Television
IPTV	IP Television
JDIC	Java Desktop Integration Components
jLirc	Java Linux Infrared Remote Control
JMF	Java Media Framework
Locator	Contains the location, a URI, of a content element
LTCS	Long Term Conceptual State
MDS	Multimedia Description Schema
MHP	Multimedia Home Platform
MPEG	Moving Pictures Expert Group
OWL	Ontology Web Language
P2P	Peer-to-Peer
Package	A collection of content items, which typically share a common subject. Defined by the TV-Anytime forum.
PDA	Personal Digital Assistant
PVR	Personal Video Recorder
RDF	Resource Description Framework
STCS	Short Term Conceptual State
TU/e	Eindhoven, University of Technology
UM	User Model
VHS	Video Home System
VLC	The Video Lan Client media player
XML	eXtensible Markup Language
URI	Unified Resource Identifier

List of Figures

Chapter 1

Introduction

In the year 1900 the the word "television" is used for the first time. 35 years later, the first (black & white) commercial broadcast starts. After the second world war, in 1954 the first electronic color television is commercially available. In most households (in the Netherlands), the concept of watching television has not changed till the year 2000.
In the last 5 years, every analog transport (like television and telephone) is steadily converting to *digital* transport (e.g. Voice over IP, IPTV). Also the cable companies in the Netherlands are converting their cable network for digital broadcasting. The advantages of digital broadcasting are that 8 digital channels take the same "bandwidth" as 1 analog channel, allowing for more channels to be sent to the consumer. Furthermore, the image quality is much better with digital television (less distortion). Because all televisions still have the analog tuner inside, the cable companies offer a converter, from digital to analog signal, which is also called a *set-top box.* Another advantage of digital television is the fact that it is possible to send little applications to the 'set-top' box. There are three well-known programming languages to create these applications: MHP (java based), OpenTV (c++ based) and Liberate (Web-based), where OpenTV is most commonly used today. The next step of digital television is (and this is already established in England by Sky+) that there is *interactivity* on the television. For example when there needs to be a vote (for programmes like pop-idol) from the viewers, the customer can vote for a person he likes in pop-idol, with his remote control.
A few years ago, next to the digitalization of the television, there was the introduction of the DVD. The DVD replaced the popular VHS-tapes completely. VHS-tapes also were very popular for recording purposes. Also this process is digitalizing slowly. At this moment the VHS video recorder is out-of-date. Dealing with VHS tapes, trying to fit 2 movies on 1 tape, is something of the past. No limit of the numbers of recording anymore, etc. With the introduction of the *Hard disc recorders* all these problems are solved. These new functionalities will also change the manner of watching television.
Using an Electronic Program Guide (EPG) it is easy to record a few hours of television, or use the new recording feature *"time-shifting"*, With Time-shifting it is possible to see a movie which is playing, while meanwhile the rest of the movie is recorded, or "pause" live television to pick up the telephone for example.
Because it is very easy to record a huge amount of programs, the probability that something is recorded the user likes, is high. So why should the user switching the channels, when there is already nice content on the Hard disk of the recorder. Currently there are three different types of these recorders.

- HD-Recorders(Hard Drive) of electronica fabrics like Philips, Sony and JVC. These devices can perform all mentioned tasks, but some of those devices are also have DVD-Recorders, to burn a program to a DVD-disc.
- Mediacenters. Computer companies have a so-called barebone computer, where recording

software on is installed. Such a mediacenter can do everything a HDD/DVD-recorder can, and more. A media-center can also play all movie-types and plays music files. The most commonly known media-center is the Microsoft Windows XP Media Center [82]. There are also different Linux-based media centers, like the KISS box [40] or Dreambox.

- At last also the cable companies like Essent and UPC are introducing digital television in the Netherlands. They offer a set-top box, which converts the digital signal to the conventional analog. There are set-top boxes, which have a built-in Harddisk drive. The set-top box can record like the other devices to the disc.

At this moment the "hot topic" for television (including all related devices) is HD-TV, High Definition Television. High Definition Television is nothing more than television in higher resolution, with better sound quality. To give a comparison, normal broadcasts (and DVDs) are about resolution 640x480. Where HDTV can be up to 1080p (which is a resolution of 1920x1080). In 2006 the world-cup in Germany will be broadcasted in High Definition format, which is the first time that a big event is broadcasted in HD. For High Definition television it is necessary to have a television, which supports this higher resolution, which implies for most customers that they have to buy a new television.
Next year also movies will be recorded in High Definition, to have a more detailed image, which enhances the movie experience. There is one problem, these HD movies won't fit on a traditional DVD, because it contains much more pixel data for each frame. Therefore there are two successors of the DVD. One is called the *"Blu-Ray Disc"* [38] and the other successor is called the *"HD-DVD"* [39]. These two standards tried to work together, but that failed. Now the market is going to decide which standard will be used. The introduction of the *"Blu-Ray-disc"* and *"HD-DVD"* will be in the first quarter of 2006.

In the PassePartout project we will go further than the current situation description. In our project we want to change the concept of watching television, by combining internet, disc and traditional broadcast.

1.1 ITEA

The ITEA programme is a European success story, but will end 31 December 2008. Its successor, ITEA 2, will be an even more ambitious, strategic pan-European programme for advanced pre-competitive R&D in software for Software-intensive Systems and Services (SiS).

ITEA 2 stimulates and supports projects that will give European industry a leading edge in the area of SiS (in which software represents a significant segment in system functionality, system development cost & risk, system development time).

ITEA's ambition is to mobilise a total of 20,000 person-years over the full eight-year duration, requiring a significant increase in investment level. This level of ambition follows from the experience in ITEA, the need to close further the gap in R&D investment (3% of GDP, Lisbon objective) and the ever growing importance of SiS.

ITEA works closely with other EUREKA projects and the Framework Programmes of the European Commission. ITEA projects are supported financially by all 35 countries in the EUREKA framework.

1.1.1 The Passepartout Project

1.1.2 Involved partners

Eindhoven University of Technology

The Eindhoven University of Technology (TU/e) is one of the leading universities of technology in the world (often ranked in the top-10). The TU/e offers master- and bachelor courses and teaches more than 5000 students. The Adaptive Web-Based Information Systems group, is part of the department of Computer Science. The supervisor of the group is prof. dr. Paul de Bra. The group is leading in the field of adaptive hypermedia and is a member of the PrLearn Network of Excellence. The group published the AHAM reference model for adaptive hypermedia-applications and developed the general-purpose engine for adaptive hypermedia AHA! This software is used in multiple research areas and adaptive courses in different institutes over different countries. The group maintains the Adaptive Hypertext and Hypermedia Homepage (http://wwwis.win.tue.nl/ah/) and the Adaptive Hypermedia Mailing List (ah@listserver.tue.nl), also. The TU/e was the host of the Third International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, in 2004. Together with Philips Research and other faculties of the University, the group is involved in the area of Ambient intelligence. the group is active involved into the research of user modelling for personalization and adaptation of content, navigation and presentation.

Philips

Philips worldwide Royal Philips Electronics of the Netherlands [78] is one of the world's biggest electronics companies, as well as the largest in Europe, with 160,900 employees in over 60 countries and sales in 2004 of EUR 30.3 billion. Philips is active in over 60 businesses, and with more than 115,000 registered patents, Philips is currently number 1 in the global markets for lighting, electric shavers and DVD recorders. Philips is number 2 in medical diagnostic imaging worldwide. Philips is currently active in three related areas: health care, lifestyle and technology.

Today Philips is recognized as a world leader in digital technologies for televisions and displays, wireless communications, speech recognition, video compression, MRI, external defibrillators, storage and optical products as well as the underlying semiconductor technology that makes these breakthroughs possible.

Philips has world-class solutions in lighting, medical systems and personal and domestic appliances where our substantial investments in research and development, design and new sustainable materials are critical to Philips' future success. The strength of Philips' global operations is reflected in the leadership position across all of the product categories and regions.

Philips Applied Technologies Philips Applied Technologies (AppTech) is a new specialized business within Philips, since January 2005. AppTech is involved in the application of new innovative ideas and technologies by Philips itself, the distributors, and the customers, Where Philips Research is delivering new ideas and concepts. Philips AppTech translates these ideas to processes, resources or products, by applying the newly researched technologies in practice, With 1300 employees divided over 11 locations world-wide, there is much expertise available to manage this transition fluently. Philips Applied Technologies is a combination of Philips Digital Systems Lab and Philips Centre for Industrial Technology. Both divisions have gained much experience in the past and achieved a good reputation in the field of applying technology.

Glastonbury Philips Applied Technologies is divided in a few subdivisions and clusters. Each cluster has its own tasks and responsibilities. One of these clusters is the Glastonbury cluster. This cluster is most of the time working on big European subsidy projects, which are being executed with collaboration with other companies and universities. The Glastonbury cluster exists of 5 employees and a variable number of students. Each employee works in general on multiple projects and uses a few students to support him in the project.

Stoneroos

Stoneroos [77] is a relatively small innovative company that creates interactive television concepts. Stoneroos believes that Interactive television is the future, They describe is as ”The perfect combination of the power of television, interactivity, more control and individual choice”. Stoneroos uses its position as a full solution provider for interactive television in The Netherlands as springboard for international growth. Stoneroos combines the skills of consultancy, concept development, project management, design and programming in one company.

V2_

V2_ [69], Institute for the Unstable Media, is an interdisciplinary center for art and media technology in Rotterdam (the Netherlands). Founded in 1981, V2_ is an organization that concerns itself with research and development in the field of art and media technology. V2_'s activities include organizing (public) presentations, research in its own media lab, publishing, developing an online archive and a shop offering products that are related to V2_'s area of interest.

CWI

CWI [83] (Centrum voor Wiskunde en Informatica), founded in 1946, is the national research institute for Mathematics and Computer Science in the Netherlands. CWI performs frontier research in mathematics and computer science and transfers new knowledge in these fields to society in general and trade and industry in particular.

1.1.3 ITEA Passepartout

Successor of the Jules Verne Project [42]. The pace in which society has been going digital has continued to accelerate. A key factor in this acceleration is now software technology. This project focuses on the convergence of digital systems and applications in home media-centers in compliance with the ITEA roadmap “The Road towards Convergence” thus matching the vision of industries, institutions, SME and government partners. It is expected that from this project new technologies will emerge that propel the European software industries on to convergence, over terminals and network towards the final goal of ambient intelligence. The Passepartout project is aimed at coupling home media-centers to home networks for rendering scalable content from high definition television (HDTV) to lower definitions in a seamless fashion. Integral to the content will be reactive access and interactivity of high-resolution graphics using ISO and W3C standards for object oriented TV. With the project's goal to make a step towards ambient intelligence through mass personalization of reactive content, implementation shall use the most practical elements of MPEG-4 and MPEG-7 with W3C standards such as SMIL and related content synthesis and syndication in XML. Implications will stretch far beyond infrastructure and basic services but will also affect content, human system interaction and engineering. Implementation will be based on content access using a PVR media-center as server to new generations of access networks, including Blu-ray optical storage and WIMAX wireless technology. These networks will support the creation of home media-centers that move beyond current STB and PVR-DVD players using MPEG-2 technology, to create true mass-customization device for family entertainment. With the goals of content packaging and personalization to match the cultural and linguistic needs of the states of the EU and their economies.

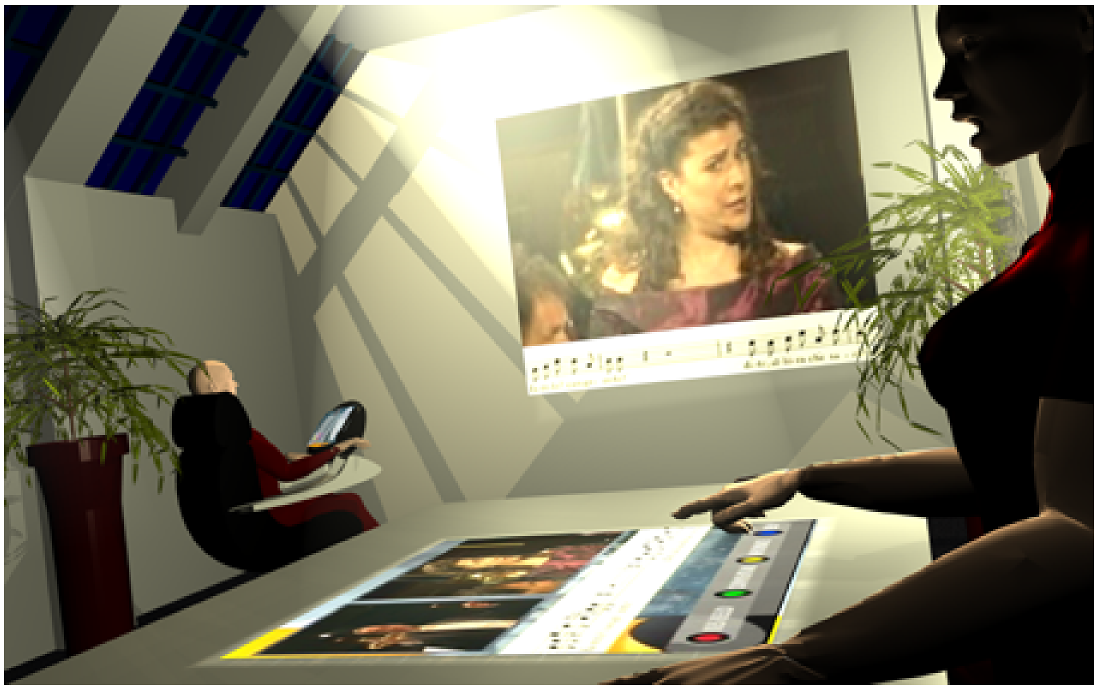


Figure 1.1: A visualisation of the Passepartout scenario

1.1.4 NL-Passepartout

NL-Passepartout represents the Dutch consortium within the ITEA Passepartout team. The entire ITEA consortium consists out of a set of international partners in which the Dutch partners are members. Core of NL-Passepartout within the Passepartout project aims at studying the automatic generating of personal recommendations using dynamically built user profiles. These user profiles are used and presented in such a manner that the user always preserves the control over his Mass-Customization box and is able to locate interesting content in the huge offer. The personal recommendations are used to make content accessible on either TV-screens, projectors, portable displays and possibly other media-players in a wireless home network. In this context some research will be done into the mechanisms of incremental authoring of content on top of existing broadcast content and the sharing of this extra content with others by means of Internet video-chat groups. The user profiles follow the user constantly and are automatically updated/adapted if the user changes its viewing preferences. Besides this the research also contains the study of standardized techniques as SMIL, MPEG4/7 and XMT as a basis for a new form of content-distribution and peer-2-peer sharing environments (P2P).

1.2 Maxima Scenario

This section describes a scenario to illustrate the goal functionality of our demonstrator. The setting is a family home somewhere in the EU, in the year 2010. The family is assumed to be financially well off, living in a region outside their original parental background. While they wish for the children to integrate with the local community and live and learn from their neighbors, they also value their heritage; linguistic, cultural and religious, to effectively communicate with distant relatives and friends. The family consists of the mother, father, a child about four years old, a deaf nine year old and a teenager. The parents are determined that the children should be effectively multi-lingual and multi-cultural, and will invest time to adapt the multimedia content in the home. They therefore act as media guides and to some extent teachers for their children, by selecting and adapting the content. Since the parents have immigrated to the region, they will have a different preference of content than the default local selection and they use their home media center to include also programmes from their original home area, e.g. for news, music and movies. They may also choose to alter the language or subtitles of the content.

As the family gathers for a movie night together, the home media center has suggested a movie that suits each family member's preference and interest. The mother has briefly scanned the story of the movie and discovered that the ending is in her opinion not suitable for the children. She therefore changes to an alternative ending. As they start the movie, they all together use a shared big screen. However, this evening the deaf child also includes additional sign language on his own small screen, although sometimes they use subtitles on the shared screen. The teenager needs to practice her second language and has therefore been asked by her parents to listen to an alternative language version with her headphones. Although they enjoy the movie together, the father also wants to follow a live soccer game broadcast, and therefore uses his own handheld screen to view this private video stream. The media devices in the home are all connected to the ambient connected home media environment.

1.3 Problem definition

Given the above scenario and project context, a lot of different research topics need to be addressed to realize the proposed situation. One topic is to investigate how the content in the system that we envision can be modelled. A recently emerging standard, which could be used for content modelling, is the TV-Anytime standard and has come under the attention of Philips, which have started the research to some extent. This technique, and it's alternatives, should be investigated and conclusions should be drawn on what content modelling technique is best used in our system and how this should be done. Another topic of importance in the envisioned system is what personalization techniques should be used to provide the mentioned personalized experience of the maxima scenario. These are the two critical goals in our graduation assignment. We will do research on these two topics and will propose a system design, which makes use of these content modelling and personalization techniques. Furthermore, we will implement and evaluate a prototype of this system. To summarize all of our research goals and questions:

- Investigate how the TV-Anytime concept can be used to model the content in our system, and investigate what the alternatives and the state-of-the-art content modelling techniques and metadata specifications are.
- Investigate how the TV-Anytime concept can be used in a system which is able to retrieve content from three different distribution channels: broadcast, IP and optical disc.
- How can an end-user of the system use this content modelling technique to create, modify, share, receive content using the three different distribution channels?
- Investigate what the state-of-the-art personalization techniques are and how we can create a personalized experience in our system using both the user's preferences and variables from it's environment as input.

- Design a system with the proposed content modelling and personalization techniques.
- Implement a prototype based on this design.
- Evaluate the prototype.

1.4 Document overview

The remainder of this document is structured as follows. First we will describe all the findings of our literature research in two separate chapters. Chapter 2 will describe how we can model the content in our system by using different interactive TV metadata and content techniques. Chapter 3 will then describe how we can achieve personalization by discussing user modelling strategies and by listing the state-of-the-art in personalization techniques. Then we will describe the design of our system in chapter 4, whereafter the implementation of this design and an evaluation of the prototype is discussed in chapter 5. Finally, we conclude with some conclusions and suggestions for further work. Note that parts of this work has been published in a paper by M. Björkman et al. [43] where the authors of this document are co-authors. The paper has been included in appendix E. Furthermore, some texts of this document have been used for the ITEA and NL-passepartout deliverable documents.

Chapter 2

Interactive TV metadata and content

2.1 Introduction to metadata

Metadata is generally defined as 'data about data'. The most visible parts of metadata are the attractors/descriptors or hyperlinks used in electronic programme guides, or in Web pages. This is the information that the end-user will use to decide whether or not to acquire a particular piece of content.

The metadata allows the end-user to find, navigate and manage content from a variety of internal and external sources including, for example, enhanced broadcast, interactive TV, Internet and optical discs.

Metadata is a very important issue for personalization, filtering and searching, because all these processes are dependent on the correctness, completeness and expressiveness of the description and classification of the content, the metadata. Metadata can roughly be devided into three different categories.

- Descriptive metadata
  Descriptive metadata is a description about the contents of the data. A description of what to expect when opening the content, i.e. a title, synopsis, etc.

- Technical metadata
  Content exists in a number of different media types (text/audio/image/video). Each content type has its own *technical* characteristics. An image for example has some technical attribute like width, height, resolution and size in bytes, while a video could have some attributes like encoding format and bitrate.

- Review metadata
  Reviews of the specific content from specific users. This can be reviews from different categories of users like experts, normal users, friends, etc.

2.2 State of the art: Metadata description languages

In every system where a user can retrieve and use data in all kinds of formats and styles it is need to have a sort of metadata which describes the specific chosen content element. Also when a system needs to recommend data to a user, it needs to be able to parse the data based upon this metadata before effectively sending it to the user. In this perspective we see that the format and style of the metadata is very important for both clarity for the user and performance of system. First we will start by listing a number of metadata description techniques. Afterwards we list the effective metadata specifications and standards which make use of the listed technique

Most *metadata specifications*, which are described in section 2.3, use an underlying *description language*. The common specifications are *XML*, and its successors *RDF* and *OWL*.

2.2.1 XML

According to W3C [9], the Extensible Markup Language (XML) is "a simple, very flexible text format derived from SGML, designed to meet the challenges of large-scale electronic publishing".

XML by itself is just a text format, it does not set any rules for the structure to use when describing metadata. Adding structure or rules to an XML file this can be achieved by means of a *DTD* (Document Type Definition) or *XML Schema* [10]. According to W3C, the purpose of a Document Type Definition is "to define the legal building blocks of an XML document." It defines the document structure with a list of legal elements. A DTD can be declared inline in an XML document, or as an external reference.

Like DTD's, XML Schema can define the structure, content and semantics of an XML document. However, an XML schema allows for stricter validation of an XML document; it can do everything a DTD can do and more. For example, a DTD can only define the type of the content within elements of an XML document as "string", where an XML schema can specify the required type of content within an element. Another advantage of XML Schema is that the constraints are specified in the XML language (instead of the DTD).

Although XML Schema can be used to define how metadata is represented in XML, it can also be used to describe equivalent, non-XML representations of the same metadata. For example, metadata could be encoded in a binary format for transmission or storage. XML in combination with a DTD or XML Schema is currently widely used (according to w3c) as a technique for description of metadata. XML is so established and well-known that it can be used in almost all circumstances. The only disadvantage of an XML Document is that it is very inefficient in bandwidth and storage space, because it is very verbose.

2.2.2 RDF

The *Resource Description Framework* (RDF) [11] is a framework for describing and interchanging metadata which is built around the XML [9] syntax. RDF is intended to provide a simple way to make statements about Web resources. RDF is based on the idea of identifying content using Web Identifiers (URIs) [22] and describing this content in terms of simple properties and property values. Each property's value is either another resource (specified by a URI) or a literal (a string encoded conforming to XML-specified syntax rules). In short, each RDF statement is a triple, consisting of the resource being described, the property's name, and the property's value. RDF triples can be linked, chained, and nested. Together, they allow the creation of arbitrary graph structures.

RDF however, provides no mechanisms for describing these properties, nor does it provide any mechanisms for describing the relationships between these properties and other resources. Thus, a user can make RDF statements without committing to a specific ontology. That is the role of the RDF vocabulary description language, *RDF Schema* [12].

RDF Schema defines a language on top of RDF that supports the definition process. By predefining a small RDF vocabulary for defining other RDF vocabularies, we can use RDF schema to specify the vocabulary for a particular application domain.

RDF Schema structures give sufficient information to allow basic queries regarding the concepts' semantics and their relationships in the application domain. Developers are working on a formal semantics for both RDF and RDF Schema.

RDF Schema defines classes and properties that may be used to describe classes, properties and other resources. It provides mechanisms for describing groups of related resources and the relationships between these resources. RDF Schema vocabulary descriptions are written in RDF. These resources are used to determine characteristics of other resources, such as the domains and ranges of properties.

2.2.3 OWL Web Ontology Language

OWL [8] supports in contrast to RDF *reasoning* over its properties. Through this OWL can inference new properties and links based upon present information. Some of the properties which take care of this behavior are: the *transitiveProperty*, *symmetricProperty*, *functionalProperty*, and so on.

OWL supports features related to equality and inequality with properties like *sameClassAs*, *samePropertyAs*, *differentIndividualFrom*, etc. These can for example be used to create synonym classes or to state that two individuals are different from each other.

Furthermore, OWL supports property restrictions, cardinality restrictions, versioning and annotation features, class axioms like disjunction, union, intersection, and many more features.

There are three different sublanguages of OWL to suit different groups of users. These are called OWL Lite, OWL DL and OWL Full, with OWL Lite being the simplest and OWL Full being the most expressive of the three. OWL Full can be viewed as an extension of RDF, while OWL Lite and OWL DL can be viewed as extensions of a restricted view of RDF.

OWL Lite is designed for users, who primarily need a classification hierarchy and simple constraints. OWL DL is designed for maximum expressiveness while retaining computational completeness and decidability. OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL Full is meant for maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right.

2.2.4 Conclusion: Metadata description language

All in all, we can conclude that OWL is the description language of choice, because it is the most expressive of the three description languages XML, RDF and OWL. However, it depends on the fact that there is a good metadata specification that uses this description language in order to be able to use it in the Blu-IS system.

2.3 State of the art: Metadata specifications

2.3.1 XML-TV

XMLTV [13] is a set of utilities to manage TV viewing. They work with TV listings stored in the XMLTV format, which is based on XML. The idea is to separate out the backend (getting the listings) from the frontend (displaying them for the user), and to implement useful operations like picking out your favorite programmes as filters that read and write XML documents.

At present there are backends grabbing TV listings for Canada, the USA, Britain and Ireland, Germany, Austria, Finland, Spain, the Netherlands, Hungary, Denmark, Japan, Sweden, France, Norway, and Romania.

There are filters to sort the listings by date, to remove shows that have already been broadcast, and a couple of programmes to organize your viewing by storing preferences of what shows you watch. There are a couple of backends to produce printed output.

The format used differs from most other XML-based TV listings formats in that it is written from the user's point of view, rather that the broadcaster's. It doesn't divide listings into channels, instead all the channels are mixed together into a single unified listing. Each programme has details such as name, description, and credits stored as subelements, but metadata like broadcast details are stored as attributes. There is support for listings in multiple languages and each programme can have "language" and "original language" details.

A simple example might look like:

```
<tv>
```

```
  <programme channel="bbc2.bbc.co.uk" start="20010829000500 BST">
  <title>The Phil Silvers Show</title>
    <desc>
      Bilko claims he’s had a close encounter with an alien in order
      to be given some compassionate leave so he can visit an old
      flame in New York.
    </desc>
  </programme>

  <programme channel="channel4.com" start="20010829095500 BST">
    <title>King of the Hill</title>
    <sub-title>Meet the Propaniacs</sub-title>
    <desc>
       Bobby tours with a comedy troupe who specialize in
       propane-related mirth.
    </desc>
    <credits>
      <actor>Mike Judge</actor>
      <actor>Lane Smith</actor>
    </credits>
    <category>animation</category>
  </programme>
</tv>
```

It is possible to get more complex than this example, giving information on channels (instead of the id “channel4.com”), picture and sound, subtitles, age ratings and more. But almost every feature is optional so you can start with something simple. The complete specification is described in the XMLTV DTD [13].

2.3.2 MPEG-7

The MPEG-7 standard, [15] formally named “Multimedia Content Description Interface”, provides a rich set of standardized tools to describe multimedia content. Both human users and automatic systems that process audiovisual information are within the scope of MPEG-7.

MPEG-7 offers a comprehensive set of audiovisual Description Tools (the metadata elements and their structure and relationships, that are defined by the standard in the form of Descriptors and Description Schemes) to create descriptions, which will form the basis for applications enabling the needed effective and efficient access (search, filtering and browsing) to multimedia content.

These Descriptors define the syntax and the semantics of each feature (metadata element); the Description Schemes specify the structure and semantics of the relationships between their components.

Through a Description Definition Language (DDL) we can define the syntax of the MPEG-7 Description Tools and thus allow the creation of new or the modification of existing Description Schemes.

The MPEG-7 standard has seven parts, each responsible for one aspect of the functionality:

- The systems component specifies the tools for preparing descriptions for efficient transport and storage, compressing descriptions, and allowing synchronization between content and description.

- The DDL specifies the language for defining the standard set of description tools (description schemata, descriptors, and data types), new tools, and the main parser requirements.

- Visual consists of schemata and descriptors covering basic visual features such as color, texture, shape, and face recognition.

- Audio specifies a set of low-level descriptors for audio features (for example, a signal's spectral, parametric, and temporal features) as well as high-level application-specific description tools (such as general sound recognition and indexing schemata used for instrumental timbre, spoken content, audio signature, and melody).
- Multimedia description schemes (MDS) specify generic description tools pertaining to multimedia, including audio and visual content. MDS covers the basic elements for building a description, the tools for describing content and relating the description to the data, and the tools for describing content on organization, navigation and interaction levels.
- Reference software provides the software corresponding to the tools defined in the standard (parts 3-5).
- Conformance specifies the guidelines and procedures for testing an implementation's conformance to the standard.

MPEG-7 provides a rather heavy-weight approach to describing the creation and classification properties of an audiovisual resource. MPEG-7's strengths lie in its ability to specify detailed media-specific formatting and encoding information, and its fine-grained descriptions of temporal, spatial and spatio-temporal components of audiovisual content and their associated audio and visual features.

2.3.3 MPEG-21

Today, many elements exist to build an infrastructure for the delivery and consumption of multimedia content. There is, however, no "big picture" to describe how these elements, either in existence or under development, relate to each other. The aim for MPEG-21 [16] is to describe how these various elements fit together. Where gaps exist, MPEG-21 will recommend which new standards are required. ISO/IEC JTC 1/SC 29/WG 11 (MPEG) will then develop new standards as appropriate while other relevant standards may be developed by other bodies. These specifications will be integrated into the multimedia framework through collaboration between MPEG and these bodies.

Thus, the vision for MPEG-21 is to define a multimedia framework to enable transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities.

A Digital Item is a structured digital object with a standard representation, identification and metadata within the MPEG-21 framework. This entity is the fundamental unit of distribution and transaction within this framework.

In practice, a Digital Item is a combination of resources, metadata, and structure. The resources are the individual assets or (distributed) resources. The metadata comprises informational data about or pertaining to the Digital Item as a whole or to the individual resources included in the Digital Item. Finally, the structure relates to the relationships among the parts of the Digital Item, both resources and metadata.

In MPEG-21 a User is an entity that interacts in the MPEG-21 environment or makes use of Digital Items. Such users include individuals, consumers, communities, organizations, corporations, consortia, and governments or even software agents. Users are identified specifically by their relationship to another User for a certain interaction. From a purely technical perspective, MPEG-21 makes no distinction between a "resource provider" and a "consumer"-both are Users. A single entity may use a resource in many ways (publish, deliver, consume, etc.), and so all parties interacting within MPEG-21 are categorized as Users equally. However, a User may assume specific rights and responsibilities according to their interaction with other Users within MPEG-21.

At its most basic level, MPEG-21 can be seen as providing a framework in which one User interacts with another User and the object of that interaction is a Digital Item.

Within MPEG-21 a resource is defined as an individually identifiable asset such as a video or audio clip, an image, or a textual asset. Hence, terms like multimedia content, media data, image, graphics, video, movie, visual content, audio data, speech content, etc. become unnecessary and should be avoided in the context of MPEG-21 when referring to resources with no specific context.

2.3.4 TV-Anytime

The TV-Anytime Forum [1] was established in September 1999 to develop open specifications designed to allow all interested parties to exploit the potential of high volume digital storage in digital television platforms. One of the main tasks that it set itself was the development of a metadata standard that would not only enable viewers to navigate through the growing television offer, but also meet the needs of advertisers (wishing to target appropriate messages to viewers) and of all the other actors in the television value chain (including content originators, broadcasters, network operators and equipment-makers).

For the purpose of interoperability, the TV-Anytime Forum has adopted XML as the common representation format for metadata. XML schema is used to represent the data model. TV-Anytime descriptions may however be instantiated in a format other than textual. TV-Anytime has described some of these mechanisms such as binary encoding for more efficient transmission of the XML format.

A metadata schema is the formal definition of the structure and type of metadata. TV-Anytime uses the MPEG-7 Description Definition Language (DDL) [15] to describe metadata structure as well as the XML encoding of metadata. DDL is based on XML schema as recommended by W3C. TV-Anytime uses several MPEG-7 data types and MPEG-7 Classification Schemes.

2.3.5 Metadata Conclusion

Each metadata specification has its own disadvantages and of course advantages. In this conclusion we give a summary of the main concerns of described the specifications. Finally, we will motivate which metadata specification we are going to use as basis for the proposed system.

The disadvantage of XMLTV is that it is purely oriented at *broadcasted* content, however for our system, and for the passepartout project, we also want to receive content via *IP services* and on *Blu-ray disc*. So if we want to use XML TV as a specification language for our content, we need to extend the specification to be able to use if for our purpose.
The main disadvantage for MPEG-21 (and MPEG-7)is that most companies and instances consider MPEG-21 (and MPEG-7) too intricate and elaborated.
TV-Anytime (phase 1) provides a very extensive metadata specification which allows for a very rich and detailed content classification and description. However, like XMLTV, this metadata specification is again oriented at broadcasted TV programmes. The specification needs to be extended to make it suitable for description of content that is located online or on a Blu-ray disc. A content structure which goes beyond a fixed linear time structure and allows multiple languages, alternative versions etc. put high demands on the content model. It needs to have a dynamic structure, rich metadata and suitable for various media. We believe that the recently emerging standard, TV-Anytime [1] phase 2 standard may serve as the basis in such requirements.

2.4 TV-Anytime metadata in detail

TV-Anytime defined a first metadata specification, which is called TV-Anytime *Phase 1* and after the completion of TV-Anytime Phase 1 the forum developed an extended version, which is called TV-Anytime *Phase 2*. Next we describe the key concepts of TV Anytime Phase 1 and its metadata specification and the extended Phase 2 specification. These specifications form the basis of the metadata structure which are used in the Blu-IS system.

2.4.1 TV-Anytime Phase 1

TV Anytime phase 1 [1] is a broadcast-oriented metadata specification. In TV-Anytime phase 1 each program (or more general: content element) is not specified by its physical location in the form of a URI, but it is uniquely identified by a "special" unambiguous identifier, a so-called *CRID* (Content Reference Identifier). This CRID needs to be resolved by a so-called *CRID-authority* which returns a *locator*, the actual location of the content element that the CRID points to.

Furthermore, a CRID can resolve into multiple other CRID's, which allows for grouping content items. The syntax of a CRID URI is of the form:

```
crid://<DNS name>/<data>
```

in which <DNS name> is a registered Internet domain name that takes the form of domain name described in section 3 of [20] and section 2.1 of [21].

<data> is a free format string that is URI [22] compliant, and that is meaningful to the authority given by the authority field. The portion of the field is case insensitive. It is recommended that all characters not within the range of characters allowed in a URI must be encoded into UTF-8 and included in the URI as a sequence of escaped octets. the *crid://* part of the syntax is case insensitive. The following is an arbitrary example of a valid CRID, which represents a CRID published by the "example.com" authority, with data part "exampledata":

```
crid://example.com/exampledata
```

Because of the dynamic behavior of the CRID identifier, TV-Anytime specified some scenario's to express the possibilities of CRIDs in combination with a PVR (Personal Video Recorder). In the TV-Anytime vision, each TV programme is uniquely identified by a CRID. The CRID of the program resolves into a *DVB-locater* which indicates the channel and time of the broadcasted program. To record that specific program, it is not the DVB-Locator that is used to program the event into the PVR, but the CRID of the program. The advantage is that when the locator changes, the PVR will notice this when it resolves the CRID. This will allow the publisher to change for example the date, time, length of the programme, while the PVR of a user who has scheduled this programme will still record the programme correctly. Furthermore it is possible to instruct the PVR to record a programme, even when it is unclear when the programme will be broadcasted. Given that a CRID also can resolve into other CRIDs it is possible to group content elements and have a "root"'-CRID, which resolves in all series of a programme. For example a publisher could decide to create a "Friends-CRID", which resolves into all CRIDs, which resolves into locaters of episodes of the TV programme "Friends". This "root"-CRID can be entered when using the timer function of the PVR, so that the PVR will automatically record every episode of the programme "Friends".

Next to the CRID technique for referencing content elements, TV-Anytime Phase 1 has a detailed metadata specification. The specification defines an XML Schema to describe broadcasted programmes. Each section of the XML schema describes a specific subject related to the broadcasted programme. In the section below a brief overview of the sections in XML format is given. TV-Anytime metadata files must have the following structure.

```
<TVAMain>
  <CopyrightNotice/>
  <ContentDescription>
    <ProgramInformationTable/>
    <GroupInformationTable/>
    <ProgramLocationTable/>
    <ServiceInformationTable/>
    <CastMemberInformationTable/>
    <ProgramReviewTable/>
    <SegmentationInformationTable/>
  </ContentDescription>
</TVAMain>
```

The Content Description metadata is divided into seven areas:

1. `ProgramInformationTable`
   Contains descriptions of content items, i.e. television programmes. Tags are included for the title of the programmme, a synopsis, the genres it falls under, a list of keywords that can be used to match a search, among others.

2. `GroupInformationTable`
   Contains descriptions of groups of related items of content, i.e. all episodes of "Foxes in the Wild".

3. `ProgramLocationTable`
   Contains descriptions of a certain broadcast of a programme. This can be used in an EPG or for automatic recording. The main purpose is to give an overview of all broadcasts on a specific channel.

4. `ServiceInformationTable`
   Contains information about a specific broadcast channel.

5. `CastMemberInformationTable`
   A mapping of cast members to unique identifiers. The identifiers can be used in other metadata instances making searching easier.

6. `ProgramReviewTable`
   Contains reviews for content elements.

7. `SegmentationInformationTable`
   Segmentation metadata is metadata about a specific segment of a certain programme. Based on this segmentation metadata it is possible to make different compilations of a (recorded) programme. A typical scenario could be the highlights of a soccer game or only the goals of a soccer match. It is also possible to have more segment metadata descriptions of one piece of content.

2.4.2 TV-Anytime Phase 2

The Phase 2 TV-Anytime Metadata Schema [6] is a backwards-compatible extension of the Phase 1 schema. It extends Phase 1 datatypes for content description and user description and makes use of imported datatypes from MPEG-21 to enable new areas of functionality. It also extends the TV-Anytime root document type, `TVAMainType`, to enable publication of metadata described using the new datatypes.

Furthermore, the TV-Anytime Phase 2 specification allows for content items to be grouped into *packages*. A Package is defined as a collection of *Items*, where an *Item* is a consumable entity. An item is made up of one or more *components*, which again can be selected depending on such criteria as:

- Terminal Capabilities
- User Preferences
- User Input

The data model of a package adopts the multi-level structure of the MPEG-21 Digital Item Declaration Language (DID) [19], i.e. *container-item-component* structure, along with some refinements and extensions.

The TV-Anytime Phase 2 defines a package to include additional content for a certain TV programme. Each component has a *resource*, which can contain any type of content (e.g. text, images, video, audio). The package doesn't contain the resource itself, but has a reference to the content (a CRID). A TV-Anytime package is defined in the XML-format. The XML-syntax of the

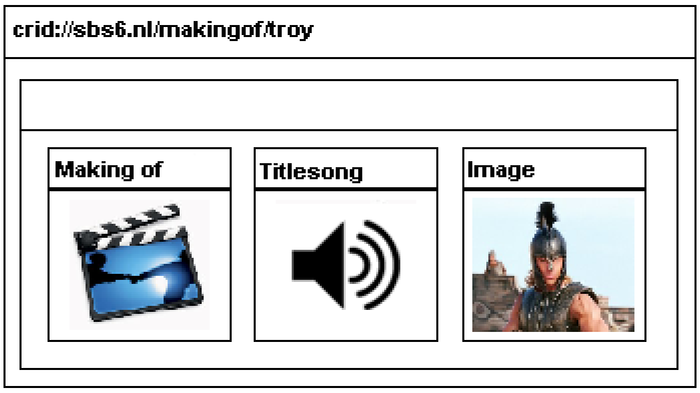


Figure 2.1: A Package as defined by the TV-Anytime Forum

package which is visualized in figure 2.1 is shown below.

```
<tva2:PackageTable crid="">
  <tva2:Item>
    <tva2:Component>
      <tva2:Resource mimetype="video/mpeg"/ url="" />
    </tva2:Component>
    <tva2:Component>
      <tva2:Resource mimetype="audio/wav"/ url="" />
    </tva2:Component>
    <tva2:Component>
      <tva2:Resource mimetype="image/jpeg" url="" />
    </tva2:Component>
  </tva2:Item>
</tva2:PackageTable>
```

This `PackageTable` description is the foundation of a package, but a package can hold much more than just a structure of "container-item-component". The TV-Anytime forum defines three types of relations between the components in a package [4].

- Selection Relation
  The selection relation indicates the relative importance among components when they are selectively consumed. These relations correspond to the set-theoretic relation defined in the `BaseRelationCS` of MPEG-7 MDS (Multimedia Description Scheme) [15]. This relation can be added to the package XML specification in the following way.

  ```
  <Selection select_id="Selection_Optional">
      <Descriptor>
          <Relation type="urn:tva:metadata:cs: SelectionCS:2003:Optional"/>
      </Descriptor>
  </Selection>
  ```

- Temporal Relation
  A temporal relation indicates the order of component consumption in time. The `TemporalRelationCS` has been designed based on the Allens well-known 13 temporal relations [24] and adopted the MPEG-7 `TemporalRelationCS`. The specified terms in the TV-Anytime `TemporalRelationCS` are as follows: precedes, meets, starts, overlaps, etc. The following example shows the instance of the precedes relation.

```
<Selection select_id="Temp_precedes">
  <Descriptor>
    <Relation type="urn:mpeg:mpeg7:cs: TemporalRelationCS:2003:precedes">
      <TemporalInterval>
        <MediaIncrDuration mediaTimeUnit="PT1N1000F">3000 </MediaIncrDuration>
      </TemporalInterval>
    </Relation
  </Descriptor>
</Selection>
```

The temporal relation needs to represent a quantitative amount of time of gap and overlap among components. A basic way to express this is to use absolute time itself, another way is to use relative time. As shown in the example, we adopted the MPEG-7 `MediaDuration` for the absolute time and `MediaIncrDuration` for relative time [15].

- Spatial Relation
  The spatial relation is devised by borrowing the MPEG-7 `SpatialRelationCS` and the topological relations from the MPEG-7 `BaseRelationCS` [15]. The `SpatialRelationCS` specifies a set of spatial relations along with their inverse relations: south and north, west and east, left and right, below and above, over and under, overlaps, touches, disjoint, and separated, etc. The following example shows the instance of the northwest relation. It expresses that the logo is northwest from the content of the first chapter in the presentation interface.

```
<Item>
  <Choice minSelection=1 maxSelection=1>
    <Selection select_id="Spatial_northwest">
      <Descriptor>
        <Relationtype="urn:mpeg:mpeg7:cs:SpatialRelationCS:2003:northwest"/>
      </Descriptor>
    </Selection>
  </Choice>
  <Item>
    <Condition require="Spatial_northwest"/>
    <Component>
      <Resource mimeType="image/gif" crid="http://www.example.com/image"/>
    </Component>
    <Component>
      <Resource mimeType="video/mpg" crid="http://www.example.com/video"/>
    </Component>
  </Item>
</Item>
```

Chapter 3

Personalization

3.1 Introduction

In chapter 2 the TV-Anytime metadata format and it's dynamic packaging structure has been described as a way to model the content in our system. The typical usage of the system we envision will give rise to a great number of users who will create, publish and share these kind of packages. When the system is being used like this, it is just a matter of time before there will be a huge amount of packages available in the system. Searching and selecting the desired package will be very time-consuming in this case and the user might be overwhelmed with packages that he is not interested in. Personalization techniques can help in solving this issue in several ways. With personalization, the system can reason on the actions that the user has performed in the past in order to provide the user with recommendations for content that the user is probably most interested in. The results of a user-initiated search for keywords can be personalized to reflect the user's interests. This chapter will describe how personalization can be achieved and gives a literature research on the state-of-the-art of personalization techniques. First we will describe how the system can model the data about a user, through user modelling techniques. After that, a set of personalization techniques are described, which show how the data in the user model may be used to realize personalization.

3.2 User modelling

The basis for creating a personalized experience that is offered by a personalized system is a good internal representation of the user(s) of the system. All relevant data about each user of the system is stored in a user model [49]. The information that is contained in the user model should be chosen in such a way that it provides enough data about each user to be able to use the personalization algorithms that are used in the personalized system. This is because information contained in the user model will serve as the input for these algorithms.

The personalization techniques that we have researched can be divided into two categories, namely *profile-based* and *context-based* techniques. To this end, we also need to store information on both the user profile and user context in our user model. The user profile contains personal information as well as information about the user's interests, like preferences for specific classifications, etc. The user context part consists of all relevant information about the user's current situation and environment, i.e. about the different devices, time preferences, audience information, etc. All this information is used as input for the filtering algorithms, which are described in section 3.3.

3.2.1 Usage of user models

The user model of a user is created when a user first interacts with the system. While a user is interacting with the system, all his actions are logged and information can be derived from these user actions to determine a user's interest in certain subjects. All derived information is used to *update* the user model. The more a user uses the system, the more information about the user will become available and the better the personalized experience will become. However, we need to make sure that the information we update the user model with is adequate and correctly reflects the user's interests. The SWALE project, [51] uses the following method to update the user model.

Firstly, the SWALE project uses the notion of two different *conceptual states*, in particular a *Long-Term conceptual state* (LTCS), which can be seen as the complete history of all user actions (acting as the user model), and a *Short-Term conceptual state* (STCS), which contains the actions of a user which have been performed during a single session. Now when a user performs an action, it is not instantly used to update the UM, but it is put in the STCS instead. After the user ends a session, the entire STCS is merged into the LTCS with the use of some rule set, thus updating the User Model. This is done to make the set of actions, which is used to update the user model, more meaningful when they are grouped as a session, which allows for a more intelligent updating of the user model.

So if we want to use this SWALE approach, the user model in our system needs to consist of three components: user *profile*, user *context* and *viewing history*. The last part of the user model, the viewing history, contains the history of a user's performed actions for a certain session (STCS). After a session, the information that can be derived from these actions is used to update the *profile* and *context* parts (LTCS) of the user model.

Cold start problem An issue that arises when the user model is created, and no updates have been performed yet, is that the user model does not contain any information about the user. This means that the system can not give a personalized experience yet, as it has to gradually learn about the user's behavior and interests. This is of course not what we want, as we would like to have a personalized experience as fast as possible. This problem is often referred to as the *cold-start problem* [50]. It is typically solved by having a user enter some of his personal data and preferences manually at registration time, to initialize the user model with a small data set. We will treat this cold-start issue and solutions to it in more detail in section 3.3 in the description of the state-of-the-art personalization techniques.

User modelling in the context of a television setting Furthermore, we can state that user modelling in a packaging/television environment is rather different than modelling in a web environment. The main difference lies in the fact that a television can be, and is likely to be, shared with multiple users. This gives rise to the creation of a user model that represents the information and preferences of multiple users correctly.

3.2.2 Common user models

To be able to apply personalization for a group of users (create a common user model), the group's preferences need to be modelled first. In addition to the user models for individual users, a *common user model* needs to be constructed to correctly represent the information and preferences of the group of users.

There are currently three well-known systems which make use of group modelling techniques. These are *MusicFX* [74], *PolyLens* [73] and *Intrigue* [75]. Based on known preferences of all individual users (existing individual user models), these systems try to compute preferences for a series of items that reflect the preference of the whole group, thus creating a common user model. The creation of such a common user model by the system can be done in two different ways.

1. Deriving a common user model based on individual profiles
   The system initially creates one user model for each user. When a group of users is interacting

with the system at the same time and want to receive a personalized experience that is targeted at the group as a whole, an algorithm is used to construct a common user model from these individual models based on the active group of users. This common user model is then used as input for the personalization techniques, instead of using only the user model of one group member. The system *learns* from the group's behavior by mapping derived information from the group's actions back to the individual user models, thus updating the individual user models. The common user model is not updated with new information, as it is computed each time the group of users require a personalized response from the system.

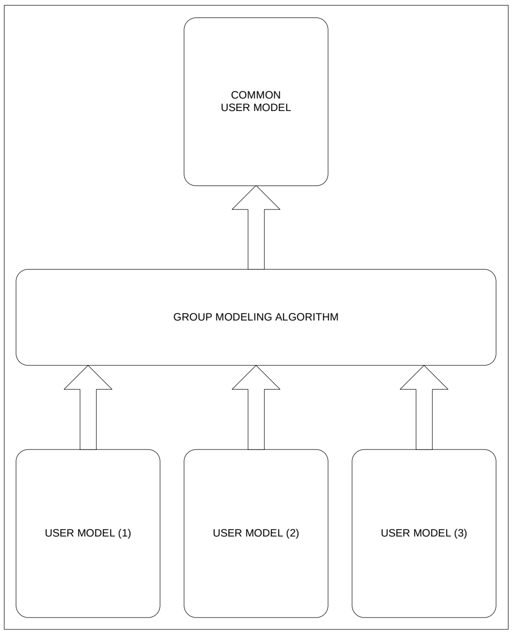


Figure 3.1: Deriving a common user model

[72] contains a series of approaches on creating such a common group user model, derived from individual user models, as well as a set of experimental results for each strategy. As can be learned from this paper, there are three social choice strategies that are most used by groups to come to a decision. The three strategies are explained in detail below. To clarify the strategies, a small example is included, which clarifies the differences between the three strategies with the calculations. The table below shows the individual ratings of a set of three users for three movies: Toy Story, Braveheart and Lord of the Rings. The given ratings are on a scale of 1 to 5, with 5 being highest and 1 lowest on the scale.

	Toy Story	**Braveheart**	**Lord of the Rings III**
User 1	+5	+1	+5
User 2	+4	+3	+2
User 3	+3	+4	+3

- Average Strategy
  The "average" strategy computes the average of the individual ratings of items from the group members and uses this average as a rating for the whole group.

	Toy Story	**Braveheart**	**Lord of the Rings III**
Calc.	(5+4+3)/3	(1+3+4)/3	(5+2+3)/3
Result	3,33	2,67	3,33

- Average without Misery Strategy
  The "average without misery" strategy works like the "average" strategy, but if an item is rated below a certain threshold for an individual, the item is not rated at all and should not be considered for selecting/viewing.
  This example uses threshold: 1,5.

	Toy Story	**Braveheart**	**Lord of the Rings III**
Calc.	(5+4+3)/3	**0**	(5+2+3)/3
Result	3,33	**0**	3,33

- Least Misery Strategy
  The "least misery" strategy computes the minimum of the individual ratings and uses this value as a rating for the whole group. The idea behind this strategy is that a group is as happy as its least happy member. *PolyLens* [73] uses this strategy to recommend movies to groups of users. The motivation of *PolyLens* to use this strategy is that it assumes that groups of people, who want to watch a movie together, tend to be small and a small group is assumed to be as happy as its least happy member. A disadvantage is that a minority opinion can dictate the group: if everybody really wants to see something, but one person does not like it, then it will never be selected.

	Toy Story	**Braveheart**	**Lord of the Rings III**
Calc.	↓(5,4,3)	↓(1,3,4)	↓(5,2,3)
Result	3	1	2

2. Create a separate common user model
   The system initially creates one user model for each user *and* one for each possible group of users. So in fact this method treats a group of users as an individual. When having identified the group of users interacting with the system, the corresponding common user model can be used to apply personalization. In this case, the user model of the group does not have to be computed every time the group of user's ask for a personalized response, as the common user models always exist. The system *learns* from the group's behavior by simply updating the corresponding common user model.

To illustrate the consequences of the different methods described above, we describe the tasks of creating and updating the user model for both methods.

Creating a common user model

Both methods need to construct a common user model, but method 2 needs to do this only once when the group of users first identifies itself to the system, whereas method 1 needs to compute the common user model every time a group of users starts a session. An advantage of the first method however is that upon creation of the common user model, the most recent preferences of the individual users are retrieved to construct the model, whereas with the second method, the group models are not updated with changing preferences of individual users because they are computed only once.

Updating a common user model

The second method allows for a very easy updating of the stored group model with the group's behavior, as the group is treated as a single person. This allows the system to update the group's preference as a whole rather than update the individual user models.

However, the issue that arises here is that the *individual* user models are not updated until a user is using the system alone. This might be frustrating if a user is watching with a group a lot of the time and then watches TV alone for one night. He is then faced with an immature user model, which does not represent his preferences as good as the group's user model. In a family situation, the process of watching TV alone will not occur that much, and individual user models will be very immature. In the optimal case, you want the system to update your user model when you are using the system, with or without a group, so that the system will get to know you when you are using the system.

The first method requires a more complex solution in updating the common user model, as this model is not stored permanently. This method needs to update the *individual* user models with the derived information from the *group's* actions. Watching with a certain group of other users introduces restrictions and extensions to a user's preferences [72]. This behavior should be mapped to the individual user models to allow the system to learn about that user's behavior in a group. However, this information may be hard to capture.

For example, a group of users can choose to take an action that one particular member of the group does not agree with. How can this event be captured? An additional complication is that an individual's ratings might depend on the group they are in. For example, a teenager might be very happy to watch a program with his younger siblings, but might not want to see it when he is with his friends. How do we know in this case if such a change in preference of a single user occurs because of the audience or if it is merely a change in the user's own opinion?

So the main question is: How can the system learn things about the behavior of individuals when watching TV with a group?

- Behavior in a group
  Analyzing the behavior of individuals in a group could be done by storing audience information together with the viewing behavior of a user in the individual user model. The system can then match the user's own preferences with the user's preferences in a certain group to determine the user's *influence* in that group. This allows the system to reason and learn on the user's behavior in a certain group. However, we face the cold-start problem here, as the system has to gradually learn this (complex) behavior. This could of course be solved by having the users set this group influence level explicitly: think of a parent who sets his children's influence level to a low value if they are watching TV with their parents.

- Deriving individual preferences
  An easy way out to derive the individual user's opinion of a group action could be that the thumbs-up and -down system [56] is used when watching TV with a group so that ratings can be stored for each individual user. For example if the system knows who is holding the remote control and pressing the button, that user's model could be updated with his explicitly entered choice. An individual user's liking of a program that is being viewed could be derived from sensors [69], which can detect how the user is feeling. When sensors detect a response in the user's emotion, an analysis can be done to identify if the user likes or dislikes the program. This way, the system can detect the user's preference independent of the group the user is viewing in.

Conclusions

So the difference between the two methods lies in the fact that with the first method all information is stored in individual user models, whereas with the second method information about the group of users is also stored, allowing the system to dynamically update the group user model as well. The first method however suffers from the difficulty of updating the individual user models and modelling the individual user's behavior in a group. The second method allows for storing and updating the group's preference in an easy way, but does not make the observations on the group's behavior reusable for the group's individual users and subgroups. For example, if a new member joins a group, all information that the system has on the subgroup cannot be reused.

So we can conclude that if a good solution can be found for updating individual user models derived from group behavior, method 1 is the method of choice. Method 2 is a simplistic alternative if no suitable solution can be found. Its simplistic view of modelling a group does not allow the system to reason on a user's behavior in a group, whereby it loses reusability of group behavior information.

If we look at the existing group-modelling techniques: *MusicFX*, *PolyLens* and *Intrigue*, we can conclude that none of these systems employ a method that updates the user models or the common user model with actions that the group takes. These systems only provide a group with recommendations derived from the individual user models and that is where their personalization functionality ends. There is no learning behavior in these group-recommender systems.

3.3 State of the art: filtering techniques

When moving from an analogue TV system to a digital system, which incorporates digital networks and a broadband IP-connection, hundreds of programs, channels and various other input sources become available for the viewer. If we want this user to be comfortable, we need to be able to filter all this information based upon the users preferences. In this way we can leave out everything the user dislikes and recommend what might interest him. To this end we have a number of content filtering techniques at hand [50].

Recommending systems are currently very popular in large e-commerce web sites. Potential customers are presented with highly relevant product recommendations based on their demonstrated likes; thus increasing the probability of purchase. Customers purchasing one or more items are presented with like-items increasing the likelihood of cross selling. And returning customers are retained by the quality of recommendations based on their previous purchases. In general, the goal of any recommendation system is to present users with a highly relevant set of items.

There are different approaches and techniques available to derive which content elements need to be displayed to a specific user. These personalization techniques can be split up into two categories: *profile-based* techniques, which will personalize based on the user's profile (personal information, preferences, etc) and *context-based* techniques, which will use information about the user's *environment* to personalize the experience.

3.3.1 Profile filtering

Content-based filtering

Content-based filtering [53] is a technique based on content analysis of previously viewed content in relation with the user's personal preferences. Every time a user performs an action, for example viewing a content item, the user model of that user is updated with certain characteristics of this action. If an algorithm then wants to filter newly available content items, it just has to compare the metadata of the new items with the stored preferences for this metadata found in the user model. The relevance of a given content item to a specific target user through content-based filtering is proportional to the similarity of the metadata of this item to the preferences for this type of metadata, which are stored in the user's profile. The higher the similarity, the higher this new item will be rated for recommendation.

Content-based filtering suffers from two major setbacks. Firstly, before a content-based filtering algorithm can work, a solid *content description* is required (metadata) for each content item. Although this issue can be solved by a metadata specification, the metadata for each content item needs to contain enough information to provide the content-based filtering with input.

The second issue that needs to be handled is the limited diversity problem. If the algorithm always is going to recommend items, which are similar to items liked before the risk arises of ending up in a limited region of the item-space. This is particularly important for new users who still might suffer from an immature user model/profile. Items, which might be liked by the user, but do not have any resemblance to previous items might never be selected.

Collaborative filtering

The *collaborative filtering* technique [54] is kind of like the successor of the content-based filtering and tries to provide an answer to its shortcomings. Where content-based filtering looks at the history of one user, collaborative filtering tries to look at the experiences and history of a population or *community* of users. By comparing the user model of one user to all the user models available, a user can be assigned to a certain community of nearest *neighbor users* which have similar interests. If a certain user in the neighborhood likes a certain program, there is a chance that you will also like it. All the users in the community act as recommendation partners for the target user. Through this technique items are recommended based upon user model similarities instead of item similarities.

The above described collaborative filtering technique is used by most webshops like "amazon.com" and is also implemented in the Tivo system [56], which is a commercially available set-top box published in the USA. This technique is called *user-based* collaborative filtering [58]. The similarity is usually computed using information about the items each user demonstrated interest in. Most recommender systems are based on ratings a user can give to items available in the system. Next to this technique, there is another collaborative filtering technique known as *Item-based* collaborative filtering [64]. This is not a probabilistic approach and is solely based on observed associations between pairs of items. This makes it possible to suggest related items to a user when he selects a certain content item. However, this technique does not handle unobserved items or features. This means that if the user selects an item that has not been a selection of any previous users, then the system will not be able to produce a recommendation. For an item to be part of a recommendation, it must have been associated with other items by one or more previous users. The advantage of item-based collaborative filtering is, that no user models need to be stored at a certain device.

Unfortunately both collaborative filtering techniques also suffer from a few drawbacks. An important first issue is the *cold-start problem* [52]. Here new *users* have not rated enough items to assign them in a valid community, through which it proves hard to make good recommendations. Another issue is the *latency* or *first-rater* problem [52] in which a new *item* is not available for recommendation until it is rated by enough users. As a final issue we also mention the problem that occurs when we have to deal with a strange user. The possibility exists that a certain users does not fit in any of the available communities, or cannot be compared with any other user in the system, thus excluding that user from recommendations.

To give a clear view of how a collaborative filtering process works, a detailed example is described as it is used in GroupLens [60], which is well-known movie recommendation system. This example uses 4 user models with its corresponding viewing history, which are shown in figure 3.2.

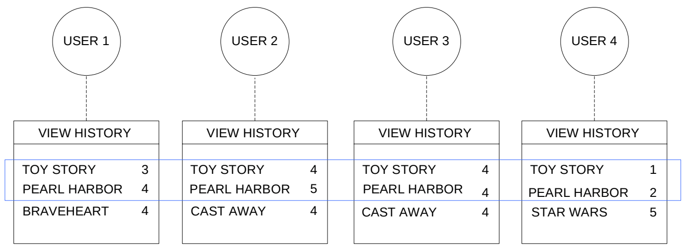


Figure 3.2: Determine collaborative filtering neighborhood

The first step in a collaborative filtering process is to determine the neighborhood of a specific user. This is done by calculating the similarity between the users. A commonly used algorithm for this is the Pearson-R algorithm [44]. Below the Pearson-R algorithm is displayed, where the similarity between user i and j is calculated. In this formula stands $\overline{s_i}$ for the mean rating of user i and s_{ik} for the rating of user i to item k. The variation of the correlation coefficient ranges from -1.00 till $+1.00$. The value -1.00 indicates a negative correlation and the value $+1.00$ indicates a positive one. The value 0 indicates an absent of correlation.

$$r_{ij} = \frac{\sum_k (s_{ik}-\overline{s_i})(s_{jk}-\overline{s_j})}{\sqrt{\sum_k (s_{ik}-\overline{s_i})^2 \times \sum_k (s_{jk}-\overline{s_j})^2}}$$

Applying this algorithm on the user 1 and 2 in described in the rating matrix 3.1 (and figure 3.2).

	User 1	**User 2**	**User 3**	**User 4**
Toy Story	4	3	4	1
Pearl Harbor	5	4	4	2
Braveheart	4	4	-	-
Cast away	-	5	4	5
Star wars	-	-	-	5

Table 3.1: Rating matrix

Calculate the similarity weight between user 1 and user 2 will be calculated as follows, taking all rated items that are rated by both users into account:

$\overline{s_i} = \frac{(3+4+4)}{3} = 3.67$
$\overline{s_j} = \frac{(4+5+4)}{3} = 4.33$

First calculate upper part of division:
$(3-3.67)(4-4.33)+(4-3.67)(5-4.33)+(4-3.67)(4-4.33) = 0.22+0.22-0.11 = 0.33$

Calculate $\sum_k (s_{ik} - \overline{s_i})^2$:
$(3-3.67)^2 + (4-3.67)^2 + (4-3.67)^2 = 0.45 + 0.11 + 0.11 = 0.67$

Calculate $\sum_k (s_{jk} - \overline{s_j})^2$:
$(4-4.33)^2 + (5-4.33)^2 + (4-4.33)^2 = 0.11 + 0.45 + 0.11 = 0.67$

These values result in the last calculation :
$\frac{0.33}{\sqrt{0.67\times 0.67}} = \frac{0.33}{\sqrt{0.45}} = \frac{0.33}{0.45} = 0.73$

Based on this similarity computation, User 1 and User 2 have a positive similarity of 0.73. That means that user 1 and 2 have a positive correlation. This calculation needs to be performed on the other users, to have a correlation coefficient between all other users. Based on the calculated similarity weights a neighborhood can be formed.
When the neighborhood is formed, there are two possible calculations possible.

- Predict a rating for an item
- Recommend a list of items

The prediction of an item is calculated by calculating a weighted average of all ratings of that specific item. Where $\overline{x}$ is the mean of all ratings of the user where the prediction is calculated for (the active user) and w_{ij} is the correlation between user i and j. $\overline{j}$ is the mean rating of user j and j_k is the rating of the item p:

$$prediction_p = \overline{x} + \frac{\sum_j (j_p - \overline{j}) * w_{ij}}{\sum_i |w_{ij|}}$$

A neighborhood of users is formed, based on the correlation coefficient. When a user is in the neighborhood or not depends on the implementations threshold. Above a certain value of correlation, the user is in the neighborhood of the active user. When the neighborhood is formed, each item that is rated by the neighborhood, will be predicted for the user. All predictions are sorted in one list and the above N items are recommended to the active user.

Demographic or Stereotypical filtering

Demographic or *stereotypical* filtering [65] tries to categorize users into certain groups or *stereotypes* to be able to initialize their user models and get rid of the cold start problem 3.2.1. The success of this technique depends on the categorization of the stereotypes. A few typical stereotypes are: housewife, child, businessmen, grandpa, etc.

These defined stereotypes are filled with stereotype-specific data possibly obtained out of population studies. For example the stereotype of 'housewife'. In the stereotype of the housewife, there will be some positive rankings for soaps and cooking programs whereas the stereotype 'businessmen' will probably like to watch the news and political debating programs.

In almost every system using stereotypes [65] it is common to 'ask' the user (when it is the user's first time of using the system) a few 'questions' about his preferences for specific genres, hobbies, etc. Based on these parameters the software indicates with which stereotype the user matches. This stereotype is then used to *initialize* his user model. The parameters are compared to all the stereotypes and the user will for each stereotype score a percentage of how well his profile matches the stereotype. For example a user's profile can fit with 60 percent "businessman" and 20 percent "grandpa". Based upon all these percentages the user will be fitted into a specific stereotype, which will cause him to receive recommendations that are assumed to be the best fit for his particular situation.

By using this technique the cold-start problem of content-based filtering can be tackled, as next to the standard information that the user answered on the questions asked at registration time, the user model is also updated with the stereotype-specific data obtained by the stereotypical filtering algorithm. This allows the system to instantly provide the user with some reasonable recommendations, without having to learn everything about the user from scratch. The same initialization can be put to use to bootstrap the collaborative filtering technique [63].

However, the main disadvantage in this filtering technique is its static behavior. The stereotypes are created by a designer at one point and are used until somebody enters new stereotype information by using new population research results, for example. Also when this information comes from population studies we can always assume that they do not cover the entire possible population and therefore there will exist people who do not match any of the stereotypes categories properly. This will initialize the user model for such a user in an inappropriate way.

Explicit filtering

To make the recommendation system self-learning the system must know if the user appreciates the proposed recommendations. This awareness can be achieved *implicitly* by observing what the user is doing after the recommendation took place. If the user watches the content that the system recommended, the system could conclude that the user liked the recommendation. Likewise, when the user chooses to watch content that the system did not recommend instead, the system could conclude that the given recommendation was wrong. Another way to receive validation information is *explicitly*, by asking the user if the provided recommendation is appreciated and useful.

Also the thumbs-up and -down system used in the TiVo System [56] could be put to good use in the explicit information filtering algorithm. In this way a user can communicate to the system whether he likes or dislikes a certain program. The biggest disadvantage of this method is that it is dependent on the user's input. The less accurate the input information provided, the less accurate the recommendations produced. This explicit validation information is the easiest way of receiving information on the user, but it can also be the most annoying for the end-user if he is constantly confronted with questions on his opinion of the content he is watching. Therefore, in designing an explicit filtering algorithm, unobtrusiveness is an important factor. The explicit filtering technique is best used for supporting the implicit filtering techniques to receive feedback directly from the user itself.

3.3.2 Context filtering

The problem with explicitly having the user rate content and with indicating initial preferences is that the user is often unable to express his preferences, does not know what his preferences exactly are or does not want to be bothered with thinking on what these exactly are. A better way to gather information for the user model is to reason on the user's preferences by observations on the user's actions, as described in the previous section. However, another important source, which can be used to reason on user preferences, is the *situation that the user is in.* We call this characterization of a user's situation the user's *context.* [67] provides a good overview and exact definitions of user context and context-awareness. It mentions four important aspects to reason about user context, namely *what*, *when*, *who* and *where.* These can then be translated to so called "primary" context types *activity*, *time*, *identity* and *location.*

The focus of the research about context filtering concentrates on three important aspects of user context in the design of the packaging system. These are the *different devices* the user has, *time* and the *audience* for which the system needs to apply personalization. These have been chosen based upon the most important "primary" context types as defined in [67], and because they fit best into the maxima scenario, which is the context of this packaging system. *Audience* (who), because the maxima scenario deals with a family, so it will be quite common that multiple users are interacting with the system at the same time. *Different devices* (what), because the household will have multiple devices through which users can interact with the system. *Time* (when), because this fits in with the "primary" context types mentioned in [67]. *Location* (where) has been left out in the design of our context personalization, because it requires the user or device to be fitted with a GPS locator or, in general, require the system to know where users or devices are, which is not really an important aspect in our maxima scenario. An outline is now given to describe how these aspects can be used to achieve personalization in the system and which issues this type of personalization will introduce.

Different Devices

In the current situation, it is common that a PVR is only connected to one display device (the TV) and that a single user interacts with the system with the use of a remote control. The Passepartout project aims to let multiple users interact with the system (the PVR) using a set of *different devices*, like TV, PC, PDA, GSM, etc. at the same time. However, a number of issues arise when the user is able to use different devices to interact with the system. Each device has its own capabilities to display certain content types. Content that is included in a package may not be suitable for the device that the user is working with. For example, the user has selected a package containing a movie content element in MPEG-4 format and the user is interacting with the system with a PDA that does not support this format. There are two options that the system has to solve this situation. It can either present an alternative piece of content, for example, images accompanied by a piece of text describing what can be seen in the movie. Alternatively, the system can simply omit content elements that the device can not handle from the package's presentation or refer to another device that is available in the user's environment that can be used to display the content.

Different devices also have some other limitations, like bandwidth, screen size, number of colors that can be used, interaction possibilities, processing power, etc. All these capabilities and limitations can be taken into account in presenting packages, content and recommendations on these packages and content to the user. This can be done by creating certain *presentation templates* [23], [66] for each device (TV, PC, PDA, GSM), which define what these capabilities and limitations of the device are and how to generate presentations for these specific devices. Using these templates, presentations can be adapted and personalized for any device type.

Furthermore, personalization can be used to track which devices are the preferred device for certain content types, for example HD-content is always being viewed on the user's big HD-TV screen. This way, the system can give suggestions or automate the process of selecting the most appropriate or preferred device for a certain user.

Time

The user's preferences can differ depending on the time of day or the day of the week. For example, a user may be interested to watch the news in the morning, but may want to view entertainment shows in the evening to relax. This observation can be used to create personalization features based on *time* context information. Different recommendations can be given, based on the time of day or the day of the week. Additionally, the amount of time the user has to watch TV can be analyzed to give appropriate recommendations. For example if the user has only 15 minutes to watch TV, recommendations for longer programs can be omitted, or programs can be adapted so that the length of the program is reduced. This user time schedule information could be retrieved from the user's agenda, which can be made available in the local home network so that the PVR has access to it. This information can then be used to give appropriate recommendations that fit into the user's schedule. Another feature that can be implemented with time-awareness is that of a reminder service. If a favorite program is about to start, the system can send a message to the user that a program of interest is starting.

Furthermore, an analysis of the moments that the user is using the system can be used to generate recommendations. If the system detects that the user likes to watch programs in the evening, then recommendations for programs in this timeslot can be chosen to have higher preference over other programs that the user might also like. To be able to do this, time information should be stored together with the user's viewing history in the user model.

Audience

All the methods explained above assume that the produced recommendations are for a single user. Sometimes it is possible that more than one person is using the system in which case the system needs to produce recommendations suitable for the entire group. In this case we can still use all the methods discussed above but with the extension that we now need to find recommendations based upon multiple user models together with viewing priorities or parental intervention. To do this, we need to be able to identify the users that are using the system. This identification can be performed automatically by the system, i.e. with the use of some token that the user is carrying, which the system detects. Alternatively, identification can be done manually by letting the user tell the system who is using it, i.e. by pressing buttons on the remote control [56]. Of course, the automatic detection is the method of choice, but requires more effort to implement or requires the user to carry a certain object with him. An additional complication is that a user may be in front of the TV, but is not effectively watching it and is merely sitting there to keep the other people company [70]. However, we will not go into detail on how this identification can be performed best, as this is in fact only a minor implementation issue in the design of the personalization techniques. In this section we focus on how the audience personalization can be designed best and assume that identification of the users is possible. Masthoff [72] discusses a set of techniques, which can be used to create an ordering for a list of content items, that is best for the group. These techniques can be used in our system to create an ordering in a list of recommendations that are made to a group. The techniques that the article discusses are all based on so-called "social choice" theories, which reason on how groups get to make a certain decision when the individuals of the group have different preferences. The conclusions of experimental outcomes of these techniques indicate that here does not seem to be a clearly dominant strategy, but strategies Average, Average Without Misery, and Least Misery (section 3.2.2) are all plausible candidates for implementation. The problem with existing group recommending systems is that they do not learn from the group's behavior after a recommendation has been given, like discussed in 3.2.1.

3.3.3 Hybrid Systems

Next to all of the techniques that have been described above, there exist systems [50], [58], [63] that use a combination the above techniques in order to benefit from all of the advantages of the individual techniques and to avoid their shortcomings. These *hybrid system* combine explicit,

techniques	**advantages**	**disadvantages**
content-based filtering	predictions for new content items	cold start, tends to suggest only similar items
stereotypical initialization	avoids cold start	static behavior
explicit filtering	direct user input can improve personalization	limited by user actions
collaborative filtering	predictions based on popular items in community	cold start, no predictions for unrated new content items

Table 3.2: Advantages and disadvantages of filtering techniques

stereotypical, content-based and collaborative filtering all in one system. Table 3.2 shows the main advantages and disadvantages of the different personalization techniques. We can see that the combination of content-base and stereotypical initialization can be useful to avoid (part of) the cold start problem. Furthermore the system can learn more from the user if he is able to explicitly indicate some of his preferences. Finally, adding collaborative filtering proves to be a good way to add recommendations for popular items in the community. We envision a system which contains context-based personalization techniques as well to further improve the personalized experience. We can conclude that such a hybrid system can overcome a lot of the shortcomings and can improve the overall personalization result.

Chapter 4

Design

4.1 Introduction

Now that we have described all theory derived from the literarate available, that is necessary to understand the basic workings of content modelling and personalization techniques, we can proceed with the creation of the design of the personalized system, which will model the content using TV-Anytime in a dynamically packaged structure and personalizes the experience of using this system , which makes use of a hybrid personalization approach, where the experience is personalized using both user preferences and variables from his environment. Firstly, a description of all extensions that we have made to the TV-Anytime specification in order to enable it for usage with, next to the intended broadcasted content, content from IP locations and content on Blu-ray optical disc. Secondly, we will give an overview of the workings of a system that uses this type of content by giving a practical approach to the theoretic TV-Anytime specification. Thirdly, we will present the approach that we take to enhance the packaging system with personalization techniques. Finally, we present the system architecture of the complete system in a formal way.

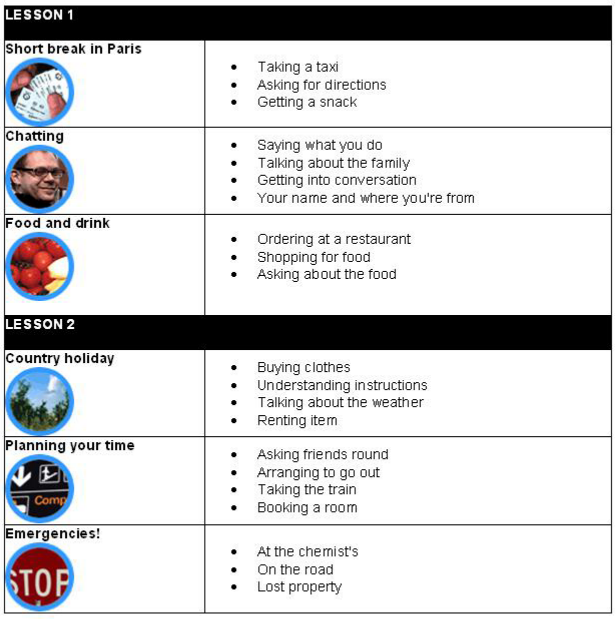
4.2 Packaging system

4.2.1 Introduction

The TV-Anytime Phase 1 specification described in section 2.4.1 and the TV-Anytime Phase 2 specification described in section 2.4.2, form the base specification of the Blu-IS system. In the Blu-IS system there are some changes and extensions applied to the original Phase 1 and Phase 2 specifications. In this section the concept of the Blu-IS platform is described, using a french course as an example in the packaging environment of TV-Anytime. We will then start with a short recap on the definition of the packaging concept, whereafter we describe how a packaging system can work.

4.2.2 French Course

Our system will make use of packaged content. To illustrate what a package exactly is, a French course from the BBC website (http://www.bbc.co.uk/languages/french/lj/menu.shtml) is used as an example of such a package. To get a good understanding of the content that is going to be used, a complete overview of the French course is outlined in figure 4.1.



LESSON 1	
Short break in Paris	• Taking a taxi • Asking for directions • Getting a snack
Chatting	• Saying what you do • Talking about the family • Getting into conversation • Your name and where you're from
Food and drink	• Ordering at a restaurant • Shopping for food • Asking about the food
LESSON 2	
Country holiday	• Buying clothes • Understanding instructions • Talking about the weather • Renting item
Planning your time	• Asking friends round • Arranging to go out • Taking the train • Booking a room
Emergencies!	• At the chemist's • On the road • Lost property

Figure 4.1: Contents of the French Course Package

4.2.3 Packaging

Introduction

In this section we propose some *extensions* to the TV-Anytime specifications, which are used as a foundation for the content modelling in the Blu-IS system. The “running example” of the French Course is used every time in the descriptions, to clarify how everything is related.

We extended the TV-Anytime specification, because we envision a system that combines internet, broadcast and optical discs. The original specification is much more broadcast orientated. To enhance the combination of internet, broadcast and optical disc, it was necessary to extend the original concept of packaging.

Packaging in Blu-IS

The basis of the Blu-IS platform is a *package*. In the Blu-IS concept the usage of a package is rather different. A package is still a collection of CRIDs (Content Reference Identifiers), which are related to each other, yet the content elements described in a package are not stored in the package itself, but are referenced using the CRIDs. The definition and usage of a (Blu-IS) CRID will be explained in section 4.2.3. The actual data model of a package is still the same as defined in section 2.4.2. The package in figure 4.2 is an illustration of the example package of section 4.2.2. The package is, just as the provided content for the French course, divided into two different lessons. Each lesson consists of content, which is referenced using a CRID.

The language used to describe a package is XML[9], which has many advantages for interoperability. Part of the XML representation of the French course package is represented in Figure 4.3. The XML snippet gives a short overview of how a package can be physically represented. The complete XML code of the package can be read in Appendix A. In this XML snippet it is obvious that there is no content in the package, but only references (CRIDs) to the actual content.

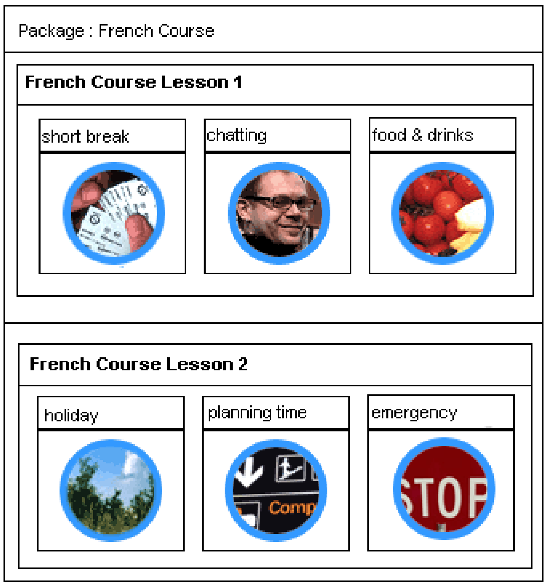


Figure 4.2: Visualization of a package

```
...
 <tva2:Item>
  <tva2:Component>
    <tva2:Resource crid="CRID://frenchcourse.com/chapter1/shortbreak"/>
  </tva2:Component>
  <tva2:Component>
    <tva2:Resource crid="CRID://frenchcourse.com/chapter1/chat"/>
  </tva2:Component>
  <tva2:Component>
    <tva2:Resource crid="CRID://frenchcourse.com/chapter1/food"/>
  </tva2:Component>
</tva2:Item> ...
```

Figure 4.3: XML Snippet of the example package

CRIDs

The CRID concept as described in 2.4.1 is technically unchanged, but the concept of how the CRIDs in the package are used is changed. In the TV-Anytime phase 1 specification the purpose of the locators is to point to a TV programme on a specific broadcast channel. In the Blu-IS system it is also possible to have CRIDs resolve to locators that contain URLs, DVB locators, or even references to a (Blu-Ray) disc. An example of this is shown in figure 4.5, where the *CRID resolving tree* of the CRID “CRID://frenchcourse.com/chapter1/shortbreak” of the “French Course” example is displayed. In the next section the content retrieval process is described in detail for the newly defined CRID resolving process.

Content Retrieval

CRIDs need to be *resolved* to get the locator(s) for the content. These locators point to the actual content. The CRID authority (specified in the CRID itself) handles the resolving process of the CRID. The response of a CRID authority on a CRID resolving request, made by the end-user, is an XML document containing a listing of all CRIDs and locators that the CRID resolves into

(Figure 4.4).

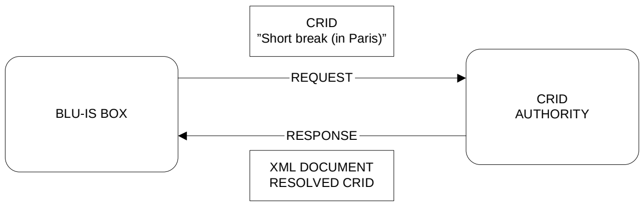


Figure 4.4: CRID resolving process

Locators

A CRID can resolve to other CRIDs and to locators. Like a CRID, a locator does not contain the content itself, but is a reference to the content. The locator can point to content, which can be obtained from the three different distribution channels (Broadcast, IP and Storage). A locator can point to a broadcast stream with this kind of syntax:

```
<Locator>dvb://1.4ee2.3f4;4f5@2001-03-27T18:00:00.00?:00</Locator>
```

A locator can also point to a website (for example "http://www.tue.nl") or a streaming media source with start and end time in the following way:

```
<Locator>http://www.tue.nl</Locator>
```

Furthermore a locator can point to content on local storage:

```
<Locator>file://2005/series/gtst.avi</Locator>
```

or disc:

```
<Locator>'CDDB-identifyer'/lordoftherings.avi</Locator>
```

Always use the CRID to point to the desired content. Never use the locator, because the locator is static. When there is a change (i.e. in the location of the content), the locator needs to change to point to this new content. However, when using a CRID, the same CRID can be used, with only the resolution process changed at the CRID Authority so that a different locator is retrieved.

In the French course package, a lesson consists of different topics. In this chapter one specific topic: “Short break (in Paris)” will be used in detail. The content in this topic is referenced by a CRID. As described before, a CRID can resolve into locator(s) or other CRID(s).

As can be seen in the introduction of the French course, the topic “Short break (in Paris)” consists of different sub-topics: “Taking a taxi”, “Asking for directions”, “Getting a snack”. These sub-topics will also be referenced using CRIDs. Some topics contain multiple content types. The following table describes what the contents of the topic “Short break (in Paris)” are and what content types there are for each sub-topic.

The CRID resolving tree for the “Short break (in Paris)” topic, containing the locators for this content, is visualized in figure 4.5.

This tree can be represented using the XML Schema specified in [3], Appendix A.1.1. To give an impression of how such an XML document should look like, an XML snippet of the complete XML document is presented in Figure 4.6. The complete XML document can be found in Appendix B.

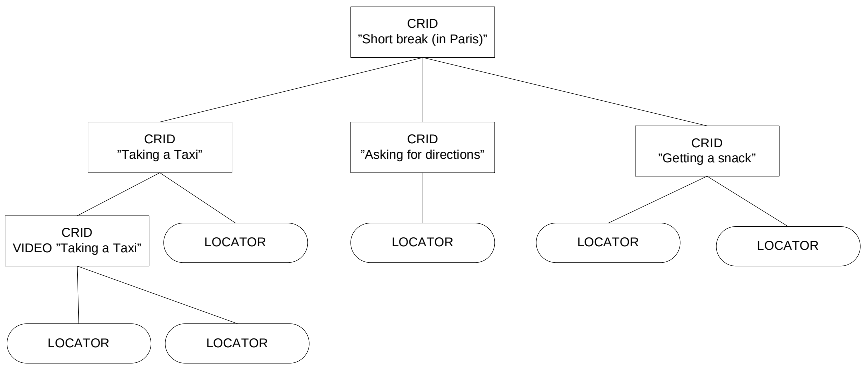


Figure 4.5: Example of a CRID representation of "Short break (in Paris"

```
 <Result CRID="crid://frenchcourse.com/chapter1/shortbreak"
status="resolved"
    complete="true" acquire="all">
    <CRIDResult>
        <Crid>crid://frenchcourse.com/chapter1/taxi/</Crid>
        <Crid>crid://frenchcourse.com/chapter1/askfordirection/</Crid>
        <Crid>crid://frenchcourse.com/chapter1/taxi/gettingasnack</Crid>
    </CRIDResult>
</Result> <Result CRID="crid://frenchcourse.com/chapter1/taxi"
status="resolved"
    complete="true"  acquire="all">
    <CRIDResult>
        <Crid>crid://frenchcourse.com/chapter1/taxi/</Crid>
    </CRIDResult>
    <LocationsResult>
        <Locator>dvb://1.4ee2.3f4;4f5@2005-08-27T18:00:00?00</Locator>
    </LocationsResult>
</Result>
```

Figure 4.6: XML snippet for the CRID resolution of the 'short break in Paris' topic

Summary

To summarize this section, retrieving content in a packaging system is a three-step process:

1. **Retrieve the package**
   The XML-file of the package needs to be retrieved to the user's platform, so that the user can read it.

2. **Resolve the CRIDs specified in the package**
   CRIDs will be resolved by their authority and will result in other CRIDs (which can be resolved in the same way) or locators. Once the locator is retrieved, the location of the actual content is known.

3. **Retrieve the content**
   Depending on the distribution channel, where the content is located, the content can be retrieved to the user's PVR, so that the user is able to view or record the content.

CRID authorities

CRID authorities are very important in the packaging system. An important issue on CRID authorities arises when end-users can create, modify and share packages (which contains CRIDs) themselves. The main problem in the case an end-user is creating a package, is the question where the end-user can obtain CRIDs to point to the content.

4.2.4 Metadata Service

A metadata service [7] is an online service, that provides metadata for a specific CRID. This service is needed because the CRID has no metadata specification. To know where the CRID is pointing to, without resolving it, it can be queried by a metadata service. Metadata retrieval occurs when a client wishes to obtain certain metadata from a metadata service that it has previously discovered and obtained a capability description for. The following list gives some examples of metadata retrieval.

- A client wishes to obtain program reviews for a CRID. The client sends a request specifying the CRID and type of metadata required, and the metadata service responds with the appropriate `ProgramReviewTable`.
- A client wishes to obtain the schedule information for a particular channel over the next week. The client sends a request specifying the channel, date range, and type of metadata required. The metadata service returns a `ProgramLocationTable` and `ProgramInformationTable` corresponding to the programs on that channel.
- A client wishes to search a metadata service that specializes in movie information. The client sends a request specifying the type of movie (i.e. the genre is "Western", and the star is "John Wayne"), and the type of metadata required. The metadata service returns a number of matching movies, using a `ProgramInformationTable` and `ProgramReviewTable`.

Any party capable of delivering compliant TV-Anytime data could provide a metadata service. Examples include: content creators, content providers, service providers, CE manufacturers and third parties aggregation services.

- The request contains parameters that specify the type of metadata required by the client.
- The types of metadata returned could be any TV-Anytime Metadata Specification along with content referencing information.

Metadata Service Capability Descriptions

In order to usefully exploit the metadata services described in the section 4.2.4, the client needs information about the nature of the metadata service being offered. This is because different metadata services will provide different types of metadata and can be queried in different ways. For example, some metadata services may offer just content referencing information, whilst others may provide program metadata but no segmentation information. Similarly, whereas one server may only be able to accept simple requests for metadata based on a CRID, another server may offer much more sophisticated querying and sorting capabilities. Moreover, different types of queries are only useful if a client is able to establish sensible values with which to query. An example of this is a query for scheduling data (`ProgramLocationTable`). In order to query for scheduling information on a particular content delivery service, the client needs to know the content delivery services for which that metadata service has data. To address this issue, each metadata service provides, on request from a client, a capability description. This capability description allows a client to flexibly query a metadata service, without making requests that will not be supported by that metadata service. Furthermore, it allows metadata service providers to flexibly implement the server in a way that is appropriate to the data that they have available.

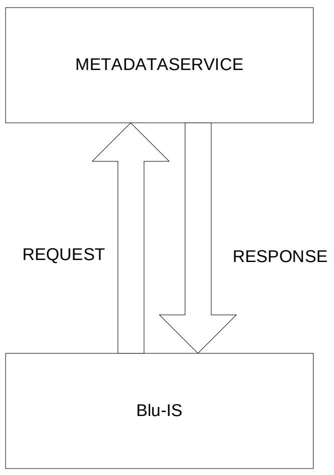


Figure 4.7: MetadataService

4.3 Packaging System

This section contains a design for a Packaging System. Section 4.5 provides an architecture for this system, giving a description of the roles of the actors and the possibilities of the package distribution channels. Section 4.3.3 will then explain how packages behave for the different distribution channels and how actors work and interact with this packaging system.

Figure 4.8 shows a schematic architecture of the system, depicting the actors of the system and the three different distribution channels. The architecture illustrates all possible ways, in which packages, content and metadata can be delivered to the user.

4.3.1 Actors

The three groups of actors that exist in the system are the content creator, the content aggregator and the content consumer (the end-user). A content creator creates the content for a package. Content can be many forms of media, i.e. video, audio, images, text, etc. The content creator also creates associated metadata for this content.

- A content aggregator receives the content, which is created by the content creators, and constructs packages for some groups of content. The aggregator also adds metadata for the package. He can add or edit metadata of the content files that the content creator provided, i.e. adding technical metadata on these files describing the bit rate, duration, etc.
- The constructed packages can then be offered to the end-users through some distribution channel. The content aggregator decides which packages are being distributed and how these are distributed.
- The end-user can receive and view content and packages. The end-user interacts with the Blu-IS system to retrieve and view the content that he wants. Furthermore the system interacts with the Internet to perform tasks such as searching for packages and resolving CRIDs.

4.3.2 Distribution channels

There exist three different distribution channels in the system through which packages, content and associated metadata are distributed to the Blu-IS system. The main difference between the distribution channels is the availability of a package, when it is distributed.

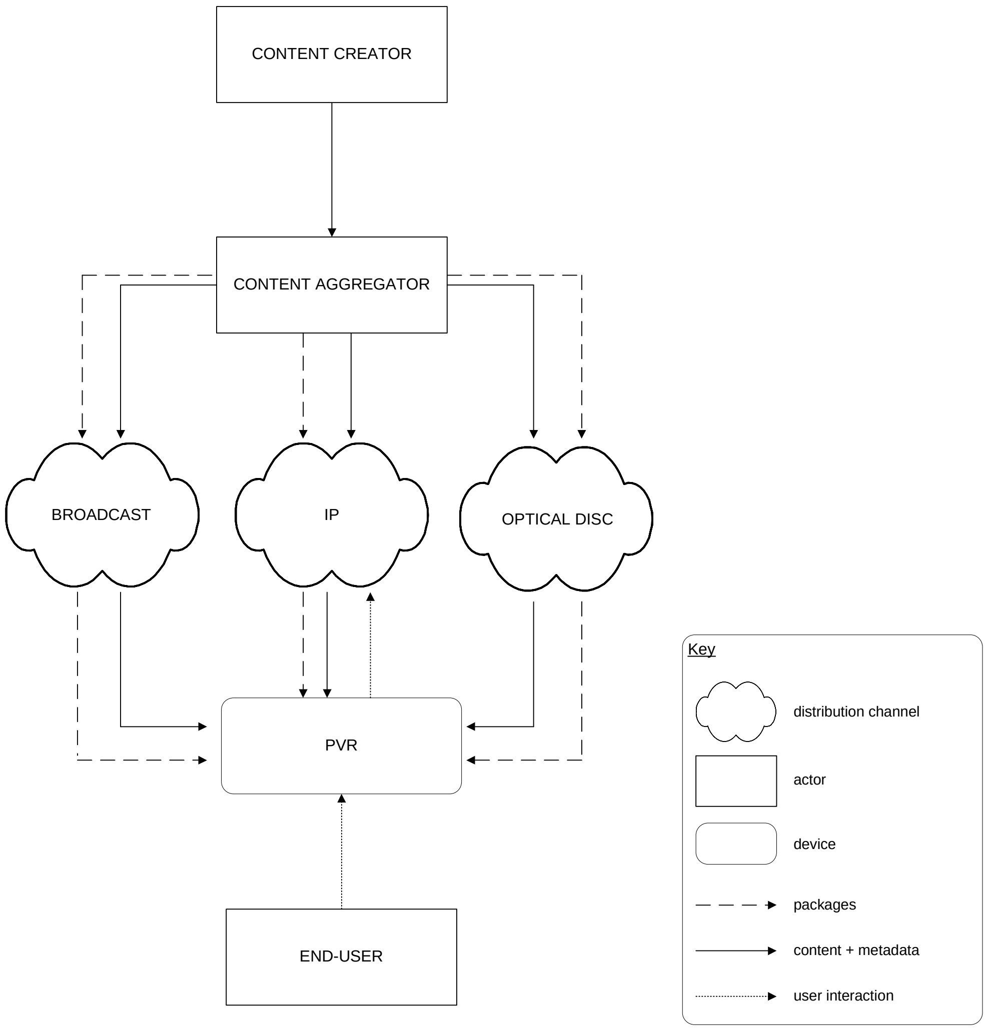


Figure 4.8: High level system architecture

Watching content over a broadcast stream is the way of watching TV, as it is known today. This viewing experience can be enhanced by also offering packages together with the broadcast stream. Whenever a package is transmitted together with the broadcast signal, the system retrieves the package and stores the package locally, so the user can access it later. When the user wants to view content, or when the system needs to record some kind of content, the content can be received, but only at the time that the content aggregator broadcasts the content (no on-demand viewing).

Packages can be distributed via the Internet, offered by some Web Server. An important feature of distributing packages via the Internet is that the end-user can download or view a package or content at any time.

Distribution of a package (and content) through an optical disc is primarily used for distributing High Definition (HD) content, as this is a relatively cheap way to offer HD content and because this type of content still takes a lot of time to download, because of the size of a HD file

4.3.3 Usage of a package

As described in the previous sections, packages and content can be distributed to the end-user's Blu-IS system in three different ways. A package acquired via one distribution channel can point to content on any of the three distribution channels. So, in some way, the three worlds of broadcasting, Internet and optical storage become interconnected. The following sections will demonstrate some example of use cases. How packages and content can be distributed through the three different channels, what the possibilities of such a package are and how an end-user can work with such a package in practice.

Digital Broadcasting

A user is watching a Teleac course about the French language that is being broadcasted. A package, containing links to the program's website, accompanying the broadcast stream is automatically retrieved. The package also contains a link to information on a Blu-ray disc, containing all episodes of the program. An icon appears at the top of the screen indicating that there is extra information available for the broadcasted program. The user presses a button on his remote control and the package is presented.

The user can also choose to record a broadcasted program to the Hard Disk Drive on his system to watch it later. In that case, the package can also be downloaded to the box so that it can also accompany the recorded program when it is viewed later on by the user.

Internet

A user searches for a package on a language course French in an online package searching system that he accesses from the system. The user finds a package and downloads it to the system. After the download, the package is read by the system. The package contains links to streaming media clips and an online presentation. There is also a link to some broadcast stream, where a Teleac course on the French language is broadcasted. Furthermore, a link to a Blu-ray disc is included, which contains HD (high definition) content with information on the city of Paris. The user is shown information about this disc and is given the option to buy it online.

Optical Disc

A content creator creates an educational documentary about the history of the French culture, used for background information on a French language course. The content creator has also created some extra content accompanying the movie, consisting of some sample lessons of different learning institutes. All content is being distributed to a content aggregator, which produces an optical disc containing a package with the movie in High Definition format. Additionally, the package will contain links to the website and a link to some reviews and forums. The end-user buys this disc in a shop (or online). When he puts the disc in the optical disc drive of his system, the package will be automatically started (presented).

4.3.4 Content authoring

The previous section describes how content is distributed to end-users in a packaging system. However, end-users may want more than just being able to retrieve and view content. They may want to author content themselves. For example, a user could create a family photo package, which the user then shares with his family. To achieve this type of functionality, the presented design of a packaging system is extended by also permitting end-users to create, modify and share packages, content and metadata. First, an updated architecture, which includes this functionality, is presented. Then the creation, modification and sharing of packages is discussed.

For the extension of the system with functionality for content authoring by end-users, the system needs to be able to share packages, content and metadata. Furthermore, end-users need to be able to distribute their content via optical disc and the Internet. Figure 4.9 shows the extended version of the system architecture presented in the previous chapter in Figure 4.8.

In the extended system, there are two actors, who can create packages. These are the content aggregators and the end-users. Content aggregators distribute the packages and for this purpose they create packages that are distributed either through a broadcast stream, through publishing on

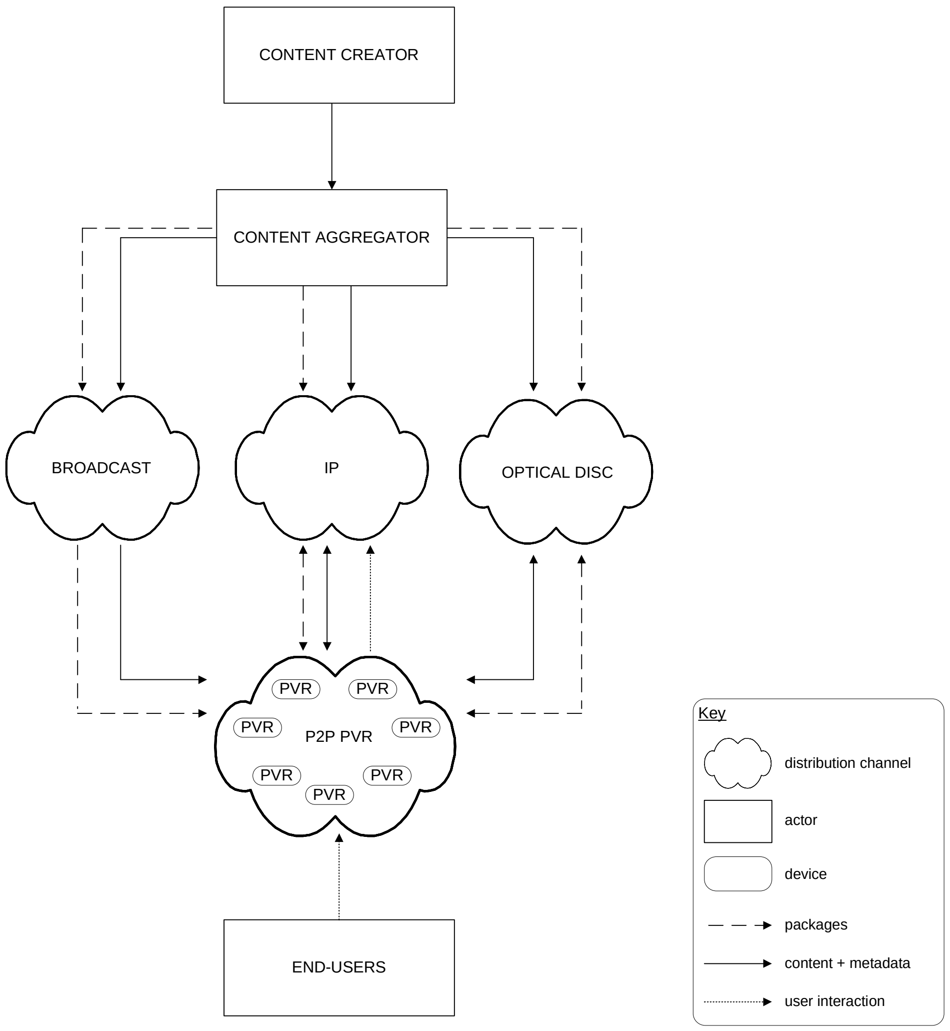


Figure 4.9: Extended system architecture

an IP location or through optical discs that an end-user can buy in a store. End-users (and content aggregators) can create and share packages using the Internet or P2P network as a distribution channel. An end-user can also burn a created package to a disc and then distribute the disc.

If a user wants to create a package, the following steps need to be taken by the system. These steps describe the process from creating a package from scratch to publishing a package.

1. Select content for the package.
   The user selects the content that he wants in the package. This can be content from any distribution channel.

2. Create the structure of the desired package.
   The user selects how all content is organized inside the package, thereby creating the structure of the package. See the French course in section 4.2.3 for an example, where the package is divided into two lessons, containing different topics, each with a set of content elements.

3. Create CRIDs for the content and the package
   The system should create CRIDs for all content elements of the package and for the package itself. A CRID authority creates these CRIDs.

4. Create the structure of the CRID resolving tree.
   With the use of the package structure and the CRIDs of the content, a CRID resolving tree is created, so that the CRIDs of the package can be resolved.

5. Share the package
   The package can be shared through a P2P network, or can be published on the Internet. Alternatively it can be burned to an optical disc and distributed by hand. The user selects which groups of users are allowed to view the package. For example, the package is only meant to be shared among family members. This can be done with the use of DRM technologies. DRM is considered to be outside the scope of the design of this packaging system. However, it should still be taken into account that this step should be taken to describe a realistic package creation process.

Authenticity

The CRID that points to a package will be resolved at a certain CRID authority, which mentions the creator of the package, thus ensuring authenticity. Likewise, the creator of the content of the package is also determined by the CRID. When someone adapts a certain package and publishes it again, a different CRID is created for that package. This CRID belongs to a certain authority, which links the adapted package to its creator. The user can then choose whether he trusts that authority or not. As a content creator, there should be a certain “trust” in the authority that publishes the content in a package. The content creator can send the content to the aggregators that he trusts and can choose not to send content to some aggregators if he thinks that they will misuse his content.

Modifying a package

Content aggregators and end-users can also *modify* a package. Modifications to a package can be split up into three parts: insertion, deletion and modification of the content elements of a package. Because the methods used to process modifications to a package are likewise for each of these parts, only the extension of a package is discussed as an example in the next discussion. There are two different possibilities/methods to modify a package. To make a good comparison, both methods are investigated. This will be done with the use of the French course example. First, a description is given about the extension of a package by adding an extra content element: “sight seeing Paris”. Method 1 involves extending a package, by adapting CRID resolution at the CRID authority. Method 2 involves extending the package itself, by changing the XML-file of the package.

Method 1: Modifying a package, by adapting CRID resolution at the authority

In Figure 4.10, a graphical representation is shown of extending the topic ’short break in Paris’, with an extra subtopic ’sight seeing in Paris’. This method adds a CRID in the CRID resolving tree of the CRID Authority. The newly added CRID is displayed in red.

To make such a change for the package, the user has to take some actions. First the user needs to publish the content and obtain a CRID for that content at an authority. After this step, the user has to change the resolving of the CRID ’Asking for directions’, so that the CRID not only resolves to a locator, but also to the newly added CRID. If these two steps are completed the package is extended with content. In this situation, the package’s XML-file itself isn’t modified at all. To give a complete overview of the changes, the modified XML-file of the CRID resolution tree is displayed in Figure 4.11, with the changes shown in red.

Method 2: Modifying a package, by adapting the package’s XML-file

Another way of extending a package is to add the CRID to the package’s XML-file as an extra component. Figure 4.12 shows how the package will look like with this extension. The red parts

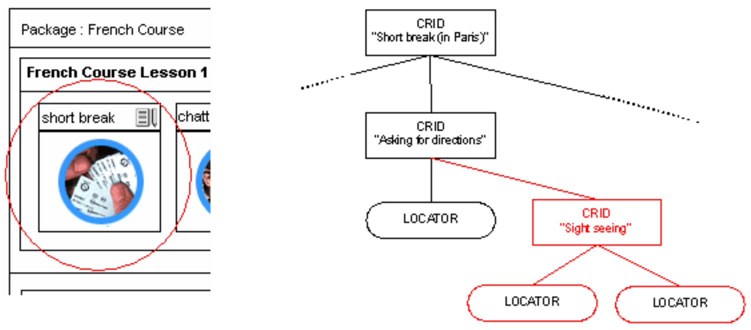


Figure 4.10: Extending a package using CRIDs

in the figure highlight the changes in the package with respect to the original package presented in Figure 4.2.

Changing the package in this way is also a two-step process. First the content needs to be published and referenced using a CRID, just as described in method 1. After obtaining the CRID, the user needs to add the new CRID to the existing package. The next step is to redistribute the package. This can be done, by creating a new CRID, which resolves to the newly created package. Figure 4.13 shows the changes in the XML-file of the package.

Analyzing the two methods

As can be seen in the description of the two methods, the needed changes to extend the package are different. This difference lies in the fact that with method 1, only the CRID resolution process needs to be updated. This change is performed at the corresponding CRID authority. With method 2, the XML definition of the package needs to be updated.

To make a well-founded decision of which method is better, the differences between these methods are discussed concerning the way of distribution, structure of the package, maintainability and scalability.

Broadcasting & Internet

Both methods make it possible to extend a package distributed through broadcasting or the Internet. Consider the following scenario. An end-user has viewed a certain package some time ago. Now the content aggregator or some end-user extends this package with new content. When a user views this package again, the CRID linking to the package's XML-file will be resolved and the package's XML-file will be retrieved. Both methods will retrieve the updated package in this way.

Optical Disc

With an (read-only) optical disc, it is not possible for a content aggregator or end-user to directly extend the package on the disc, using method 2, because of the fact that the disc is read-only. However, it is technically possible to extend a package using method 2. This can be done by creating the disc in such a way that it contains a CRID pointing to a package on the Internet. The XML-file of this package can be extended by the content aggregator or end-user. Each time the user wants to view this package, the CRID on the disc is resolved to retrieve the XML-file of the package. When the content aggregator is using method 1, the content aggregator can extend the package by modifying the CRID resolution process at the authority.

```
<Result CRID="crid://frenchcourse.com/chapter1/askfordirection"
  status="resolved" complete="true" acquire="all">
  <LocationsResult>
    <Locator>
      http://frenchcourse.com/resources/chapter1/askfordirection/speech.mp3
    </Locator>
  </LocationsResult>
  <CRIDResult>
    <Crid>crid://frenchcourse.com/chapter1/sightseeing</Crid>
  </CRIDResult>
</Result> <Result
CRID="crid://frenchcourse.com/chapter1/sightseeing"
  status="resolved" complete="true" acquire="all">
  <LocationsResult>
    <Locator>
      http://frenchcourse.com/resources/chapter1/sightseeing/eiffeltoren.avi
    </Locator>
    <Locator>
      http://frenchcourse.com/resources/chapter1/sightseeing/seine.avi
    </Locator>
  </LocationsResult>
</Result>
```

Figure 4.11: XML snippet of the extended CRID resolution tree

Maintainability & Scalability

When the content aggregator decides to update a certain package, it is possible that the user implicitly changes hundreds of other packages. This is because a single CRID can be used in more than one package. When extending a package using method 1, an update of the resolution of one single CRID is enough to update the all packages that are using this CRID. However, when using method 2, every package that is using this CRID needs to be updated manually by updating the XML-file.

On the other hand, with method 1 it is not possible to apply the changes to part of these packages that are using the same CRID. In this case method 2 is the only solution for this problem.

Package Structure

When extending a package with the use of method 1, by extending the CRID tree structure, the CRID structure needs to be suitable for this. For example, Figure 4.10 shows a CRID pointing to a set of lessons, which is extended to include another lesson. However, if the author of the package has made no such CRID, there is no CRID that can be extended to include the new lesson. In this case the author has to adapt the XML-file of the package (method 2) to solve this problem. With method 2 however, this problem does not exist, but the issue encountered here is likewise. When adapting the XML-file of the package, content can be added only at the 'top-level'. For example, it is not possible to add the new lesson, showed in Figure 4.10, to the 'lesson1' item if the chapters of this lesson are not mentioned in the XML-file at all, but are retrieved through the CRID resolution of the 'lesson1' CRID. So in this case, method 1 is needed to solve the problem.

Conclusion

There are two methods that can be used to modify a package.

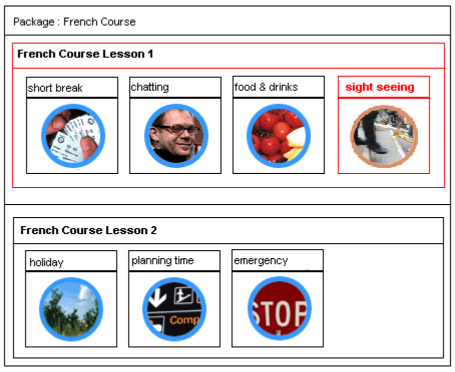


Figure 4.12: Extending a package using the package XML file

```
<tva2:Item>
  <tva2:Component>
    <tva2:Resource crid="CRID://frenchcurse.com/chapter1/shortbreak"/>
  </tva2:Component>
  <tva2:Component>
    <tva2:Resource crid="CRID://frenchcourse.com/chapter1/chat"/>
  </tva2:Component>
  <tva2:Component>
    <tva2:Resource crid="CRID://frenchcourse.com/chapter1/food"/>
  </tva2:Component>
  <tva2:Component>
    <tva2:Resource crid="CRID://frenchcourse.com/chapter1/sightseeing"/>
  </tva2:Component>
</tva2:Item>
```

Figure 4.13: XML snippet of extended package

Method 1: Modification of the CRID resolving tree
Method 2: Modification of the structure of the package

Advantages/Disadvantages of using method 1

- Modifying a single CRID results in updates on all packages using this CRID
- Structural changes in the package can be made

Advantages/Disadvantages of using method 2

- Other packages are not affected by changes made in a certain package, because the CRIDs are not modified.
- Only 'top-level' changes can be made to the package's XML file

The type of distribution channel used poses no real preference for what method to use. As for the other aspects, package structure, maintenance and scalability: each method has its own advantages and disadvantages. Both methods can be used next to each other in order to get the best of both methods. Problems that occur when using one method can be solved by using the other method. When developing the system, above described issues that occur for both methods should be taken into account.

4.3.5 Placing the CRID Authority

The resolving and *publishing* of the CRIDs is performed by the CRID authority. A next research question arises "where is this CRID authority situated, on the Blu-IS platform of the end-user, or is it an online service, which is responsible for the CRID resolving". This issue is investigated by analyzing the advantages and disadvantages of placing an authority on the box, or at an online service

Disadvantages CRID authority on the box

- When the box is power-down, the CRID resolving is not working anymore. When there was also a CRID which points to CRIDs which are not resolving to the box that content can't be accessed, because the CRID cannot be resolved
- When the box crashes or the box is replaced with a new box, all the CRID resolving trees are lost

Advantages CRID Authority (and content) on the box

- No privacy issues, because the end-user stays always in control of its own package and content, because its own box distributes it.

Disadvantages CRID authority as an online service

- The end-user needs to 'trust' an authority

Advantages CRID Authority as an online service

- The CRID authority is always available
- The pillars of the complete system are completely in hands of the system engineers
- Less security issues

Based on the mentioned advantages and disadvantages, can be concluded that it is better to have a CRID authority as an online service instead of as a service on the PVR. The major reason for this choice is mainly the availability of the CRID authority. This is very important for the working of the complete system. When the CRIDs cannot be resolved, the whole system is not working. When some content isn't available (because the box is offline, or the content is deleted) the system as a whole isn't affected by this.

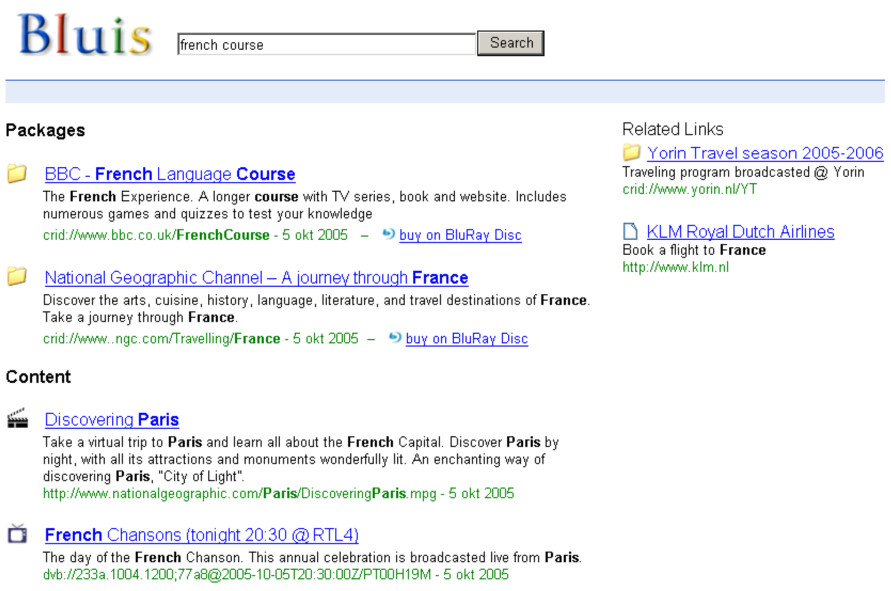
4.4 Personalization design

Now that we have described a way to model our content in the system and have given a description on how this system is going to work with this content, it is time to add personalization to our system. This section presents a design on how personalization is achieved in the Blu-IS system. First we will describe what possible queries the system is able to personalize and in what way the user can ask these queries to the system. Then an overview is given on how these queries are processed by the system and in what way personalization will play a role here. Four global steps are identified in this process: *query refinement*, *query execution*, *filtering* and *presentation.* These are then discussed in more detail.

4.4.1 Possible queries

The queries listed in this section are all possible queries for which our system is able to return a personalized result. Of course the user can query the system for lots of other tasks, but these are not related to personalization tasks. This section does not go into detail on what personalization techniques are used. This is done in the description of the personalization process in section 4.4.2. The system supports a personalized response for the following queries.

1. **Search for packages containing some keywords.**
   A user wants to search for packages and enters keywords to search for in a "Google-like" User Interface of a package browser application. The system will search for packages that the user will probably like and return a set of packages, which is personalized for the user and matches the keywords he supplied. The result of such a query is shown in figure 4.14.



Packages

BBC - **French** Language **Course**
The **French** Experience. A longer **course** with TV series, book and website. Includes numerous games and quizzes to test your knowledge
crid://www.bbc.co.uk/**FrenchCourse** - 5 okt 2005 – buy on BluRay Disc

National Geographic Channel – A journey through **France**
Discover the arts, cuisine, history, language, literature, and travel destinations of **France**. Take a journey through **France**.
crid://www..ngc.com/Travelling/**France** - 5 okt 2005 – buy on BluRay Disc

Content

Discovering **Paris**
Take a virtual trip to **Paris** and learn all about the **French** Capital. Discover **Paris** by night, with all its attractions and monuments wonderfully lit. An enchanting way of discovering **Paris**, "City of Light".
http://www.nationalgeographic.com/**Paris**/Discovering**Paris**.mpg - 5 okt 2005

French Chansons (tonight 20:30 @ RTL4)
The day of the **French** Chanson. This annual celebration is broadcasted live from **Paris**.
dvb://233a.1004.1200;77a8@2005-10-05T20:30:00Z/PT00H19M - 5 okt 2005

Related Links

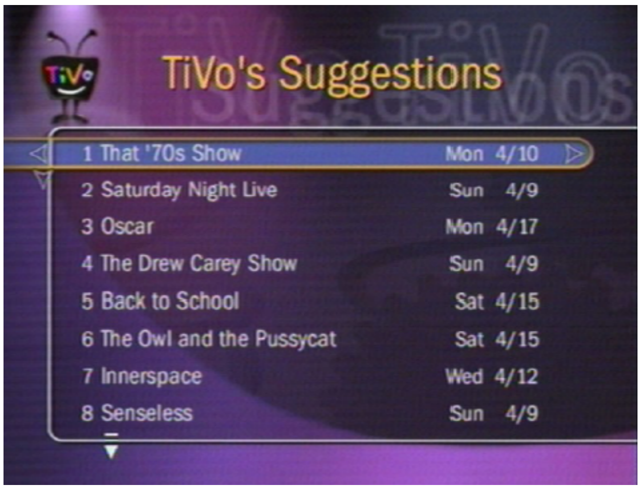
Yorin Travel season 2005-2006
Traveling program broadcasted @ Yorin
crid://www.yorin.nl/YT

KLM Royal Dutch Airlines
Book a flight to **France**
http://www.klm.nl

Figure 4.14: Example of a personalized resultset for a search for keywords (PC Interface)

2. **What will be broadcasted/has been published during some period of time?**
   A user browses to an EPG, which shows all programs that are on TV tonight. To be able to fill the EPG with the (recommended) programs for tonight, the system automatically sends a query asking for the programs that are broadcasted tonight. Similarly, in the case that the EPG contains packages that are stored online, the system sends a query to ask for recently published packages or packages that have been published during some period of time. So the query that is sent is basically a request for content in a certain time interval. The result of the query, figure 4.15, is a list of TV programs or packages that match these criteria and which is personalized for the user.

3. **Present me this package.**
   For this query the user selects a package from a result set acquired from one of the two previous queries. When the user wants to view the contents of the package, the system automatically sends a query to retrieve the contents of the selected package. The contents of the package are presented in a personalized way for the user, see figure 4.16 for an example.



TiVo's Suggestions

1 That '70s Show	Mon 4/10
2 Saturday Night Live	Sun 4/9
3 Oscar	Mon 4/17
4 The Drew Carey Show	Sun 4/9
5 Back to School	Sat 4/15
6 The Owl and the Pussycat	Sat 4/15
7 Innerspace	Wed 4/12
8 Senseless	Sun 4/9

Figure 4.15: Tivo's [56] recommendations for content in an EPG (TV Interface)

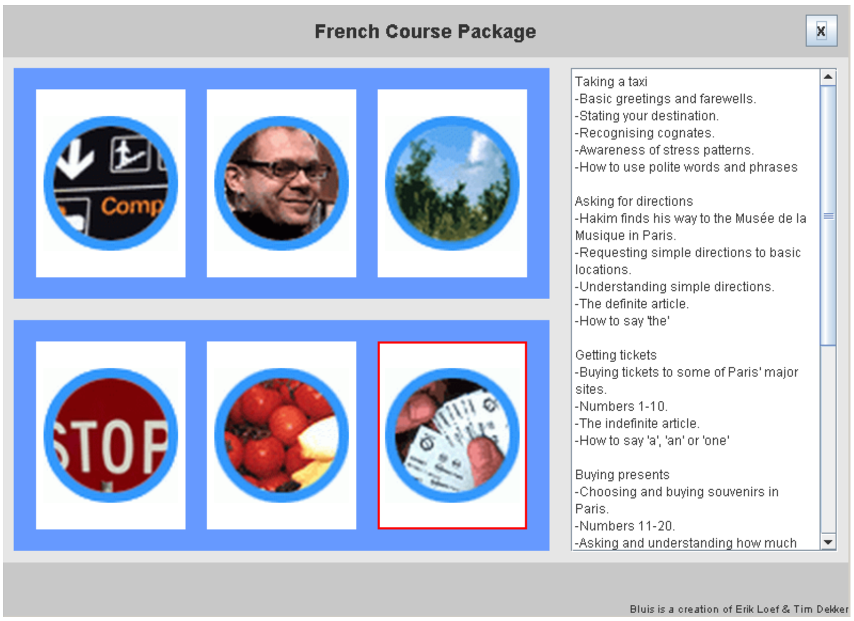
4.4.2 Personalizing Queries

The above described queries all acquire a personalized result by having the result of the query being filtered down in size, so that only relevant packages, reflecting the user's interests, are presented to the user ordered decreasing by relevance. Apart from this filtering of a resultset, the first query, a search for keywords, also allows for another personalization step. When a list of keywords is sent to the system, the system does not know the meaning and the context of these keywords. The system can improve this query by first using an intelligent query *refinement* algorithm, which tries to derive this meaning or context of keywords. The output of this algorithm will then be a query where the original keywords have been replaced by *concepts* of which the system knows what they mean and what relation they have to other concepts. Figure 4.17 gives a high level overview of the process that is taken for personalizing the result of such a query. The idea is that we can first try to extend or refine the query with additional related concepts, whereafter this query is executed by the system. The resultset of the query is then narrowed down by a set of personalization techniques.

In figure 4.17, the *User Interface* runs on a device (i.e. TV, PC) that is used to interact with the Blu-IS box. The *Query Engine* runs on the box itself. The components that are used in the description of the personalization process are not actual components of the system architecture but rather act as an illustration of the process steps that are followed to retrieve a certain personalized result set given a user's query. A description of how the components of this process fit in the system architecture of is given in section 4.5. Firstly, we describe the initial step, *query refinement* in more detail. Then we will describe how the rest of the process is done.

Query Refinement

Quite some research has been done in the area of query refinement [45]. Typically, this is done with the use of *ontologies*. Inferences on ontological knowledge are then used to refine the query. Multiple ontology domains can be used to refine the query with knowledge from all domains. We can roughly divide the knowledge we want to add for a query in *"What"*, *"Where"*, *"When"*, *"Who"* types [67] to determine the context of the query keywords. So we want to use ontologies for all of these domains in our query refinement process. For the *"What"* type, we use a lexical ontology, that is able to determine the meaning of words, like a dictionary. The *"Where"* type, we use a geographic ontology that include information on geographic places and their relation to other places. For the *"When"* type, we use a Time ontology, describing time concepts and their place in time related to other concepts. For the *"Who"* type, we can use an ontology containing (famous) people and their relation to each other. There are a couple of existing ontologies that



Taking a taxi
-Basic greetings and farewells.
-Stating your destination.
-Recognising cognates.
-Awareness of stress patterns.
-How to use polite words and phrases

Asking for directions
-Hakim finds his way to the Musée de la Musique in Paris.
-Requesting simple directions to basic locations.
-Understanding simple directions.
-The definite article.
-How to say 'the'

Getting tickets
-Buying tickets to some of Paris' major sites.
-Numbers 1-10.
-The indefinite article.
-How to say 'a', 'an' or 'one'

Buying presents
-Choosing and buying souvenirs in Paris.
-Numbers 11-20.
-Asking and understanding how much

Figure 4.16: Visualization of a package (PC Interface)

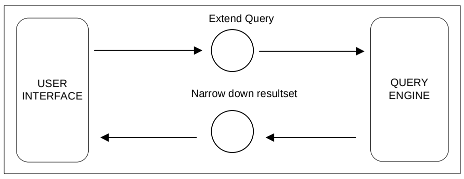


Figure 4.17: Personalization by query refinement and filtering techniques

match these profiles quite well. We will describe them next.

- **WordNET ontology**
  The WordNET ontology [46] is a freely available lexical ontology. English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept. Different relations link these synonym sets.

- **TV-Anytime ontology**
  The TV-Anytime metadata specification [6] contains a list of predefined genres, which are classified in a hierarchy of genres, linking genres to subgenres. We can use this ontology to refine a query, which searches for content from a specific genre. This classification for the moment is described using an XML structure. This structure needs to be converted to the Ontology Web Language OWL before we can use this information as an ontology.

- **SUMO ontology**
  The SUMO (Suggested Upper Merged Ontology) [47] provides a geographic ontology, describing countries and their relation to other countries as well as properties on these countries.

- **OWL-Time ontology**
  The OWL-Time ontology [48] is an ontology for describing the temporal content of Web pages and the temporal properties of Web Services. Including this time ontology in our system can enable query refinement on the *"when"* context type.

4.4.3 Personalization Process

Now that we have described a basic approach to a personalization process, by first refining the query and then narrowing down the resultset through filtering, we can take a more detailed look at the exact process.

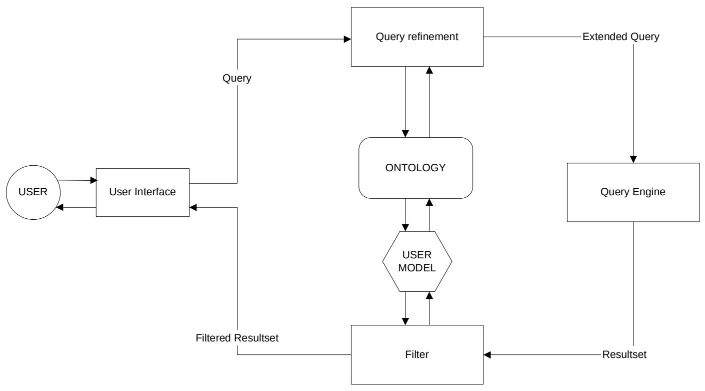


Figure 4.18: A detailed overview of the personalization process

Figure 4.18 shows that there are basically four global steps that are executed to achieve personalization. First there is the *refinement* of a user-made query, which is done with the use of ontologies. The second step in the process is to *execute* the refined query at the *Query Engine*, which returns the corresponding resultset. This resultset, a list of content items that match the query, is the input for the *filtering* techniques, which is the third step of the personalization process. These filtering techniques narrow downl the resultset using the user model(s) of the active user(s). After the filtering a subset of the original resultset will be sent to the *User Interface*, where the last step is performed, by personalizing the *presentation* of the resultset.

The data that flows between the different components are either queries (refined or not) or resultsets (filtered or not). A query contains either "plain text" keywords or a request for recommendations, see section 4.4.1 for all possible queries. A resultset is an XML-document complying with the TV-Anytime phase 2 metadata specification [6].

Query processing

We now illustrate the workings of this process by giving the possible queries we mentioned before as examples, whereafter we give a formal description of the query processing.

1. Search for packages containing some keywords.
The *query refinement* component uses an ontology to refine the query with related keywords. The refined query is executed and the resultset is filtered whereafter the result for the search is presented to the user. The query refinement step here can be seen in figure 4.19, where "France" and "Paris" are added as extra keywords for the user query: "French course".

2. What will be broadcasted/has been published during some period of time?
To be able to fill the EPG with the (recommended) programs for tonight, the system automatically sends a query asking for the programs that are broadcasted tonight. Similarly, in the case of packages that are stored online, the system sends a query to ask for recently published packages or packages that have been published during some period of time. The query is forwarded to the *Query Engine* component, whereafter the resultset is filtered and forwarded to the *User Interface* to present it to the user.

3. Present me this package.
When the user wants to view the contents of the package, the system automatically sends a query to retrieve the contents of the selected package. Like with the previous query, this query is directly forwarded to the *Query Engine* component, whereafter the resultset is filtered and forwarded to the *User Interface* to present it to the user. This way the most recommended items of the package are presented to the user.

4.4.4 Personalization algorithms

We will now, for each step of the personalization process, describe which algorithms we have designed to accomplish the proposed effects described in the previous sections. To clarify the dependencies of the different steps the process is described in a formal way. The process of personalization can be described using the following functions.

$QE(e, o) = e'$
$R(e') = r$
$F(r, u) = r'$
$P(r')$

Where

QE = query refinement function
e = query expression (keywords)
o = ontologies
e' = query expression e expanded with related keywords from ontologies o

R = retrieval function
r = result set of the performed refined query *e'*

F = filtering function
$u_0 \dots u_n$ = user model(s)
r' = filtered result set based on the user model(s) u

P = presentation function

We will now explain each separate step of the personalization process in more detail with the use of a running example. The six steps that are described next are based on the above equation, and will give an overview of what tasks each separate step is going to perform.

Step 1: The user starts the search User Interface

The search interface has a so-called "Google" interface (figure 4.14), which is familiar and intuitive for almost every user. In the search interface a user enters a certain set of keywords and activates the "personal search". To make this example more clear, the variables from the function are filled with data, to get a more realistic impression. In this example we use e = *"French course"* as our query. The query is sent to the Blu-IS server, where it is processed by the next step.
Formally:
Keyword: k
Number of keywords: $[0, n]$
Query expression: e

$$e = \begin{array}{|c|c|c|}\hline k_0 & \dots & k_n \\ \hline\end{array}$$

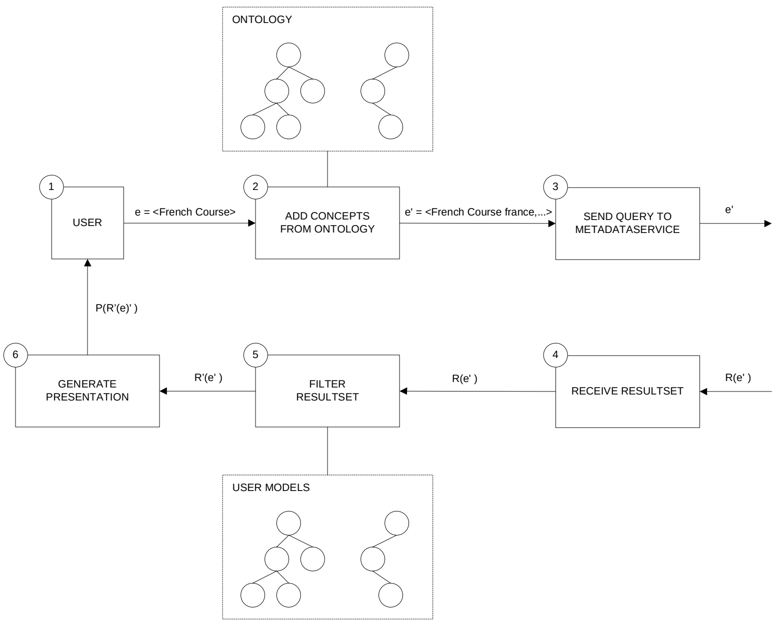


Figure 4.19: An example of how the personalization process is performed

Step 2: Ontology inferencing

The keywords *e* entered by the user in the previous step, are the input for this step, the query refinement process. The goal of this query refinement is to use ontology inferencing to add intelligence to the query in the form of related relevant keywords and by finding the context of these keywords. To this end, we use the four ontology types to refine the query as described in section 4.4.2.

We do this by first extending the query with synonyms using the WordNET ontology. This will provide us with a broadened search, which contains results that actually match the user's query, but are labelled with a synonym of a keyword that the user has searched for.

After we have extended the query with synonyms, we query all other ontologies that we use (geographic, time, TV-Anytime) to find matching concepts, thus determining the context.

When a keyword matches a concept in an ontology, we add this keyword, together with it's context, to our query. Now we can use the intelligence of the ontology to find related concepts for the matched concept. As a result of this process, the original keywords (*e*) are extended with the related concepts of the ontology by the described reasoning process, which results in *e'*.

In our example, the we start of with a user initiated search for keywordset:

e = <french, course>

In extending this query, the WordNET ontology will find a match at concept "french". Furthermore, it will find that this concept has a *pertainsTo* relation connecting the word *"french"*. to the word *"france"*. This keyword is added to the query *e*. This results in the extended query:

e' = <french, course, france>.

We now query the other ontologies with all keywords from *e'*. The geographic ontology from SUMO will then match the word *"france"* with the country *"France"*, whereby the system has derived a possible context of the word *"france"*. The information that France could actually be a country is added to the query by adding the keyword *"country:France"*. Furthermore, using the geographic ontology, we find that the country France is related to countries Belgium, Germany, Switzerland, Italy and Spain by the *"neighbourOf"* relation. These keywords are added as well, but with a lower semantic closeness value, as they do not directly correspond with what the user asked to the system. However, they may provide valuable information in retrieving related content items. So after ontology inferencing, we end up with the query:

> *e' = <french, course, france, country:France, country:Belgium, country:Germany, country:Switzerland, country:Italy, country:Spain>*

All of these keywords have a semantic closeness value linked to them, indicating the importance of that keyword in the search. The following algorithm describes in detail how we find matching entities for a keyword k and calculate the semantic relevancy of entities in ontologies w = WordNet, g = geographic, *tva* = TV-Anytime. This algorithm is executed for each keyword k of the query e. The vector *e'* will fill up with keywords during the execution of the algorithm, thus making up the extended query. *e'* is initialized with the set of keywords e, all with semantic weight *s = 1.00.*

Extension with synonyms

1. search w for entities with names that directly match or are synonyms of k.
2. for each match p, search w for synonyms.
3. for each synonym q of p, add q to the vector *e'* and assign it a semantic weight *s = 1.00*

After we have done this, we perform the following algorithm for each keyword k of *e'*.

Extension with location and genre context

1. search g for entities with names that directly match k.
2. for each match p on a country concept, add p to the vector *e'* and assign it a semantic weight *s = 1.00*
3. for each neighboring country q of p, add q to the vector *e'* and assign it a semantic weight *s = 0.50*
4. search *tva* for entities with names that directly match k.
5. for each match p on a genre, add p to a vector *e'* and assign it a semantic weight *s = 1.00*
6. for each subgenre q of p, add q to a vector *e'* and assign it a semantic weight *s = 0.25*

In general, executing this step will result in an extended query, with for each keyword the semantic weight stored.

$e' =$

k_0	$\dots$	k_n	k_{n+1}	$\dots$	k_m
s_0	$\dots$	s_n	s_{n+1}	$\dots$	s_m

p and s can also be null (unset) if no match was found for keyword. s can have value *[0,1]*.

Step 3+4: Executing the Query

The extended query (*e'*) is sent to the metadata service, to execute the query. The metadata service is an online service (this implies that the actual query execution is done outside the local BLu-IS system, wherafter the resultset r is sent back to the local system when the query has finished executing. Each content item c of r that has been found matching a keyword is assigned a combined relevance value s_c, which is the weighted sum of the semantic weights $s_0 \dots s_m$ for the set of matching keywords $k_a \dots k_b$:

$$s = \frac{\sum_{i=0}^{n} s_i}{n},\ keywords\ [0..n]$$

The resultset for our example could look like:

$r =$

Item	**Matches keywords**	**Relevance**
French Course	french, course, country:France	0.3
Les Miserables	french, france	0.2
Henry V	french, france	0.2
Cyrano de Bergerac	country:France	0.1
European Vacation	france	0.1
The Three Musketeers	french	0.1

In general, the resultset r will contain a list of content references $c_0 \dots c_\alpha$, together with a list of matching keywords $k_0 \dots k_\beta$ and their relevance value s_c.

$r =$

c_0	$kc0_0 \dots kc0_{c0\beta}$	s_{c0}
...	...	...
c_α	$kc\alpha_0 \dots kc\alpha_{c\alpha\beta}$	$s_{c\alpha}$

Step 5: Apply filtering

$F(r, u) = r'$
F = filtering function
r = result set of the performed query *e'*
$u_0 \dots u_n$ = user model(s)
r' = filtered result set based on the user model(s) u

The resultset r is, together with the user model(s) of the user(s) making the query, the input for the filtering component. The filtering component uses a set of different filtering/recommendation strategies in a filtering function F to sort and narrow down the resultset r. This provides the user(s) with user model(s) $u_0 \dots u_n$ with a personalized result *r'* for the query *e*.

In this section, we present a hybrid approach for achieving this personalization. As discussed in 3.3.3, we achieve an increased personalized experience and suffer less from the disadvantages of the individual techniques if we try to combine multiple techniques. Also, as defined in our goals, we want to have a personalized experience on both the user as well as on its context, which can only be accommodated for when using a hybrid technique.

This hybrid technique will compute prediction values for all content elements using the different technique.

First we will use content-based filtering to generate predictions $lp_0 \dots lp_\alpha$ for the content items $c_0 \dots c_\alpha$. Local filtering will look at the classification elements of the TV-Anytime metadata for the content item and then consult the user's user model for the preference for that type of content.

After local filtering, we will use the collaborative filtering technique to generate prediction values $cp_0 \dots cp_\alpha$ for the same content items. This technique uses the CRID of the content to determine the popularity of that item in the user's community.

So predictions lf and cf will reflect the user's own interest respectively the prediction based on the user's community. These different prediction values are then merged into predictions $mp_0 \dots mp_\alpha$ which reflect both predictions.

Finally, to reflect the relevance of each content item, we multiply the merged prediction value mp with the semantic relevance value s to achieve the final prediction value p for the content items.
This will result in a result like:

$r =$

Item	**Prediction Value**
French Course	2.5
Cyrano de Bergerac	2.0
European Vacation	1.8
Les Miserables	1.5
Henry V	0.5
The Three Musketeers	0.3

In general, the output of this step will be:

$r' =$

c_0	p_{c0}
$\dots$	$\dots$
c_α	$p_{c\alpha}$

We now have acquired a result, which is personalized on both the local user as well as on its community. However, as we stated in our assignment goals, we also want to personalize based on the user's context. To this end, we have chosen to include the "group modelling" strategies in our design here to personalize the result for an entire group of users. Personalization on *Time* context has been left out of the design as this proved to be a less interesting area to focus on here. This is because when watching TV, it is very likely that you are in a group of people, all with their own preferences. Having the system not recommend items for only a single user, but for the whole group is something that we considered to be essential for our system.

We filter the items for a group by calculating the above described preferences $p_0 \dots p_{c\alpha}$ for each user individually. Then we use a strategy described in Masthoff [72] to merge these individual values into group preference values $gp_0 \dots gp_{c\alpha}$. This will produce result r', which contains a list of content items, together with the prediction value of the group's preference for each item.

Step 6: Present the personalized result to the user

Finally, the result r' can be presented to the user by sorting the resulting list by decreasing prediction value and possibly by restricting the list to only contain a certain maximum number of elements. Furthermore, a presentation can be generated personalized for the user and targeted at a certain device the user is using. However, as this is again a very large research area, which requires a lot of detailed knowledge on how to achieve this, we considered this step to be outside the scope of our design. We recommend that this step is done to achieve the optimal result, but for now we suffice by having a default presentation for each query.

Personalization without query refinement

The above described personalization process handles the case in which a user searches for keywords, query 1 (section 4.4.1). However, the other two queries are handled in a different way.

Query 2: Recommend a set of items This query is not refined with ontologies, but is directly sent to the collaborative filtering process with a query for recommended packages. This is because the query is simply a request for recommended items, that does not need the refinement. The collaborative filtering process will compute a set of most recommended packages that are most popular in the user's community.

Query 3: present me this package Like with query 2, we do not refine the query and we filter the package's contents based on the user's preferences for the items of the package. We use the same process for this step as designed for query 1 (section 4.4.1 step 5). In general, this query can be seen as query 1, without the initial 4 steps, as we already have the result set available (the package's contents).

4.4.5 Updating the user model

After the queries sent by the user return a personalized result, the system should learn from the user by observing the actions the user takes. The user model(s) of the active users should be updated with the extra information that can be derived from these actions. We have designed a simple updating of the user model which takes note of the following actions.

- View an item
  When a user selects a certain content item or package for viewing, we assume that he is interested in this package. Therefore, we will register this action and store it in the user's Short Term Conceptual State (STCS) (see section 3.2.1).

- Rate an item
  Furthermore, we offer the possibility that a user explicitly rates an item using a thumbs-up and -down system. This action is also registered and added to the STCS on the Blu-IS server.

If a user ends a session (logs out of the system), the information contained in the STCS is used to update the corresponding user(s) user model(s). With only a single user logged in, we do this by updating the preferences for each rated or viewed item individually. With multiple users logged in, the problem arises on how to determine if a group action is really agreed on by all users.

For our approach we assume that a group will decide by itself the best possible action it can take to please everyone, or at least the majority of the group. We assume that when a user watches some content with a group, he automatically likes it, unless explicitly rated otherwise. If a user does not like some content, he will simply not watch it and walk away to do something else. This also corresponds with one of the popular techniques of [72], the "average strategy with least misery". The learning behavior is easy in this case: User Models of the individuals can be updated with view actions for items that the group watches. If however some users do not like what the group watches and still stay with that group, the damage that is done to the user models is relatively small, because groups in our systems are small (one household) and the majority of the group will probably like what is being watched. Furthermore, the user can simply disapprove or ignore a future recommendation that is made based on this error in his user model. The system can then learn from this mistake. Bringing such a small error into a user model is also not a bad thing because there are actually techniques that propose to improve the system's understanding of a user by doing this [81].

4.5 System architecture

The architecture of the Blu-IS system is described using the UML modelling method [76]. this means that we design the system using the following steps.

- Use Cases
- Component View
- State-Transition Diagrams
- Deployment View

4.5.1 Use cases

In the Blu-IS system we consider six different use cases, which are specified in this section. The use-case description is a generalized description of how the system will be used, with the purpose to provide an overview of the intended functionality of the system.

- Use case 1: User registration
  When a user interacts with the Blu-IS platform for the first time, the user needs to register himself to the box. Therefore the user enters some personal information and fills a short questionnaire, which will be used to initialize his user model. When the user is registered, the platform will automatically login the user.

- Use case 2: Login into the system, with a group of users
  When there are multiple users registered at the system, the user selects using his remote control, the users that are currently going to use the system.

- Use case 3: Search for package
  In the platform there is a possibility to search for a package. When an user enters some keywords, based on those keywords a search will be performed. Based on the preferences and his viewing history the system will sort and filter the resultlist, which is most convenient for the user. For each result, the available metadata can be retrieved by the system.

- Use case 4: View/Rate a Package
  When a result is displayed (e.g. after a search or recommendation), the system makes it possible to View the package, this will result in a overview of the package. when the user likes the package, the user can specify this by rate it higher (or lower is the user didn't liked it). Based on those ratings, the BLu-IS system gets 'smarter'.

- Use Case 5: Recommend packages
  In the Blu-IS platform, after the user is logged in into the system, it is possible to get recommendations for packages, without entering keywords. Based on the preferences and history the platform is aware of, the system calculates which packages the user will like.

- Use Case 6: Logout
  When the user (or users) logs out, all actions that the user(s) performed in the last session are analyzed and put into the corresponding user model(s). After the updates, the login screen of the system is displayed.

4.5.2 Context View

We first present a high level context view, which will show the Blu-IS server as a black box and showing all external connections that the server depends on. After that we will show the internals of the Blu-IS server. Figure 4.20 gives a clear picture of the context that the Blu-IS server is in and shows the external components and how these components are connected to the system. We will now describe each of these components in detail.

External Components

System services

The prototype makes use of two "system services". These are the *recommendation* and *user model service*. These services are treated as external components, because the recommendation system will use collaborative filtering strategies, which will require that the user models of multiple users are gathered. For the same reason, the user model service is also an external component.

- **Usermodel service**
  The *user model service* stores and updates all user models of all users of the Blu-IS system.

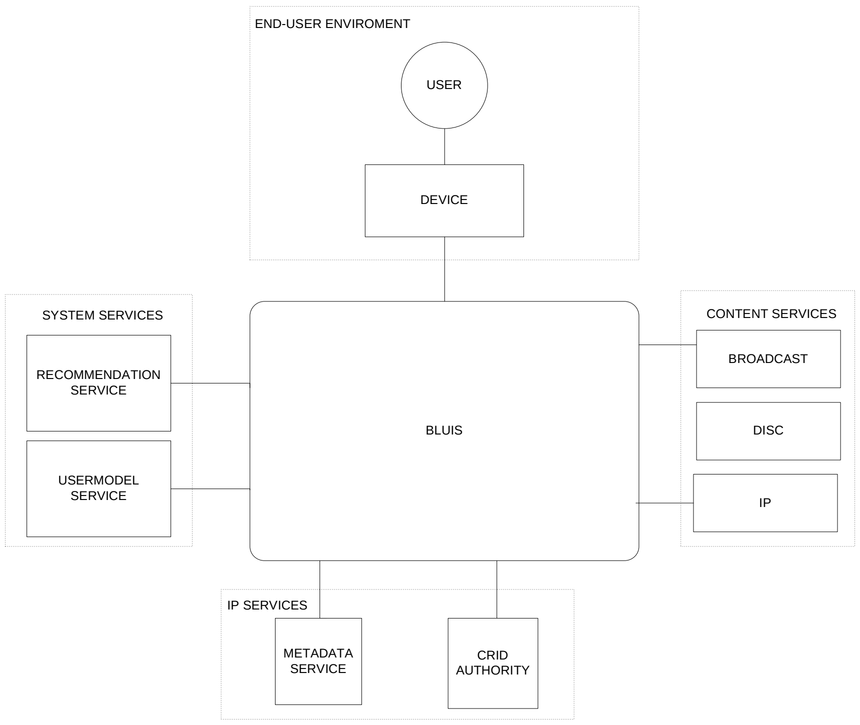


Figure 4.20: High level overview of the system

The prototype communicates with the *user model service* to request information about a certain user or to send information about a user to the *user model service* to update the user model. The advantage of an external *user model service* is that the user model could also be used with systems other than Blu-IS, for example a mobile phone. This way even more information can be used to update the user model, from which the personalization techniques will benefit.

- **Recommendation service**
  The *recommendation service* uses all available user models (for all Blu-IS-boxes) to create recommendations for content and packages. Furthermore, the *recommendation service* can filter result sets for certain criteria and interests of a certain user.

IP Services

The prototype makes use of packaged content (extended TV-Anytime phase 2 **??**). The content inside the packages is referenced with the use of Content Reference Identifiers (CRIDs). To search for packages, a connection to an (external) *metadata service* is necessary. When a package has been found with a search and when a package is retrieved, the CRIDs of the components inside the package need to be resolved to acquire locators for content. This is done at another IP service, the *CRID-authority*.

- **Metadata service**
  The *Metadata service*, as described by the TV-Anytime forum [7], is an external service for the Blu-IS system. The Blu-IS system can send a request for metadata for a certain CRID. The *metadata service* will then send the available metadata on this CRID back as response. Furthermore it is possible to request a certain type of metadata (descriptive, technical, review) for a certain CRID. Furthermore, we will extend the *metadata service* as

described by the TV-Anytime Forum, so that it can also handle more elaborate queries for metadata such as a request for metadata which matches some set of keywords.

- **CRID Authority**
  The *CRID Authority* is an external service, which can be used to resolve CRIDs to other CRIDs and/or Locators. Furthermore, it is possible to create or modify the CRID resolving tree at the authority. The prototype communicates with the *CRID authority* when CRIDs of a package need to be resolved (in order to retrieve content). The modification and creation of CRIDs is done at the authority and not at the Blu-IS box.

Content services

The prototype can receive content via three different distribution channels. These three channels together are called the “content services”. Each distribution channel has its own specific properties.

- **Broadcast**
  The *broadcast* channel will mainly be used to receive regular TV-programs. The obvious disadvantage of this channel is that content can only be received, no data can be sent back, which would allow for interactivity.

- **IP**
  The *IP channel* can offer any type of content. Furthermore, data can be sent back (the so-called return-channel), which makes the *IP channel* well suited for use in interactive applications. Multiple protocols have been specified, which describe how data can be retrieved from an *IP channel*. The prototype will support the most commonly used protocols http, ftp and streaming media.

- **Optical Disc**
  An *optical disc* will primarily be used to distribute High Definition content (HD-content), as the *broadcast* and *IP-channel* both are not very well suited for this task (yet), because of the necessary bandwidth.

End-user environment

The end-user environment consists of the *devices* and *users*, that are communicating with the prototype. The devices communicate through a client application (over IP) with the prototype. The devices allow the user to interact with the prototype.

- **Device**
  A client application runs on the *device* (PC, TV, handheld). This can be a Java application (if the device supports Java), but it could also be a web browser, depending on the devices capabilities. The application on the device uses asynchronous communication between user and the prototype system.

- **User**
  The *user* in the system communicates with the system through one or multiple devices at the same time. The system identifies the user to be able to realize personalization.

4.5.3 Component View

Figure 4.21 presents the internal components of the Blu-IS-box and the connections to the other services. In this section, each component is described and its internal structure is specified in an UML-Class Diagram. All interface descriptions of the individual components are included in appendix C.

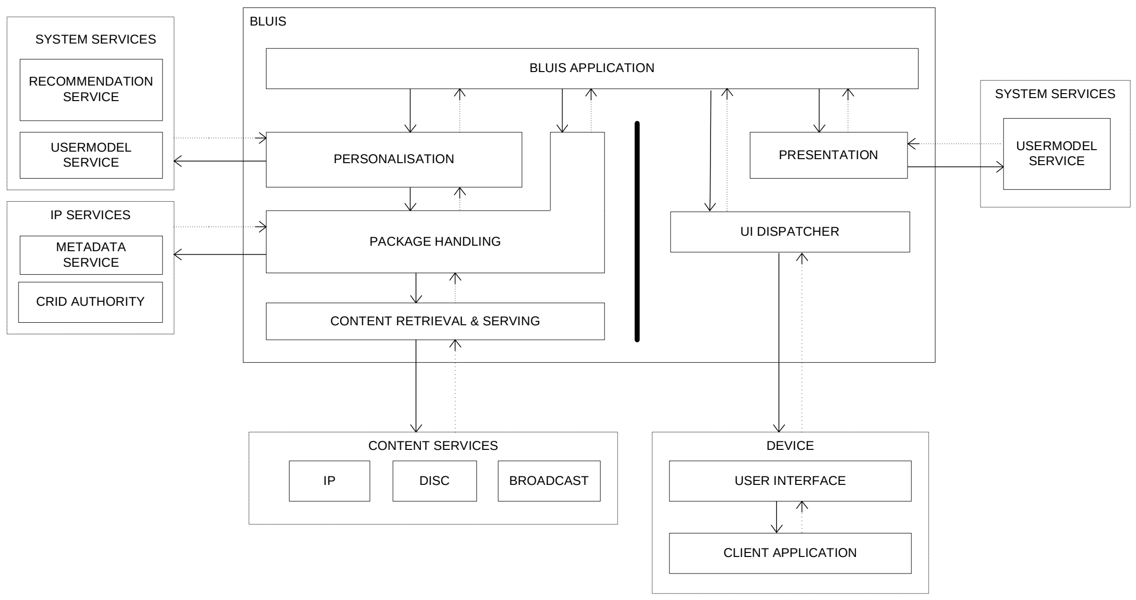


Figure 4.21: System Decomposition

Components of Blu-IS

Content Retrieval & Serving

This component (figure 4.22) handles the serving and retrieving of content. The component can retrieve content from either three distribution channels, which is indicated by the provided locator. If the content is not directly available, i.e. it has to be downloaded first, than the component sends an event to indicate that the content is available. The component is also responsible for serving content that the user has published. Incoming connections are accepted, requests for locally stored content are handled and content is sent back.

Features

- Retrieve content from an online location
- Retrieve content from a broadcast channel
- Retrieve content from a disc
- Store and cache content locally on the box
- Retrieve locally stored content
- Serving content which is stored locally on the box
- Recognize an optical disc, by calculating the digital finger print of the disc

Package Handling

This component (figure 4.23) provides the main functionality for performing operations on packaged content. It basically allows the system to work with packaged content. The *Package Handling* component communicates with the external IP Services concerning metadata descriptions and queries of package CRIDs. It enables the system to work with and parse packaged content as well as editing or making new. Naturally, the *Package Handling* component keeps track of a

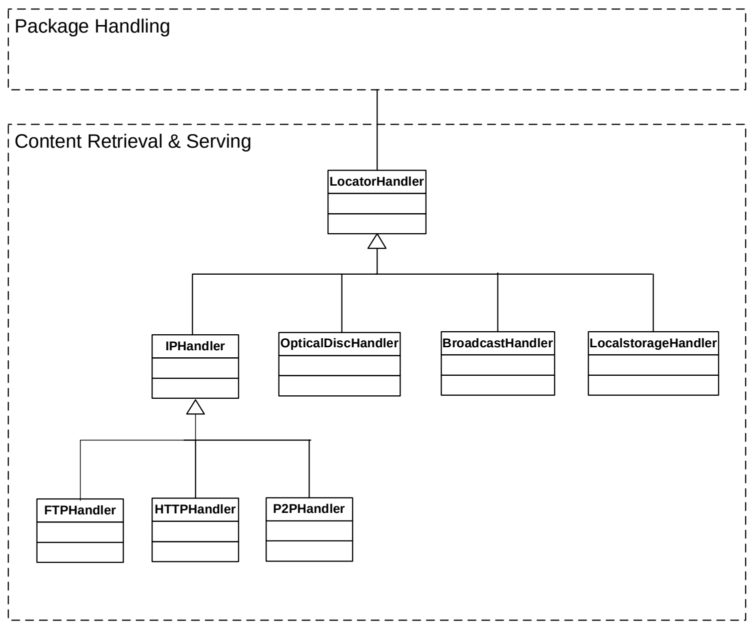


Figure 4.22: Content Retrieval & Serving Component

CRID for locally stored or cached content to reduce communication. Furthermore, for any content that a user wants to publish, the PH communicates with the CRID Authority to receive a unique CRID and make this public. Thereafter, other systems can retrieve it via the *Content Retrieval & Serving* component.

Features

- Converts the Package XML files to Objects and vice versa.
- Sends keywords to the metadata service and receives a list of packages.
- The following functionality for packages, content and metadata:

  search

  retrieve

  store locally

  delete locally
- create new packages and metadata
- modify existing packages and metadata
- publish packages and metadata

Personalization

The *Personalization* component (figure 4.24) has been described in detail in section 4.4.4 and we therefore only mention here that it communicates with the *Package Handling*, *System Services* and the Blu-IS *Application* components. The internal components are basically a split-up of the different personalization techniques and steps of the personalization process. First, the *QueryRefiner* is used to extend the query, using *ontologies*, which are handled by the *Jena* parser. The refined query is then sent to the *Metadata Service* by the *Metadata Handler* component, which

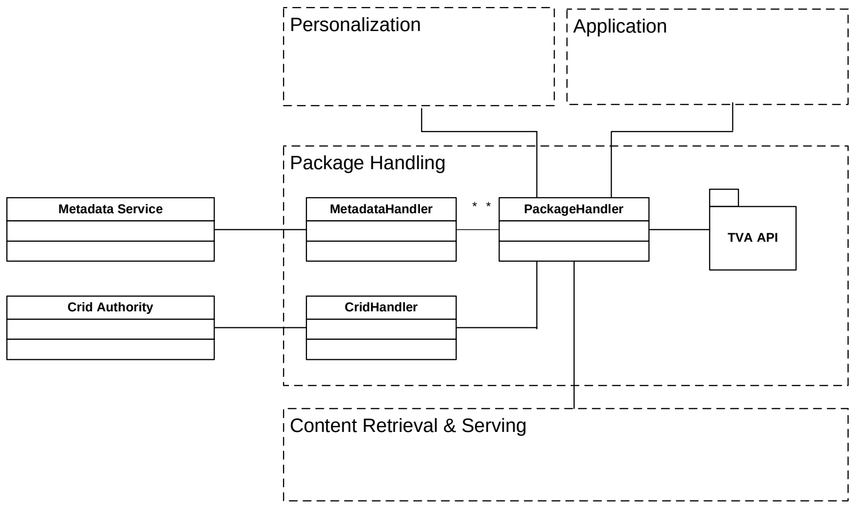


Figure 4.23: Package Handling Component

links to the *Package Handling* layer to retrieve this result set. The result set is then filtered using a number of filtering techniques. The user models of all users are stored on the *User Model service*, which can be accessed by the *User Model Handler*. The information from the user model is used to determine the Local and Community prediction values, which are taken care of by the *LFClient* and *CFClient* (connected to *Collaborative Filtering Service*) class respectively. *Group Strategies* can then be used to merge the individual preferences into predictions that are targeted for the entire group. The *PersonalizationClient* is the class that handles all communication between these components and which guides the personalization process.

Blu-IS Application

The BluiIS *application* component (figure 4.25) is the “heart” of the packaging system. This component connects all the components together and contains the business logic for the application. A session manager is used to keep track of all active users and devices.

Features

- Handles the communication between the different components
- Base functionality for the system
- Session management

Presentation

The presentation component (figure 4.26) generates the presentation for a certain application. The presentation can be personalized for a device, using pre-defined templates. The different devices include TV, PC, PDA and Mobile phone. As it is linked to the System Services, the user model can be used in this process to retrieve a user’s preferences of which device to use or the look-and-feel using templates.
Features

- Generate User Interface
- Adapt interface based on delivered data and device capabilities

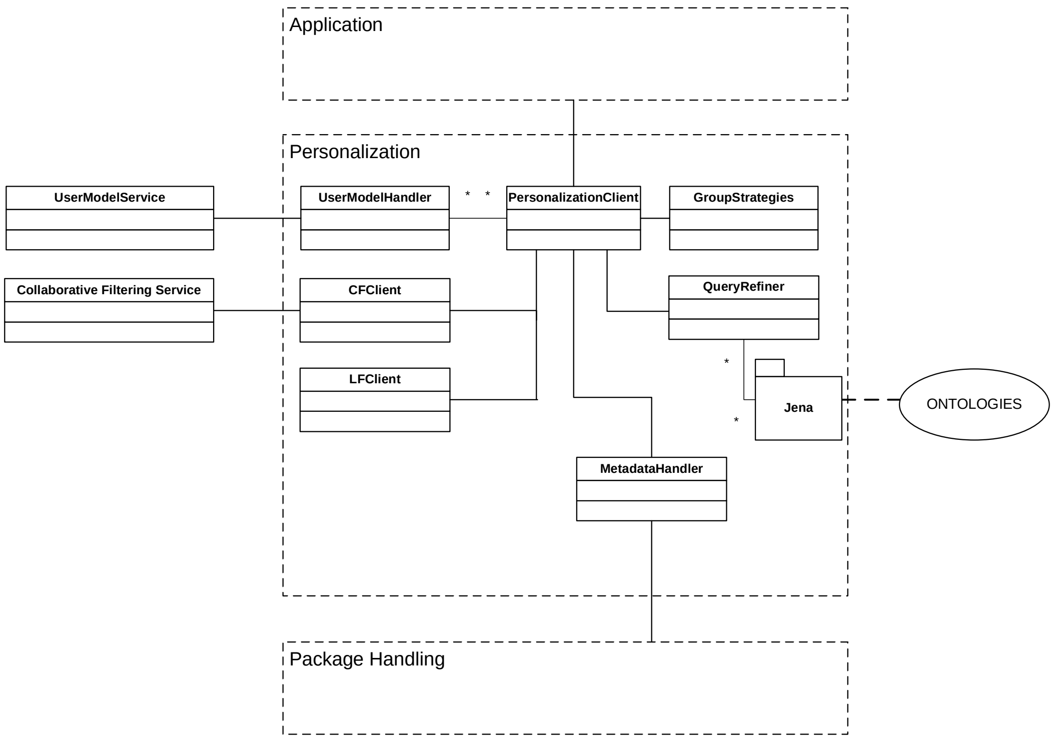


Figure 4.24: Personalization Component

User Interface Dispatcher

The communication with client applications running on connected devices is based on a request-response model via a *User Interface Dispatcher*. The *UI Dispatcher* (figure 4.27) will forward requests from the client to the Application component for processing and responds by sending generated presentations or data back to the Client application.
Features

- Initiates, closes and maintain connections between device and box
- Handles two-way communications between connected devices and the box
- The component makes a request-response client/server behavior

Client Application

The *Client Application* runs on a device and allows the user to interact with the system. This component uses a request/response model to communicate with the Blu-IS box; It sends all requests of the *client application* to the Blu-IS box and handles the response that the box sends back. The *Client application* provides the user with a User Interface, which is the presentation generated by the *presentation* component.

Features:

- Provides the user with a user interface through which he can interact with the Blu-IS box
- Handles all communication with the box.

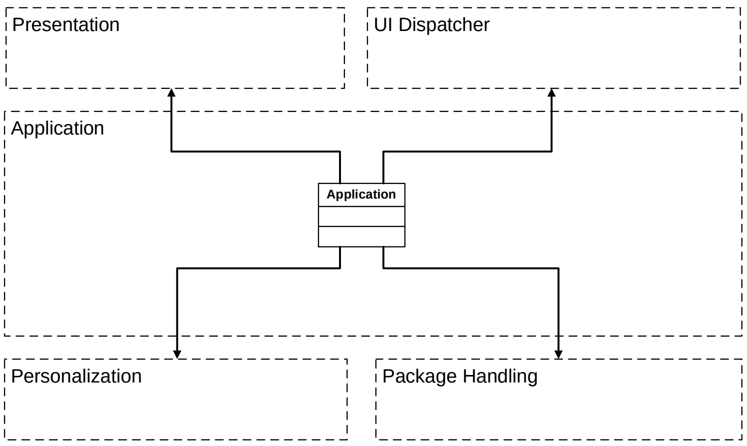


Figure 4.25: Application Component

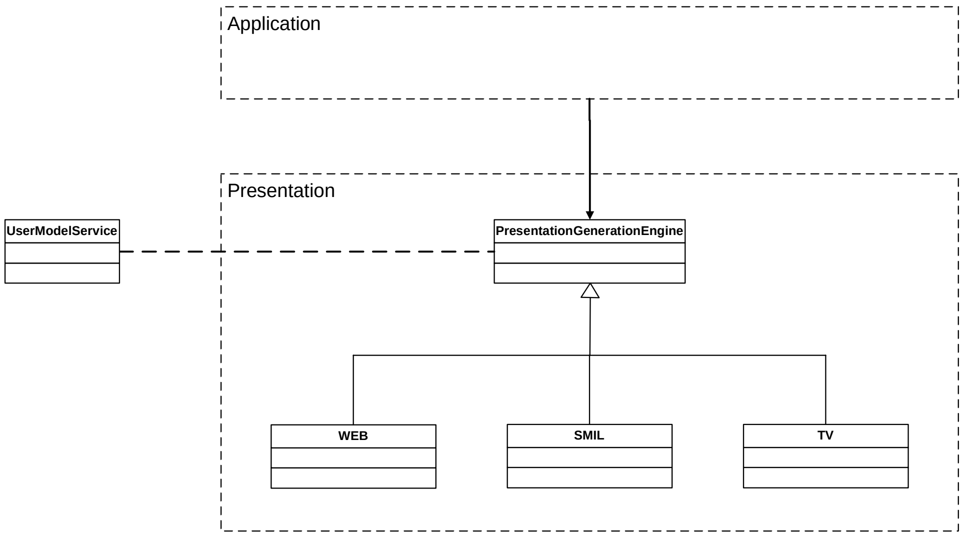


Figure 4.26: Presentation Component

4.5.4 State-Transition diagram

State transition diagrams have been constructed for each use case and are presented in appendix D.

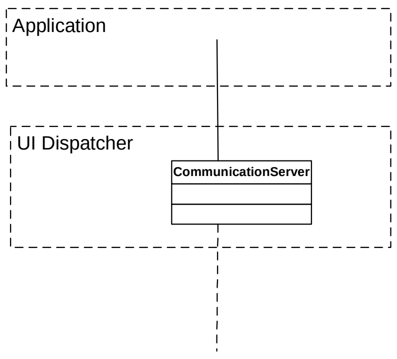


Figure 4.27: UI Dispatcher Component

Chapter 5

Implementation

This chapter will describe how a prototype for the suggested system has been implemented. First there is a description of the functions that the prototype offers and with what techniques this has been realized. Then an evaluation of the performance of this prototype is discussed. In the prototype that is created, not all the functionality, which has been described in the system architecture, has been implemented. The open issues are left as recommendations for further work.

5.1 Prototype

The prototype has been implemented entirely using Sun Java 1.5 [26]. The choice for the Java programming language is based on the requirement that the prototype must be able to run on different operating systems (MS Windows, Linux). The Blu-IS "box" has been simulated by a PC with a Tomcat Server running the Blu-IS application loaded as a servlet. Java Web Start technology has been used to allow a client (PC, TV) to connect to this "box". Furthermore, when the BLu-IS box itself, with connected TV, is used to run the client application, an infrared receiver allows the user to interact with the prototype using a remote control device, which interacts with the system using the jLirc interface [27].

The system has a client-server architecture, which can be programmed in several ways (using RMI, SOAP, sockets, Servlet). For the prototype we chose for the Java Servlet technology, because it is a desire to also connect web clients to the prototype (instead of only Java-enabled clients). For implementing the prototype, we have used several existing technologies and libraries. Figure 5.1 shows an overview of the implementation, mapped on the high level overview shown in figure 4.20. In the next section an overview will be given of all used technologies and the external libraries which are used, and which task they perform.

5.1.1 Programming languages and libraries

Sun Java 1.5

Java [26] is an object-oriented programming language developed by Sun. It shares many superficial similarities with C, C++, and Objective C (for instance for loops have the same syntax in all four languages); but it is not based on any of those languages, nor have efforts been made to make it compatible with them. The language was originally created because C++ proved inadequate for certain tasks. Since the designers were not burdened with compatibility with existing languages, they were able to learn from the experience and mistakes of previous object-oriented languages. They added a few things C++ doesn't have like garbage collection and multithreading; and they threw away C++ features that had proven to be better in theory than in practice like multiple inheritance and operator overloading.

Java was designed from the ground up to allow for secure execution of code across a network, even when the source of that code was untrusted and possibly malicious. This required the

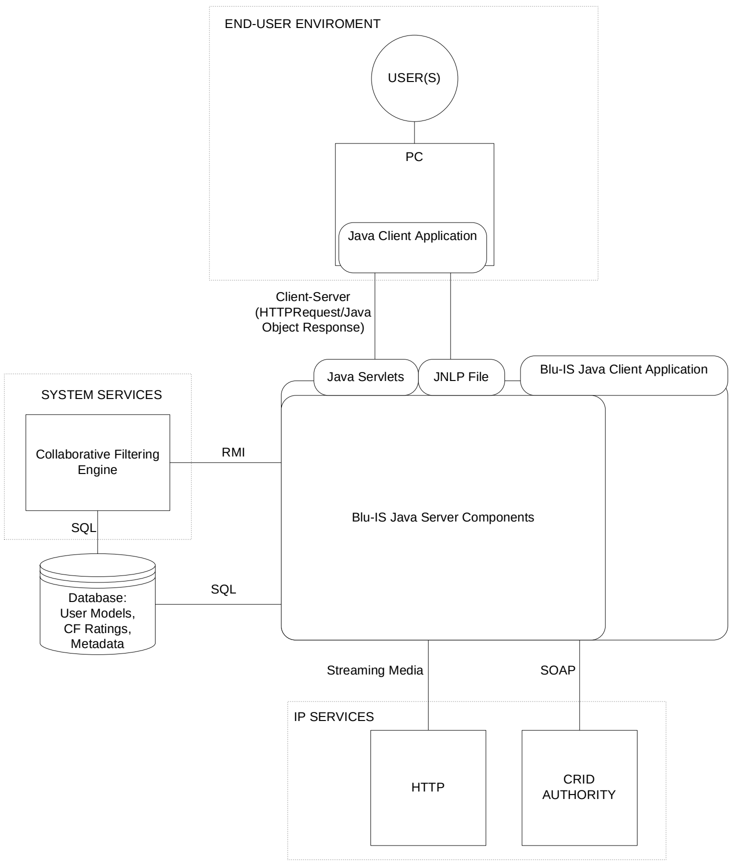

END-USER ENVIROMENT
USER(S)
PC
Java Client Application
Client-Server (HTTPRequest/Java Object Response)
Java Servlets
JNLP File
Blu-IS Java Client Application
SYSTEM SERVICES
Collaborative Filtering Engine
RMI
Blu-IS Java Server Components
SQL
Database: User Models, CF Ratings, Metadata
SQL
Streaming Media
SOAP
IP SERVICES
HTTP
CRID AUTHORITY

Figure 5.1: An overview of the implementation

elimination of more features of C and C++. Most notably there are no pointers in Java. Java programs cannot access arbitrary addresses in memory.

Furthermore Java was designed not only to be cross-platform in source form like C, but also in compiled binary form. Since this is frankly impossible across processor architectures, Java is compiled to an intermediate byte-code which is interpreted on the fly by the Java interpreter. Thus to port Java programs to a new platform all that is needed is a port of the interpreter and a few native code libraries.

Java Web Start

Java Web Start [26] provides a platform-independent, secure, and robust deployment technology. It enables developers to deploy full-featured applications to end-users by making the applications available on a standard Web server. By using any Web browser, end-users can launch the applications and be confident they always have the most-recent version. In our prototype, Java Web Start is used to launch the client applications from any PC, which has internet access.

Java Servlet

A Servlet [26] is a Java programming language class used to extend the capabilities of servers that host applications accessed via a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by Web servers. For such applications, Java Servlet technology defines HTTP-specific Servlet classes. In our prototype, a Java Servlet is used to give access to the Blu-IS applications for clients.

Java Desktop Integration Components

The Java Desktop Integration Components (JDIC) [28] provides Java applications with access to functionalities and facilities provided by the native desktop. It consists of a collection of Java packages and tools. JDIC supports a variety of features such as embedding the native browser, launching the desktop applications, creating tray icons on the desktop, registering file type associations, creating JNLP installer packages, etc.

JDIC is used in our prototype to embed media elements (video, audio) in HTML pages using the native browser of the client using the native media player (Windows Media Player, VLC Player). This allows for a variety of media types (avi, mpeg, wmv, asf, mp3, wav, etc) to be supported by the prototype. Media elements can be played streaming from the internet in this way. The advantage of using these components is that a lot more media types can be played using a faster, more reliable media player as compared to native Java media players like the Java Media Framework (JMF). After some bad experiences with this JMF (slow, lots of crashes, few media types supported), we decided to look for something else, which resulted in JDIC.

Java LIRC

jLirc [27] is a java API and tools for the LIRC and WinLIRC infrared remote control packages. jlirc lets developers add support for infrared remote controls to java applications. We have used this jLirc API to allow a user to operate the Blu-IS client using a remote control to simulate the TV viewing style.

Windows Media Player

Windows Media Player 10 [29] is used in the client application to view streaming media, when using the Microsoft Windows PC client. On Linux machines, the VLC Media Player is used.

VLC Media Player

VLC (initially VideoLAN Client)[30] is a highly portable multimedia player for various audio and video formats (MPEG-1, MPEG-2, MPEG-4, DivX, mp3, ogg, ...) as well as DVDs, VCDs, and various streaming protocols. It can also be used as a server to stream in unicast or multicast in IPv4 or IPv6 on a high-bandwidth network.

5.1.2 Applications

Apache Tomcat Webserver

Apache Tomcat [32] is the servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed by Sun under the Java Community Process. Apache Tomcat is used in our system to load the Servlets necessary for server-client communication.

Lirc

LIRC [31] is a package that allows you to decode and send infra-red signals of many commonly used remote controls. LIRC offers a daemon that will decode IR signals received by the device drivers and provide the information on a socket. It will also accept commands for IR signals to be sent if the hardware supports this.

MySQL

MySQL [33] is an open source RDBMS that relies on SQL for processing the data in the database. MySQL provides APIs for the languages C, C++, Eiffel, Java, Perl, PHP and Python. In addition, OLE DB and ODBC providers exist for MySQL data connection in the Microsoft environment.

MySQL is most commonly used for Web applications and for embedded applications and has become a popular alternative to proprietary database systems because of its speed and reliability. MySQL can run on UNIX, Windows and Mac OS.

Apache Web Server

Apache Web Server is used in our prototype to host the CRID-resolving process of the CRID Authority.

5.1.3 Communication Interfaces

Client-Server communication

The communication between client application and Java Servlet is a of request-response type. The application sends a HTTP POST request, which the Servlet handles, whereafter a Java Object stream is sent back to the client.

SOAP

SOAP (Simple Object Access Protocol)[34] is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses.

RMI

Java-RMI [25] is a Java-specific middleware spec that allows client Java programs to invoke server Java objects as if they were local. Java-RMI is tightly coupled with the Java language. Hence there are no separate IDL mappings that are required to invoke remote object methods. This is different from DCOM or CORBA where IDL mappings have to be created to invoke remote methods. Since Java-RMI is tightly coupled with The Java Language, Java-RMI can work with true sub-classes. Because of this, parameters passed during method calls between machines can be true Java Objects. If a process in an RMI system receives an object of a class that it has never seen before, it can request that its class information be sent over the network.

5.1.4 Operating system

Debian Linux OS

The Debian Project is an association of individuals who have made common cause to create a free operating system. This operating system that we have created is called Debian GNU/Linux, or simply Debian for short.

Debian systems currently use the Linux kernel. Linux is a piece of software started by Linus Torvalds and supported by thousands of programmers worldwide.

A large part of the basic tools that fill out the operating system come from the GNU project; hence the names: GNU/Linux and GNU/Hurd. These tools are also free. Debian comes with over 15490 packages (precompiled software that is bundled up in a nice format for easy installation on your machine).

Microsoft Windows 2000/XP

The prototype also runs on Microsoft Windows 2000/XP, because the system is programmed in Java.

5.1.5 Data

User Models

The user models are stored in a database, which contains personal information on the user like age, gender, occupation, etc. as well as information on preferences for TV-Anytime genres. Furthermore, all explicit ratings of the collaborative filtering process are stored. The User Model service has been omitted in the implementation of the design. Instead, a direct connection to an external database is made by the Blu-IS server.

Metadata

Because in the project there was no partner, which could provide us with content, we constructed the metadata by ourselves. One common known database for recommendations is the Eachmovie dataset. This dataset contains 1 million ratings from 3000 users over 4000 movies. This dataset is often used when a prototype is made that includes collaborative filtering techniques. Unfortunately we also needed much metadata of each movie for our personalization engine. We solved this issue to retrieve the movies from the Eachmovie dataset to the well-known Internet Movie Database. The Internet Movie Database is available for download and contains 500.00 movies, including a rich set of metadata.

5.1.6 Implementation of Query Refinement

The following OWL ontologies are used in the system. We have used Jena [35] for loading, parsing and reasoning on these ontologies.

WordNET ontology

We have used an OWL version of the official WordNET [46] ontology. This ontology was a bit too large (approximate 75 MB), to use it for the prototype for performance reasons. Therefore, we decided to delete everything from the ontology we did not use. After deletion of these concepts, we only have the relations *"synonyms"* and *"pertains to"* between concepts left. We use this ontology to find synonyms of an entered keyword and add these synonyms to the query.

TV-Anytime ontology

For the TV Anytime ontology we converted all the genres, which are specified by TV-Anytime [6], into an ontology. Because this ontology wasn't available, we derived one our self using the OWL language based on the XML-Schema [10] of the genres.

Geographical ontology

The geographical ontology from SUMO [47], contains each country of the world, grouped by its corresponding continent, like *"Europe"* and *"France"*. To add more functionality to this ontology we added the *"neighborOf"* relation to link neighboring countries.

5.1.7 Implementation of filtering

The filtering part of the personalization component consists of different personalization algorithm:

- Collaborative filtering prediction
  For the collaborative filtering component the open-source CoFE system (Collaborative Filtering Engine) [44] was used as a basis for the recommender system. The CoFE system implemented he recommendation algorithm as GroupLens [60] suggests. The CoFE system has a client-server architecture, and was programmed in the Java programming language, which makes it very suitable for integration in our project. Because it is an open-source

framework, we were able to add extensions and enhancements to the system. So we managed to connect our own database onto the CoFE system and restructured some major functions, to have a better performance.

- Query implementation
  We have only implemented the first two queries in our prototype: the keyword search and the recommendation feature. Package filtering has been left out here to focus more on the most important queries. After all, package filtering is basically the same as the keyword search, without the first steps of query refinement and query execution.

- Preference filtering
  For the filtering based on preferences we used the TV-Anytime genres. We converted the genres from XML-schema [10] to OWL [8], because in OWL it is possible to make a hierarchical structure, which is very useful for the TV-Anytime genres. The advantage of the hierarchical structure is that if the user likes "action", it implicitly likes all descendants of "action" (genres "War", "Adventure", etc).

5.2 Use Case implementation in the prototype

Use case 1: User registration

When a user interacts with the Blu-IS platform for the first time, the user needs to register himself to the box. Therefore the user enters some personal information and fills a short questionnaire, which will be used to initialize his user model. These questions are shown in figure 5.2 and figure 5.3. The questionnaire consists of two parts. The first part purely focusses on personal information on the user. The questions consist of:

- name
- gender
- age
- occupation
- zipcode
- avatar

This list is kept short on purpose to prevent the obtrusiveness of a long registration process, while the user just wants to get started. The information from this first page is used to match the user to a certain stereotype, which will initialize his user model. We currently use only the following stereotypes:

- **Woman with age < 18**
  Like genres: Romance, Comedy, Detective, Classical Drama, Adventure, Fantasy
  Dislike genres: Musical, Action, Horror, War, Science Fiction, Western, Effect movies

- **Woman with age >= 18**
  Like genres: Romance, Comedy, Musical, Detective, Classical Drama
  Dislike genres: Action, Horror, War

- **Man with age < 18**
  Like genres: Action, Horror, War, Adventure
  Dislike genres: Romance, Musical, Drama

- **Man with age >= 18**
  Like genres: Action, Non Fiction, War, Mystery, Effect movies
  Dislike genres: Romance, Horror, Drama, Musical

These stereotypes will set the corresponding genre *"likes"* and *"dislikes"* in the user model. A *"likes"* genre will receive value 5 (highest). A dislikes genre will receive value 1 (lowest).

The user can now choose to end the registration process after this first step, whereafter he is logged in to the system. However, he also has the option to continue the registration process and tell the system more about himself so that he will get a better personalized experience from the start. If he presses the *"next"* button, he will be confronted with the second part of the registration process, where he can set his allowed *certifications* (figure 5.3). These certifications are based on KijkWijzer [36] symbols, which allow the user to specify what type of content he would like to see or not. He can choose to enable or disable filtering out content with the following certifications. These certifications are linked to TV-Anytime genres (from the TV-Anytime `ContentCS` specification) and TV-Anytime Parental Guidance certifications, which they can filter out. We have chosen to match these icons to TV-Anytime genres and parental guidance certifications, because the data set we use for the prototype, the EachMovie data set, only uses genres and local parental guidance certification tags in their metadata. It would be better to have metadata that contained items from the `IntendedAudienceCS` specification of TV-Anytime and `ContentAlertCS`, but unfortunately this metadata was not available and we had to improvise by matching our Content Alert Icons to the TV-Anytime genres.

Certification	**Filters out:**
AL	NL:AL
MG6	NL:MG6
12	NL:12
16	NL:16
Violence	3.4.6.8: war 3.4.6: action 3.4.6.9: western 3.4.6.10: thriller 3.4.6.6: horror
Scary	3.4.6.10: thriller 3.4.6.6: horror 3.4.13: drama 3.4.6.7: science fiction 3.4.6.3: mystery
Sex Discrimination	3.4.8: erotica 3.4.6.8: war 3.4.6.9: western
Drugs	3.1.7.1: reality
Bad Language	3.1.7.1: reality

This way, the user is not confronted with content that he definitely does not want to see. We could also use this screen for parents to set the allowed content for their children, i.e. *parental supervision* options. The questionnaire can also be further extended by adding more screens to the registration process, like including the Motivaction [37] questions to really initialize the stereotype in a proper way. However, as this would require the user to answer over +- 100 questions, we considered this to be too obtrusive to be used in our system and we think this would hardly be used by any user of our system.

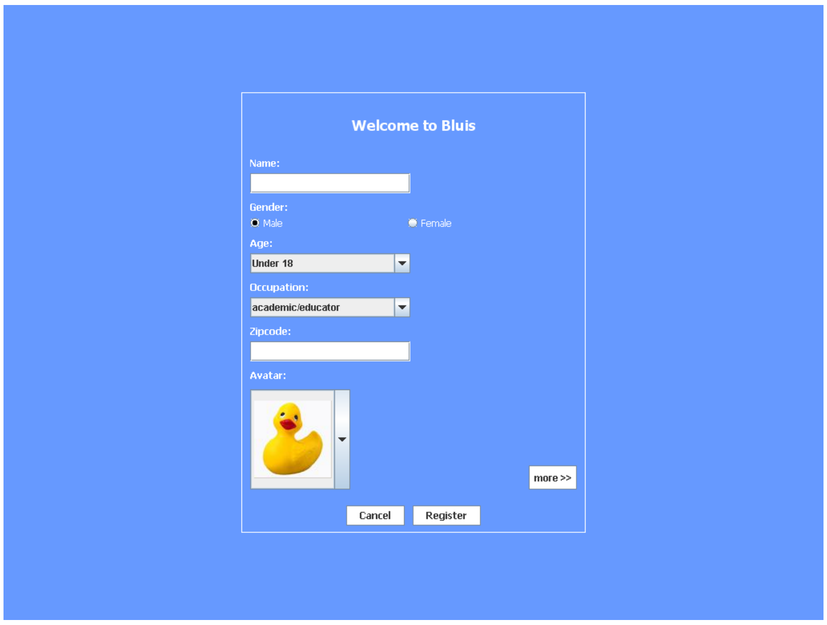


Figure 5.2: registration of Blu-IS

Use case 2: Login into the system, with a group of users

When there are multiple users registered at the system, the user selects the users that are currently going to use the system, using his remote control. This can be done in the login screen of the Blu-IS application, shown in figure 5.5. The interface of this login procedure is intuitive as the user just has to select the active users, which will then appear in a list on the right, together with a login button. After a group logs in, all search results are personalized for the entire group, rather than for a single user. The group filtering strategies that we have implemented are the three strategies that are described as best in the Masthoff paper [72]: Average, Average without misery and least misery strategy.

Use case 3: Search for package

In our prototype there is a possibility to search for packages or content elements using a keyword search. When an user enters some keywords, a search will be performed on the Blu-IS server. Based on the preferences and his viewing history the system will sort, filter and then present the resultlist, which is personalized for the active user(s) (figure 5.6. For each result, the available metadata can be retrieved by the system. The matching keywords for each content item are shown to the user to indicate why the content item is in the list. Furthermore, the extended query that has been obtained by using ontology inferencing is displayed.

Use case 4: View/Rate a Package

When a result is displayed (e.g. after a search or recommendation), the system makes it possible to *view* (figure D.6) the package, this will result in a overview of the package: a listing of all items and components the package consists of. A user can click on any of these items or components to see what content is available in that item or component. When a component is selected, a list of all available CRIDs and locators is shown in a tree, which can be expanded by the user, thus resolving the CRIDs. When a user double-clicks a content item, the playback starts using either Windows Media Player or VLC player depending on the operating system. The user can also explicitly indicate his preference by rating the package or its components high or low using the thumbs-up and -down icons. The system will register this explicit preference as a preference for

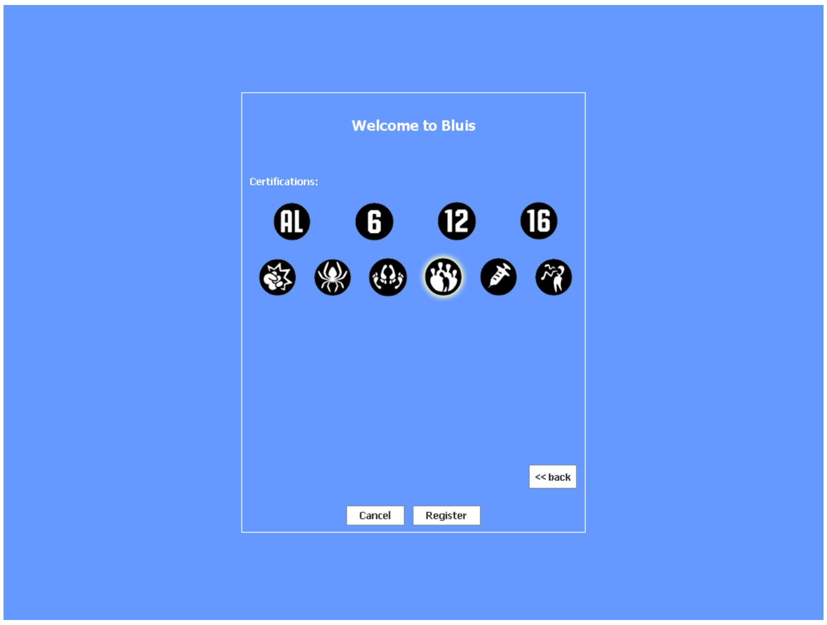


Figure 5.3: Certifications of Blu-IS

the TV-Anytime metadata classification accompanying the content element. In our prototype, we use the rating to update the preference for the TV-Anytime genres matching the package's genres. In a further implementation, this can be improved by registering the preference of more metadata elements. Based on those explicit ratings, the BLu-IS system will learn more about the user and update his user model after the user logs out.

Use Case 5: Recommend packages

In the Blu-IS platform, after the user has logged in to the system, it is possible to get recommendations for packages, without entering keywords. The system will calculate the top most recommended items using a collaborative filtering technique, which will result in the top rated items in the user's community. These items are presented to the user in a result list (figure 5.8).

Use Case 6: Logout

If a user presses the "Sign out" button of the user interface, he will log himself and the active group out. This will cause the system to end the active session, with all the actions that the users took (viewing, rating) logged. These actions are used to update the user model. In our prototype, we do this by updating the TV-Anytime genre preferences for all rating and viewing actions and we update the collaborative filtering rating database with the explicit ratings the user supplied.

5.3 Evaluation

We have tested the prototype by creating a set of different users, all with their own preferences. With these users, we have executed various personalization tasks. We have evaluated the results of these tasks by first looking at the quality of the proposed recommendations and personalization results, and secondly by looking at the time it took to execute and personalize a query.

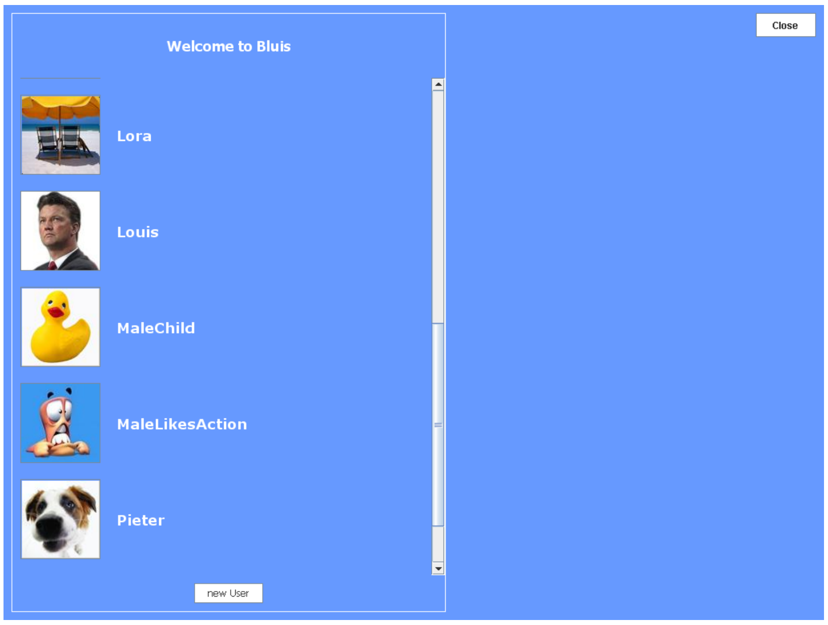


Figure 5.4: Loginscreen of Blu-IS

5.3.1 Quality

We have evaluated the quality of the given recommendations and ordering of content items in order to determine if these are actually what the user wants to retrieve. Letting a set of novice users test the system using a specified set of tasks would be the best way to determine this quality, because in the end, they will be the typical users of the Blu-IS system, which will be a personalized home media system in a typical family household. Furthermore, deciding whether a recommendation is suitable or not is also a matter of taste. Therefore it is better to use the opinions of real persons rather than deciding for ourselves what is "right" or wrong. However, as the available time for this evaluation was too short we decided for a simpler approach by using observations on the proposed recommendations by ourselves, as we could easily point out some of the problems we encountered during evaluation of the prototype. We have used the following case in evaluating the quality of the recommendations given by the prototype:

We log in to the system with a user that has registered using the following data.

- age 18-25
- Like genres: Action, Non Fiction, War, Mystery, Effect movies
- Dislike genres: Romance, Horror, Drama, Musical

Quality of keyword search results

We now enter a search for "french course" in the search interface, which retrieves a list of packages that match the query (figure 5.10).

The interface shows that the system has searched not only for the proposed keywords, but also for the country France and it's neighboring countries. Thus the query refinement process has succeeded in determining a suitable context for the word french. The next step is to execute the query at the metadata service. As can be concluded from this step, the searching algorithm at the metadata service is a crucial part in retrieving the right matching packages. For example, if only the metadata tags `title` or `synopsis` (from TVA:`BasicDescription`) are searched by the metadata service with the given keywords and if the tag `RecordedLocation` or `DepictedLocation` is not searched for, then critical packages may be not retrieved at all. Thus this step will be of huge

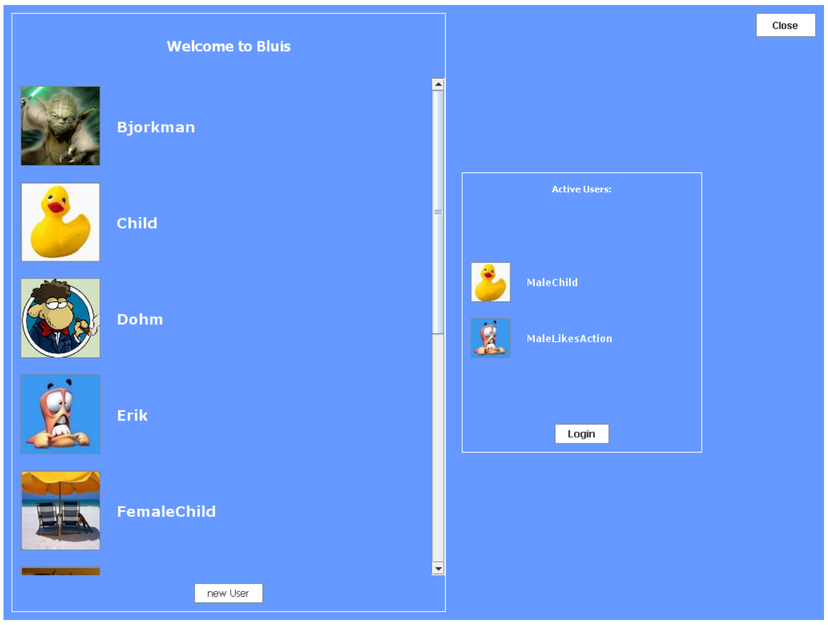


Figure 5.5: Group login

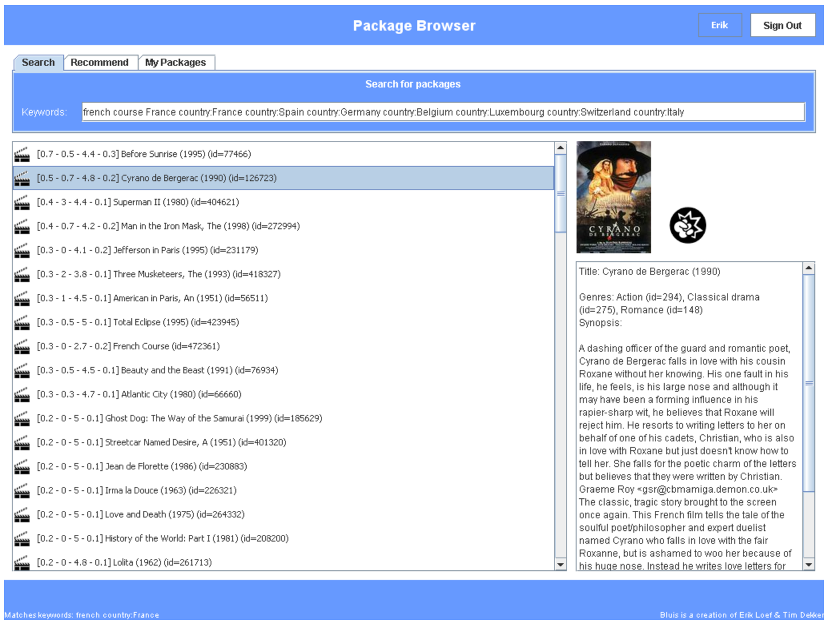
importance for all of the remaining steps in the personalization process. If matching packages are not retrieved and the metadata service returns a strange result set, then the end result will also contain strange items as the filtering step is unable to improve the result list obtained from the metadata service. Also, when the metadata service only hosts a small amount of packages, then it is possible that the returned results for a search are not exactly what the user wanted, because the matches that the user would like are simply not in the database at the metadata service.

If we look at the results that are returned for our search for “French course”, we can conclude that the database only contains one real French course package (named “French Course”) that we are actually looking for. The remainder of the list consists of movies that either occur in France or one of its neighboring country, or that contain one of the words “French, France or course” in the title or plot. So what we actually want in our result is the “French Course” item occurring at the top of the list and the rest of the items occurring later. The contrary is the case however, as our French Course occurs nearly at the bottom of the list with a large list of movies unrelated to French courses occur above it. We will examine how this is possible.

First we will look at how the list is sorted using the semantic closeness value. We see that the “French Course” matches the two keywords, “French” and “course”, thus giving it a semantic closeness value of:

$$\frac{sum\ of\ values\ of\ matching\ keywords}{total\ nr\ of\ keywords} = \frac{1+1}{10} = 0.2.$$

However, we see that for example item “Before Sunrise” has semantic closeness value 0.3, because it matches more keywords, which occur in the plot or the movie: “French, course and France” and because it plays in Switzerland (neighbor of France), giving it additional semantic closeness. However, the movie has nothing to do with an educational “French Course”, which we are actually looking for and which should have a higher semantic closeness. This suggests that the semantic closeness algorithm could use some improvement. In the current case, if an item does not match any keywords that we originally entered, but if it matches a lot of other, derived, keywords, it can occur in the list above an item that matches all of the originally entered keywords. So a better approach may be to always put items that match the originally entered keywords above items that do not match any of these keywords, but do match derived keywords. In some sense, we can see this new approach as showing the results of a normal “Google-like” search at the top of the list, and the “related items”, which only match derived keywords, below them.



french course France country:France country:Spain country:Germany country:Belgium country:Luxembourg country:Switzerland country:Italy

```
[0.7 - 0.5 - 4.4 - 0.3] Before Sunrise (1995) (id=77466)
[0.5 - 0.7 - 4.8 - 0.2] Cyrano de Bergerac (1990) (id=126723)
[0.4 - 3 - 4.4 - 0.1] Superman II (1980) (id=404621)
[0.4 - 0.7 - 4.2 - 0.2] Man in the Iron Mask, The (1998) (id=272994)
[0.3 - 0 - 4.1 - 0.2] Jefferson in Paris (1995) (id=231179)
[0.3 - 2 - 3.8 - 0.1] Three Musketeers, The (1993) (id=418327)
[0.3 - 1 - 4.5 - 0.1] American in Paris, An (1951) (id=56511)
[0.3 - 0.5 - 5 - 0.1] Total Eclipse (1995) (id=423945)
[0.3 - 0 - 2.7 - 0.2] French Course (id=472361)
[0.3 - 0.5 - 4.5 - 0.1] Beauty and the Beast (1991) (id=76934)
[0.3 - 0.3 - 4.7 - 0.1] Atlantic City (1980) (id=66660)
[0.2 - 0 - 5 - 0.1] Ghost Dog: The Way of the Samurai (1999) (id=185629)
[0.2 - 0 - 5 - 0.1] Streetcar Named Desire, A (1951) (id=401320)
[0.2 - 0 - 5 - 0.1] Jean de Florette (1986) (id=230883)
[0.2 - 0 - 5 - 0.1] Irma la Douce (1963) (id=226321)
[0.2 - 0 - 5 - 0.1] Love and Death (1975) (id=264332)
[0.2 - 0 - 5 - 0.1] History of the World: Part I (1981) (id=208200)
[0.2 - 0 - 4.8 - 0.1] Lolita (1962) (id=261713)
```

Title: Cyrano de Bergerac (1990)

Genres: Action (id=294), Classical drama (id=275), Romance (id=148)
Synopsis:

A dashing officer of the guard and romantic poet, Cyrano de Bergerac falls in love with his cousin Roxane without her knowing. His one fault in his life, he feels, is his large nose and although it may have been a forming influence in his rapier-sharp wit, he believes that Roxane will reject him. He resorts to writing letters to her on behalf of one of his cadets, Christian, who is also in love with Roxane but just doesn't know how to tell her. She falls for the poetic charm of the letters but believes that they were written by Christian. Graeme Roy <gsr@cbmamiga.demon.co.uk> The classic, tragic story brought to the screen once again. This French film tells the tale of the soulful poet/philosopher and expert duelist named Cyrano who falls in love with the fair Roxanne, but is ashamed to woo her because of his huge nose. Instead he writes love letters for

Matches keywords: french country:France

Bluis is a creation of Erik Loef & Tim Dekker

Figure 5.6: Search results

Furthermore, the same issue as described above occurs in a similar manner when the list is sorted based on derived predictions on both the user's preferences as well as on the communities preferences (like described in section 4.4.4). The thing that occurs here is that since our user likes the "action" genre and dislikes "education" topics, action movies are rated higher than the French course, that he is actually looking for. Again, this is not what we want. Just because the user likes "action" does not mean that he should be presented with action movies at the top of the list when he is searching for a "French course" to learn more about France for his upcoming holiday. As with the previous issue, adaptation of the semantic closeness algorithm and calculation of the total prediction value can help out here in preventing action movies, which only match derived keywords, from occurring above the French course item. However, if the text in the movie plot actually contains the words "French course" then the action movie still occurs above our educational French course item.

In this case, we need a better way of deriving the context of the search keywords. The system should be able to know that the "French course" keywords of our search give rise to an "educational" context, because of the word course. An ontology could contain this information, mapping the word "course" to the genre "education". However, if a user wants to search for an action movie, which is also named "French course", the system can likewise determine from an ontology that this name is linked to the "action genre" context. This gives the system a conflicting situation, as these genres are totally different from each other, which could be observed from inferencing in the "genre" ontology, specifying that these topics have nothing in common. In this case, the system could require the user to specify which context he is exactly interested in, or if he just wants to search for both contexts. If the user then selects the education context, the French course item will appear above all action movies in our result list. An alternative option is to simply remove all action movies from the list if the user has specified that he is looking for the "educational" context. This could be put to good use in our system in the following way.

If a user searches for some keywords, the entire result is presented in the way we have designed it. However, the user is now able to select the keywords that the system has derived from ontologies, to select of filter out the items that match these keywords. This way, the user can specify the intended context of the query.

A conclusion that we can draw from this evaluation is that the used ontologies are an important factor in deriving the query context. The more and the more rich ontologies we have available,

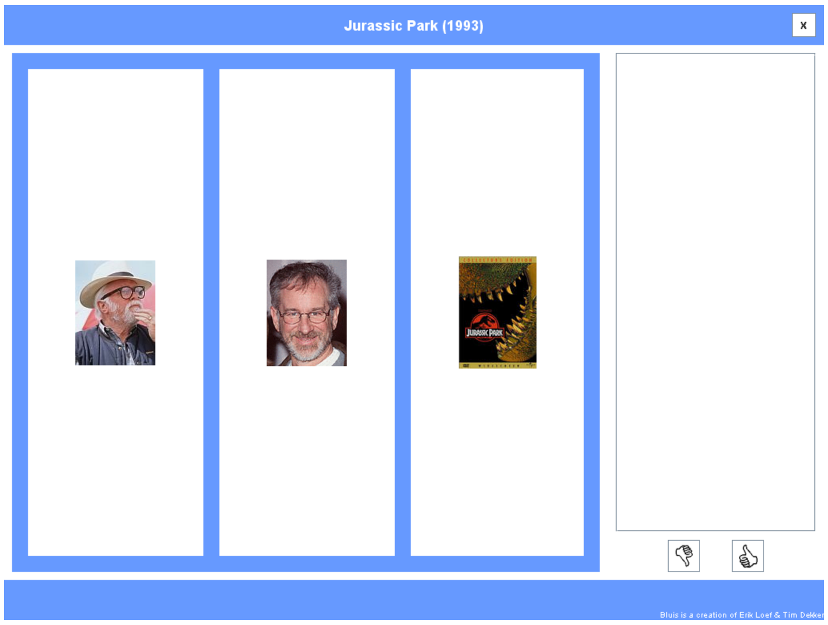


Figure 5.7: View/Rate package

the better the system is able to determine the context. We could improve the system if we add or extend the ontologies that we are using now.

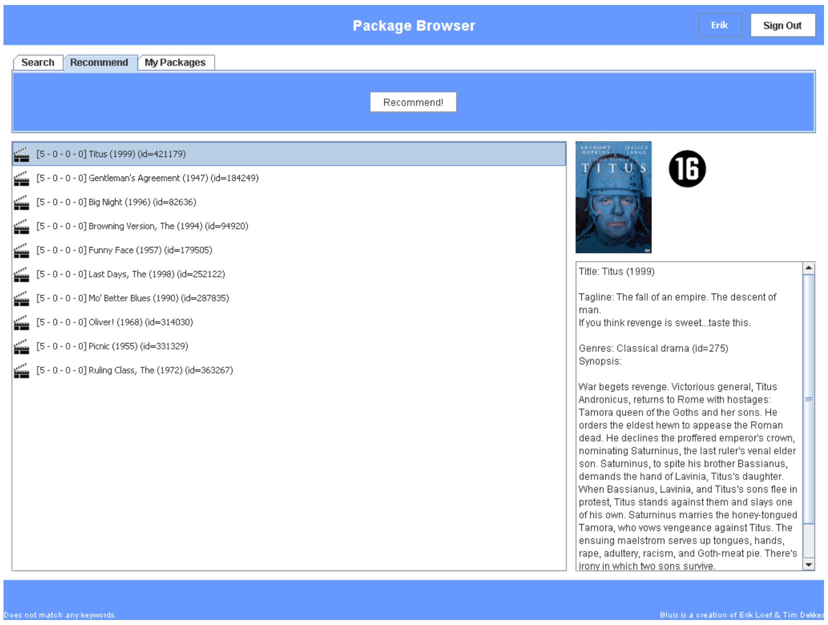
We can conclude that the *semantic closeness* value, which represents how close an item matches a user query is important and should be used to sort the items of the result list decreasing on how closely they relate to the initial search. Using this observation, we can update the design of our personalization process to first sort the result list descending on this semantic closeness value. After that, we can use the predictions to sort sections of items with the *same* semantic closeness value. This way, items are sorted based on how close they match the user's query, and furthermore, items that are matched evenly on semantic closeness are sorted by a prediction value, which is calculated (section 4.4.4) using both the individual user preferences and the community preferences. If we look at the result of the updated sorting algorithm (figure 5.10), we see that this gives much better results, the French Course we are looking for appears more to the top of the list, because items are sorted by semantic closeness first now, which more closely represents how much an item matches the initial search query.

The top result however "Before Sunrise", matches keywords "French, course and France" in the `synopsis` metadata tag and because its `depictedLocation` is Switzerland, a neighboring county of France. This causes it to achieve a higher semantic closeness value than our French Course that we want to find. If however we use more ontology inferencing, which derives the context "education" from the word "course", we could enable the user to select this context which will present the French Course to occur on top of the list as the other items in the list represent movies.

We can conclude that we should not mix the values of how much an item matches a query and how much an item is liked by the user too much. This can result in items which are not really related to the user's search to occur on top of the list, which is unwanted.

Quality of recommendations

Our prototype also offers functionality to request recommendations for packages. It does so by using the collaborative filtering technique to determine the user's community of similar users and return the top-rated packages for this community. The top ten of these packages are returned to the user and are presented as recommendations. Figure 5.11 show the results for this query. These results have been acquired after the user has first rated a few James Bond movies with a


Package Browser
Erik
Sign Out
Search
Recommend
My Packages
Recommend!
16
Does not match any keywords.
Bluis is a creation of Erik Loef & Tim Dekker

[5 - 0 - 0 - 0] Titus (1999) (id=421179)
[5 - 0 - 0 - 0] Gentleman's Agreement (1947) (id=184249)
[5 - 0 - 0 - 0] Big Night (1996) (id=82636)
[5 - 0 - 0 - 0] Browning Version, The (1994) (id=94920)
[5 - 0 - 0 - 0] Funny Face (1957) (id=179505)
[5 - 0 - 0 - 0] Last Days, The (1998) (id=252122)
[5 - 0 - 0 - 0] Mo' Better Blues (1990) (id=287835)
[5 - 0 - 0 - 0] Oliver! (1968) (id=314030)
[5 - 0 - 0 - 0] Picnic (1955) (id=331329)
[5 - 0 - 0 - 0] Ruling Class, The (1972) (id=363267)

Title: Titus (1999)

Tagline: The fall of an empire. The descent of man.
If you think revenge is sweet...taste this.

Genres: Classical drama (id=275)
Synopsis:

War begets revenge. Victorious general, Titus Andronicus, returns to Rome with hostages: Tamora queen of the Goths and her sons. He orders the eldest hewn to appease the Roman dead. He declines the proffered emperor's crown, nominating Saturninus, the last ruler's venal elder son. Saturninus, to spite his brother Bassianus, demands the hand of Lavinia, Titus's daughter. When Bassianus, Lavinia, and Titus's sons flee in protest, Titus stands against them and slays one of his own. Saturninus marries the honey-tongued Tamora, who vows vengeance against Titus. The ensuing maelstrom serves up tongues, hands, rape, adultery, racism, and Goth-meat pie. There's irony in which two sons survive.

Figure 5.8: User-based collaborative filtering recommendation

"thumbs-up". The listed recommendations show a clear preference for action movies.

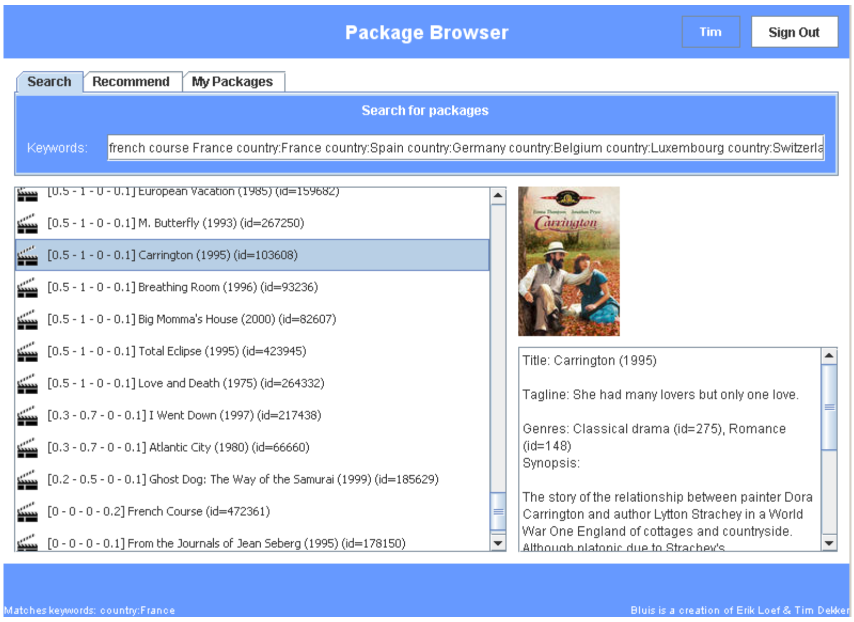
The results give a nice overview of what is popular at the moment in the user's community and provide an easy way for the user to view a package if he just wants to view a package without actually having a plan of viewing specific content. However, if the community does not perform any actions like viewing or rating content items, this list is very static. The proposed recommendations do not change in this case. In our prototype we only log in with this one user, which will not cause the recommendations of the community, a set of imported users from the EachMovie data set, to change. However, in real life the users are likely to use the system more than we do now with our prototype, which causes this list to change over time. However, for items to appear in this list, they must have been rated or viewed by the community. Newly released packages for example are not listed here until they already have become popular in the community. However, we could add another tab, which shows all recently (for some time interval), published packages, like in an ordinary EPG, but sorted by a prediction value based on the user's preferences, like done in the keyword search.

Quality of group queries

Now we log in with a group of users and look at what the results are. We use the "average" group strategy to calculate group predictions. We now log in with a group of users, of which the preferences of one user strongly contradict with the other. Take for example a parent, who wants to view content together with his child, who is only 6 years old. The child's certifications have been set by his parents so that he will only receive content that is classified using the AL (all ages) certificate. He is not allowed to view violence, scary, erotic or any other disturbing content types.

	Parent	**Child**
age	35	6
gender	male	male
allowed certifications	all	AL only

If now these two users log in at the same time, a common user model will be computed that will take note of both users allowed certifications. This will result in a merged set of allowed certificates, which will become the same as the child's set in this case: only AL is allowed. Other



Keywords: french course France country:France country:Spain country:Germany country:Belgium country:Luxembourg country:Switzerla

[0.5 - 1 - 0 - 0.1] European Vacation (1985) (id=159682)
[0.5 - 1 - 0 - 0.1] M. Butterfly (1993) (id=267250)
[0.5 - 1 - 0 - 0.1] Carrington (1995) (id=103608)
[0.5 - 1 - 0 - 0.1] Breathing Room (1996) (id=93236)
[0.5 - 1 - 0 - 0.1] Big Momma's House (2000) (id=82607)
[0.5 - 1 - 0 - 0.1] Total Eclipse (1995) (id=423945)
[0.5 - 1 - 0 - 0.1] Love and Death (1975) (id=264332)
[0.3 - 0.7 - 0 - 0.1] I Went Down (1997) (id=217438)
[0.3 - 0.7 - 0 - 0.1] Atlantic City (1980) (id=66660)
[0.2 - 0.5 - 0 - 0.1] Ghost Dog: The Way of the Samurai (1999) (id=185629)
[0 - 0 - 0 - 0.2] French Course (id=472361)
[0 - 0 - 0 - 0.1] From the Journals of Jean Seberg (1995) (id=178150)

Title: Carrington (1995)

Tagline: She had many lovers but only one love.

Genres: Classical drama (id=275), Romance (id=148)
Synopsis:

The story of the relationship between painter Dora Carrington and author Lytton Strachey in a World War One England of cottages and countryside. Although platonic due to Strachey's

Matches keywords: country:France

Bluis is a creation of Erik Loef & Tim Dekker

Figure 5.9: Search results for query "French Course"

content types are not shown in the result lists. Thus the group as a whole will not be confronted with content that is not suited for the child. For example, when searching for the word "horror", the system will identify the genre "horror" as the context of this word. However, all results that are rated as having the "scary" certification are filtered out to prevent this unsuitable content to be presented to the child. In our result set, only items that have the word "horror" in another context, which does not classify these as being scary are presented to the users.

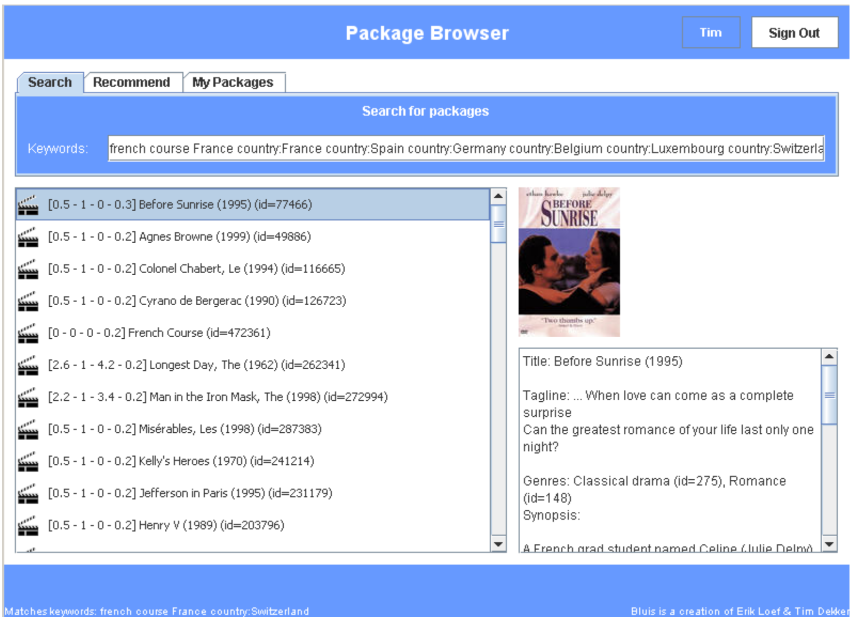
If the father decides he wants to view horror after all then he should change the allowed certificates for his child or just log the child out of the system if he does not want the child to view the content. Another option is to indicate to the users that some items have been filtered out due to the allowed certifications for some group members. Then if the group decides to view these after all, these results could be presented in the list again. However, the intention of this group filtering is that users are prevented from viewing some forbidden certificates, which are likely set for parental guidance or to protect people sensitive to disturbing content.

Next, we login with two users with different genre preferences. A couple logs in, where the man has a strong preference for genre "action" and the woman for genre "romantic".

	Man	**Woman**
age	35	34
gender	male	female
likes genre	action	romantic

Now if they query the system using a set of keywords, results that are returned are sorted based on the average of the individual ratings. This will result in an equal preference for both action and romantic genres, as is liked by both users. Usage of the other group modelling technique: "average without misery" will result here in the removal of both action and romantic genres if one of the users has a dislike for them (prediction below the threshold). We opt that the "average" of "least misery" strategy is a better strategy to follow here.

In presenting the group of users with recommendations, the collaborative filtering component computes for each user its neighborhood of related users and calculates a top ten of recommended items. We then use the average strategy to calculate the predictions of all users for all recommended items. These items are sorted by this prediction value and presented to the users.



french course France country:France country:Spain country:Germany country:Belgium country:Luxembourg country:Switzerla

[0.5 - 1 - 0 - 0.3] Before Sunrise (1995) (id=77466)
[0.5 - 1 - 0 - 0.2] Agnes Browne (1999) (id=49886)
[0.5 - 1 - 0 - 0.2] Colonel Chabert, Le (1994) (id=116665)
[0.5 - 1 - 0 - 0.2] Cyrano de Bergerac (1990) (id=126723)
[0 - 0 - 0 - 0.2] French Course (id=472361)
[2.6 - 1 - 4.2 - 0.2] Longest Day, The (1962) (id=262341)
[2.2 - 1 - 3.4 - 0.2] Man in the Iron Mask, The (1998) (id=272994)
[0.5 - 1 - 0 - 0.2] Misérables, Les (1998) (id=287383)
[0.5 - 1 - 0 - 0.2] Kelly's Heroes (1970) (id=241214)
[0.5 - 1 - 0 - 0.2] Jefferson in Paris (1995) (id=231179)
[0.5 - 1 - 0 - 0.2] Henry V (1989) (id=203796)

Title: Before Sunrise (1995)

Tagline: ... When love can come as a complete surprise
Can the greatest romance of your life last only one night?

Genres: Classical drama (id=275), Romance (id=148)
Synopsis:

A French grad student named Celine (Julie Delpy)

Matches keywords: french course France country:Switzerland

Bluis is a creation of Erik Loef & Tim Dekker

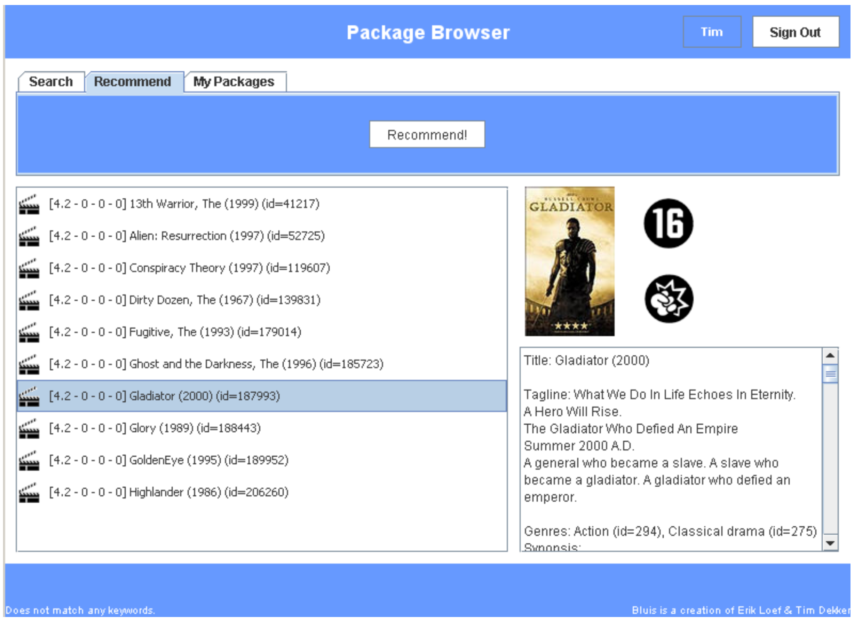
Figure 5.10: Search results for query "French Course" using the improved algorithm

5.3.2 Performance

Furthermore, we have evaluated the time it takes to perform the queries, which needed a personalized result. We have executed a number of queries and timed the execution of the various personalization steps in order to define if it is feasible to use these techniques when the system and, especially, the external services, is being used by a large amount of users. The timings of a search for "French course" for a single user are shown in the table below. Note that the query "French course" is extended by ontologies resulting in query: *"French course France country:France country:Spain country:Germany country:Belgium country:Luxembourg country:Switzerland country:Italy"*. We search the entire database of 500 000 movies for this query. The size of the result set we got was 32085 items. All timings have been performed on a 2.4 Ghz P4 platform.

Process Step	**processing time (ms)**
Query Refinement & Execution	16985
Local Filtering	282
Collaborative Filtering	>15 minutes

We can conclude from these timings that the collaborative filtering algorithm (Pearson R. Algorithm) is clearly the bottleneck here. The speed of this algorithm is a critical factor in the length of the total query execution time. This makes up for a totally unacceptable query processing time, as the user has to wait for this at every keyword search. The time it takes to rate a large set of items is simply too much for our prototype to work smoothly. Note that this time will increase linearly with the number of logged in users. If one wants to implement the collaborative filtering technique, it will require a very fast computer to be able to achieve a fast search result. Therefore, it is likely that this computation is performed by an external service. However, as a lot of users contact this service for every search they perform, it is questionable if creating such a powerful external service is feasible. However, we do recommend that the collaborative filtering technique is used in the system, as it considerably increases the quality of the proposed recommendations and query results. Furthermore, the *query execution* time at the database suffers also from a considerable (and unacceptable) delay (+- 17 seconds). However, we think that this can easily be solved by using a more powerful system.



[4.2 - 0 - 0 - 0] 13th Warrior, The (1999) (id=41217)
[4.2 - 0 - 0 - 0] Alien: Resurrection (1997) (id=52725)
[4.2 - 0 - 0 - 0] Conspiracy Theory (1997) (id=119607)
[4.2 - 0 - 0 - 0] Dirty Dozen, The (1967) (id=139831)
[4.2 - 0 - 0 - 0] Fugitive, The (1993) (id=179014)
[4.2 - 0 - 0 - 0] Ghost and the Darkness, The (1996) (id=185723)
[4.2 - 0 - 0 - 0] Gladiator (2000) (id=187993)
[4.2 - 0 - 0 - 0] Glory (1989) (id=188443)
[4.2 - 0 - 0 - 0] GoldenEye (1995) (id=189952)
[4.2 - 0 - 0 - 0] Highlander (1986) (id=206260)

Title: Gladiator (2000)

Tagline: What We Do In Life Echoes In Eternity.
A Hero Will Rise.
The Gladiator Who Defied An Empire
Summer 2000 A.D.
A general who became a slave. A slave who became a gladiator. A gladiator who defied an emperor.

Genres: Action (id=294), Classical drama (id=275)
Synopsis:

Does not match any keywords.
Bluis is a creation of Erik Loef & Tim Dekker

Figure 5.11: Community-based recommendations

Given the above results, we decided to "cut down" the database size to a more acceptable size. We performed the query again, but this time only on a subset of 3000 movies. The size of the result set we got was 131 items.

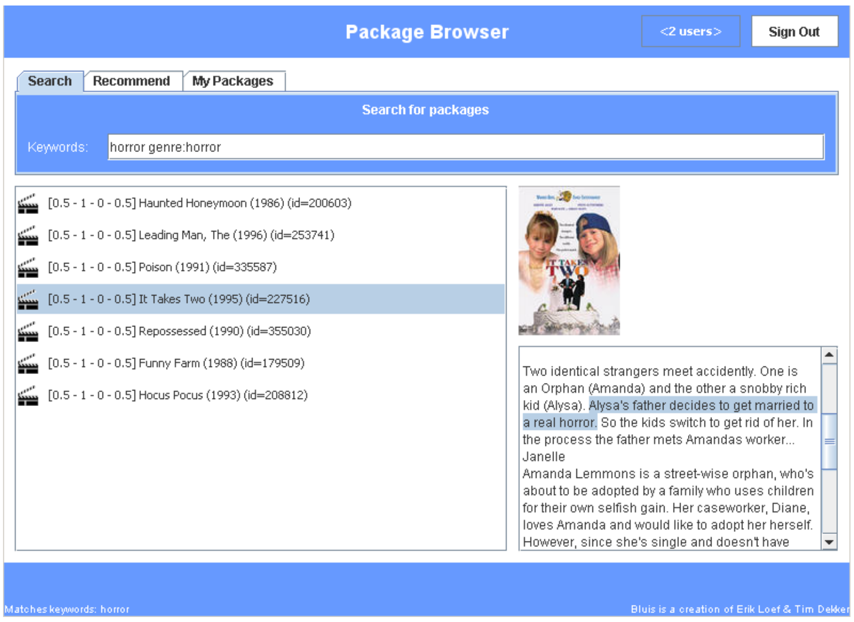
Process Step	**processing time (ms)**
Query Refinement & Execution	438
Local Filtering	109
Collaborative Filtering	2547

This timing is much more acceptable than when searching the entire database. We configured the prototype to work with this database size.

Another important factor in the performance is the initialization time of the ontologies, that are used in the system. Upon startup of a client application, the server loads al ontologies into memory to allow for fast query processing when the user performs a search. We have also timed this process and got to the following results.

Ontology	**initialization time (ms)**
Wordnet	2593
Geography	3672
TV-Anytime	391

We can conclude from these findings that the initialization time for all ontologies allow for a reasonable startup time of the system. Note that we have used our own "cut-down" version of the WordNET ontology, as the full ontology would require too much initialization time and memory usage (>1 GB) for our system to work smoothly.



Keywords: horror genre:horror

[0.5 - 1 - 0 - 0.5] Haunted Honeymoon (1986) (id=200603)
[0.5 - 1 - 0 - 0.5] Leading Man, The (1996) (id=253741)
[0.5 - 1 - 0 - 0.5] Poison (1991) (id=335587)
[0.5 - 1 - 0 - 0.5] It Takes Two (1995) (id=227516)
[0.5 - 1 - 0 - 0.5] Repossessed (1990) (id=355030)
[0.5 - 1 - 0 - 0.5] Funny Farm (1988) (id=179509)
[0.5 - 1 - 0 - 0.5] Hocus Pocus (1993) (id=208812)

Two identical strangers meet accidently. One is an Orphan (Amanda) and the other a snobby rich kid (Alysa). Alysa's father decides to get married to a real horror. So the kids switch to get rid of her. In the process the father mets Amandas worker...
Janelle
Amanda Lemmons is a street-wise orphan, who's about to be adopted by a family who uses children for their own selfish gain. Her caseworker, Diane, loves Amanda and would like to adopt her herself. However, since she's single and doesn't have

Matches keywords: horror
Bluis is a creation of Erik Loef & Tim Dekker

Figure 5.12: Group filtering based on allowed certifications

Chapter 6

Conclusions

Now that we have described everything that we have researched, designed and implemented it is time to give an overview of all important conclusions drawn. There are three different topics, where our conclusions are derived from.

- Conclusions based on the research about the concept of TV-anytime content packaging.
- Conclusions based on research about the personalization approach
- Conclusions based on the implementation of the prototype

We have found out that the TV-Anytime specification provides a very detailed metadata specification, which is, together with the packaging structure, a superior candidate for modelling the content in our system.
Based on the research we performed on the extension of a package 4.2.3 we can conclude that the best way to extend a package is to modify the CRID resolving tree at the CRID Authority. When the CRID resolution is changed, every package containing that CRID is updated. This can also have a negative effect (unwanted packages are updated). In that case the package needs to be extended.
Because it is possible for end-users to make packages, and given that packages need to be extended at the CRID authority (unless unwanted modification are expected). It is necessary that a CRID Authority is always available. This implies that a CRID Authority is not a service in the Blu-IS platform, but an external service.
In the design of our personalization process, we have concluded (based on read papers) that a hybrid personalization approach is the best choice, as this combines the advantages of all applied techniques and will remove some of the shortcomings of the individual techniques, which are described in the personalization chapter. Based on the literature research can be concluded that techniques for stereotype are useful for initialization of user models, and content based filtering together with collaborative filtering covers the weaknesses of each technique.
From the implementation we can conclude that the collaborative filtering predications step in the personalization approach is not suitable for large datasets, because it took 15 minutes to rate a dataset of 30.000 items, which is unacceptable.
Another conclusion from the implementation referring to the initial WordNET ontology, which consists of the complete WordNET thesaurus in OWL. The problem with this ontology was, that the file was too big (75MB) to load in the JAVA-parser. In our design we made the ontologies part of the system, but from the implementation we can now conclude that it would be better to have an external component, where ontologies can be queried. In our prototype we “fixed” the problem by removing all superfluous information from the ontology.
By the evaluation of the prototype the results which were displayed gives rise to change the algorithm, which combines semantic closeness and the personalization algorithms. Because the system was designed to show the items we most like. But after some testing and evaluation, we

want to have the items we search for, which are the items with the most keyword matches, have a higher ranking than items we like (derived from our user model). Based on this conclusion we changed the sorting algorithm in such a way that the system first orders the resultset, based on the obtained semantic closeness. And second the items with similar semantic closeness values are sorted , based on the outcome of the proposed algorithm.

6.1 Suggestions for further work

We consider our system to be a good basis for a personalized home media system, which will work with TV-Anytime packaged content. During our research, we encountered a lot of topics that would require a very thorough research to be able to design the part properly in our system. Unfortunately, we had to decide to leave some parts out of the scope of our graduation assignment as there was simply not enough time available to research everything. These are the following aspects.

- Multiple "box" applications
  The current prototype only works with one application, which has been described in our implementation chapter. It would be nice if the box would allow to make use of multiple applications which can be "plugged in", which would allow them to use the package handling and personalization layer, thus giving these applications access to packaged content which can be queried for in a personalized way. This would require investigation in how multiple applications could work together in parallel and how the system will be able to know which user is using what application using a certain device.

- Presentation generation
  Right now, only one default presentation is used on both TV and PC. The Java implementation allows the user interface to scale to the appropriate screen resolution, but that is where the adaptation of our presentation ends. In the future, we would want to support more device types (handheld, mobile phone), which a user can use to interact with the system. The system could then make use of different presentation templates for these devices. Another issue here is how to personalize the presentation for a specific (group of) user(s).

- Improved personalization techniques
  As the system in the end needs to be used by normal users (without expert knowledge), it is advisable to test the system using a large enough set of different users. The experiences of each user with the personalization process can be used to further improve these personalization techniques.

- Addition of ontologies
  More ontologies could be added to the system, such as a Time ontology and an ontology about famous persons for example, which would further improve the derivation of the context of the entered keywords. Also, the parameters of the query refinement step could be tweaked further to achieve a more powerful semantic closeness algorithm.

- Collaborative filtering performance enhancement
  The collaborative filtering component proved to be a considerable bottleneck in our personalization process. Improving this algorithm or finding out another way to use this algorithm in querying the system to include recommendations based on the community and is definitely a recommendation for further work.

- PVR, Blu-Ray, P2P functionality
  While the system currently works with content which can be played streaming from the internet, we have designed a system that allows for content from a lot more sources. We envision a system which would include content from Blu-Ray disc location, is connected to a P2P network and has PVR functionality to further improve the current system.

- Dealing with multiple metadata services
  In the specification of TV-Anytime it is possible to have multiple metadata services. In our prototype (and architecture) we assume that there is only one metadata service. Considering multiple metadata services, introduces some new issues. When a user publishes a CRID, on a CRID Authority, on which metadata service the user is publishing the metadata.

- Disc-locater specification
  In the Content Services part there is a component for discs. In the context of the Passepartout project this will be either Blu-Ray disc or Hard Discs. By using TV Anytime CRID references we added content references to discs, but this needs to be specified in a formal way, to address certain discs in a unique way, and implemented into the prototype.

- Package constrains
  A Package is an XML representation of a collection of content. The Package doesn't contain the content itself, but has only references to the content (using TV Anytime CRIDs). In the TV-Anytime specification, some constrains are specified, these constrains aren't yet considered in the current prototype.

6.2 Acknowledgements

In this last section of this document, we would like to thank everyone that made it possible for us to do this assignment and made it more pleasant for us to do so. First of all, we would like to thank our supervisors, Lora Aroyo from the TU/e and Rop Pulles from Philips Applied Technologies. Thank you for giving us the opportunity to work on this subject that we found extremely nice to work on. We would like to thank Peter Hulsen for his good introduction into the TV-Anytime Packaging concept and for the and clear explanations and helpful discussions we had throughout the project. We would like to thank Pieter Bellekens and Martin Björkman for their hospitality when we were lost in Amsterdam, and of course for their invaluable contributions to our project and sharing the fun with the name "bluis" pronounce "bluis". We want to thank Rop Pulles for the nice Kebab trips and the dramatically programmed Tetris source code. Thanks to John Pleunis for his help with getting the LIRC interface working and for his constant faith in us. Then we would like to thank all members and students of the Glastonbury department which we had incredibly nice time with: Louis van de Wiele and Michael Konings (aka de knaapjes) for their valuable contributions to our prototype: the CRID authority and the metadata service and for the good times we had during our work on the project. Lin-Feng and Vinhdoha (aka the döners), we had a really fun time with all the nice games : "can I use your thrash-bin", the "barbecue" games, "making the soup", "more meat", "you go down" and of course the immensely popular kebab song. Dominique, good luck with your integration into dutch society and thanks for organizing all the nice parties and contests, you are the true salad king! Thanks go out to everyone else who supported us during the project and made the whole project a nice experience: Eric, Johannes, Michael Kw. We want to thank Olaf van der Spek, aka the number one poster of Tweakers, aka SpekO, for the beautiful t-shirts and for being a nice opponent in Need for Speed and Command&Conquerer. Where he made some catastrophal mistakes, so he was beaten up by us. We want to especially thank Kinky Duiveltje, althought we never met him, we were having a great time talking about him. We also want to thank the Kebab Man of the Seriyaya doner salon for supplying us with the right and the right amount of meat when we needed it.

Of course the authors want to thank each other for the nice cooperation and for being able to share all experiences we had during our assignment and the passepartout meetings and trips. Of course Erik wants to thanks Tim for the nice introduction at GEWIS and teaching LateX. Tim wants to thank Erik for his extremely bad english and for telling nice stories when I did not know what to say anymore.

Eindhoven 2005

References

[1] TV-Anytime Forum:
http://www.tv-anytime.org/

[2] TV-Anytime System Description Specification (v1.1)
ftp://tva:tva@ftp.bbc.co.uk/pub/Specifications/SP002v11.zip

[3] TV-Anytime Content Referencing Specification (v1.2)
ftp://tva:tva@ftp.bbc.co.uk/pub/Specifications/SP004v12.zip

[4] Delivering T-Learning with TV-Anytime through Packaging
P. Hulsen, J.-G. Kim, H.-K. Lee and K.-O. Ka

[5] The TV-Anytime Content Reference Identifier (CRID)
RFC4078

[6] TV-Anytime Phase 2 Metadata Specification
ftp://tva:tva@ftp.bbc.co.uk/pub/Specifications/SP003-3v20.zip

[7] TV-Anytime Metadata Services over a Bi-directional Network
SP006v10

[8] OWL - Ontology Web Language
http://www.w3.org/2004/OWL/

[9] XML - eXtensible Markup Language
http://www.w3.org/XML/

[10] XML Schema
http://www.w3.org/XML/Schema

[11] RDF - Resource Description Framework
http://www.w3.org/RDF/

[12] RDF Schema
http://www.w3.org/RDF/Schema

[13] XML TV
http://membled.com/work/apps/xmltv/

[14] The Moving Pictures Expert Group
http://www.chiariglione.org/mpeg/

[15] the MPEG-7 standard
http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm

[16] the MPEG-21 standard
http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm

[17] DRM Watch: Analysis of DRM Technology
http://www.drmwatch.com/

[18] ITEA Passepartout Full project proposal

[19] MPEG-21 Digital Item Declaration
FDIS, ISO/IEC 2002-2 (N4813), May 2002, Fairvax, VA, USA.

[20] RFC1034: DOMAIN NAMES - CONCEPTS AND FACILITIES

[21] RFC1123: Requirements for Internet Hosts – Application and Support

[22] RFC3986: Uniform Resource Identifier (URI): Generic Syntax

[23] The Hera Project
http : //wwwis.win.tue.nl/ hera

[24] J. F. Allen, "Maintaining Knowledge about Temporal Intervals", Communications of the ACM, Vol. 26, Nov. 1983, pp. 832-843.

[25] Java RMI *http://java.sun.com/products/jdk/rmi/index.jsp*

[26] Sun Microsystems website
http://www.sun.com

[27] jLirc website
http://jlirc.sourceforge.net/

[28] JDIC website
https://jdic.dev.java.net/

[29] Windows Media Player 10
http://www.microsoft.com/windows/windowsmedia/mp10/default.aspx

[30] Video LanClient Media Player
http://www.videolan.org/vlc/

[31] Linux Infrared Remote Control
www.lirc.org

[32] Apache Tomcat Webserver
tomcat.apache.org

[33] MySQL
www.mysql.com

[34] Simple Object Access Protocol
ws.apache.org/soap

[35] Jena
http://jena.sourceforge.net/

[36] Kijkwijzer
http://www.kijkwijzer.nl

[37] Motivaction
http://www.motivaction.nl

[38] The Blu-Ray Disc Association
http://www.blu-raydisc.com/

[39] The HD-DVD Association
http://www.hd-dvd.org

[40] KISS Networked Entertainment
http://www.kiss-technology.com/

[41] The Windows XP Media Center
http://www.microsoft.com/windowsxp/mediacenter/default.mspx

[42] The Jules Verne Project
http://www.hitech-projects.com/euprojects/Jules_Verne/

[43] Personalised Home Media Centre Using Semantically Enriched TV-Anytime Content
Martin Bjrkman, Lora Aroyo, Pieter Bellekens, Tim Dekker, Erik Loef and Rop Pulles

[44] The CoFe Collaborative Filtering Engine
http://eecs.oregonstate.edu/iis/CoFE/

[45] On the Role of a Users Knowledge Gap in an Information Retrieval Process
Nenad Stojanovic
Institute AIFB, University of Karlsruhe, Germany

[46] The WordNet ontology
http://wordnet.princeton.edu/

[47] The Suggested Upper Merged Ontology
http://ontology.teknowledge.com/

[48] OWL-Time
http://www.isi.edu/ pan/OWL-Time.html

[49] User modeling in dialog systems: Potentials and hazards.
Alfred Kobsa.
Artificial Intelligence and Society, 1:214.240, 1990.

[50] User Modeling and Recommendation Techniques for Personalized Electronic Program Guides

[51] Ontology-based Interactive User Modeling for Adaptive Web Information Systems
R.O. Denaux

[52] Improving the Quality of the Personalized Electronic Program Guide
Derry OSullivan, Barry Smyth, David C. Wilson, Kieran McDonald and Alan Smeaton

[53] Content Based Filtering
http://www.kom.e-technik.tu-darmstadt.de/acmmm99/ep/kohrs/

[54] F. Heylighen
http://pespmc1.vub.ac.be/COLLFILT.html

[55] Automated Collaborative Filtering and Semantic Transports
Alexander Chislenko - 1997
http://www.lucifer.com/ sasha/articles/ACF.html

[56] TiVo: Making Show Recommendations Using a Distributed Collaborative Filtering Architecture
Kamal Ali, Wijnand van Stam Liliana Ardissono, Cristina Gena, Pietro Torasso

[57] Personalized Electronic Program Guides for Digital TV
AI Magazine, Summer, 2001 by Barry Smyth, Paul Cotter
*http : //www.findarticles.com/p/articles/mi_m*2483/ *is_*2_22/*ai_*76698531

[58] A Framework for Collaborative, Content-Based and Demographic Filtering
Michael J. Pazzani
Department of Information and Computer Science University of California, Irvine Irvine, CA 92697
http://www.ics.uci.edu/ pazzani/Publications/AIREVIEW.pdf

[59] Collaborative Filtering Research Papers
James Thornton
http://jamesthornton.com/cf/

[60] GroupLens: An Open Architecture for Collaborative Filtering of Netnews
Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, John Riedl
http://www.si.umich.edu/ presnick/papers/cscw94/GroupLens.htm

[61] Combining Collaborative and Content-Based Filtering Using Conceptual Graphs
Patrick Paulson and Aimilia Tzanavari

[62] Enhancing Collaborative Filtering with Demographic Data: The case of Item-based Filtering
Manolis Vozalis and Konstantinos G. Margaritis
Parallel Distributed Processing Laboratory Department of Applied Informatics, University of Macedonia

[63] Bootstrapping and Decentralizing Recommender systems
Tomas Olsson

[64] Item-based Collaborative Filtering Recommendation Algorithms
Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl
*http : //www−users.cs.umn.edu/ karypis/publications/Papers/PDF/www*10*_sarwar.pdf*

[65] On the Construction of TV Viewer Stereotypes Starting from Lifestyles Surveys

[66] A Context-Aware Decision Engine for Content Adaptation
Wai Yip Lum, Francis C.M. Lau

[67] Towards a Better Understanding of Context and Context-Awareness
Anind K. Dey and Gregory D.
Abowd Graphics, Visualization and Usability Center and College of Computing, Georgia Institute of Technology, Atlanta, GA, USA 30332-0280

[68] A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications
Anind K. Dey and Gregory D. Abowd
College of Computing & GVU Center, Georgia Institute of Technology Daniel Salber, IBM T.J. Watson Research Center

[69] Pursuing Engaged Interaction
Exploring the potential of artistic research in the creation of new interactive human computer interfaces
Angela Plohman, V2_Lab

[70] How and Why People Watch TV: Implications for the Future of Interactive Television.
Barbara Lee and Robert S. Lee (1995)
Journal of Advertising Research, vol.35, no.6,

[71] FIT-recommending TV programs to family members
Computers & Graphics 28 (2004) 149-156
Dina Goren-Bar, Oded Glinansky

[72] Group Modeling: Selecting a Sequence of Television Items to Suit a Group of Viewers
User Modeling and User-Adapted Interaction 14: 37-85, 2004.
2004 Kluwer Academic Publishers. Judith Masthoff

[73] PolyLens: A Recommender System for Groups of Users
O Conner et al., 2001
http://www.grouplens.org/papers/pdf/poly-camera-final.pdf

[74] MusicFX: An arbiter of group preferences for computer supported collaborative workouts.
McCarthy, J. and Anagnost, T. (1998)
ACM1998 Conference on CSCW, Seattle, WA, pp. 363-372

[75] Intrigue: Personalized recommendations of tourist attractions for desktop and handset devices.
Ardisonno et al.

[76] UML, Unified Modelling Language
http://www.uml.org

[77] Stoneroos, Interactive television
http://www.stoneroos.nl/

[78] Royal Philips Electronics, The Neterlands
http://www.philips.com

[79] URI, Unified Resource Identifier
http://www.w3.org/2001/12/URI/

[80] Interest-based Recommendation in Digital Library
http://www.ansinet.org/fulltext/jcs/jcs1140-46.pdf

[81] Exploration/Explootation in adaptive recommender systems
http://hcs.science.uva.nl/dumpers/Eunite.pdf

[82] Microsoft Media Center *http://www.microsoft.com/windowsxp/mediacenter/default.mspx*

[83] National research institute for mathematics and computer science.
http://www.cwi.nl/

Appendix A

French Course Package

```
<?xml version="1.0" encoding="UTF-8" ?>
<TVAMain
xmlns="urn:tva:metadata:2005-03"
xmlns:tva2="urn:tva:metadata:Phase2:2005-03"
xmlns:mpeg21="urn:mpeg:mpeg21:2003:01-DIA-NS"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:tva2:2005-03"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:tva:metadata:Phase2:2005-03
tva2_metadata.xsd" xsi:type="tva2:ExtendedTVAMainType">
  <tva2:PackageTable>
    <tva2:Package crid="CRID://teleac.nl/Package/12-1-2005">
      <tva2:Item>
        <tva2:Component>
          <tva2:Resource crid="CRID://frenchcourse.com/chapter1/taxi" />
        </tva2:Component>
        <tva2:Component>
          <tva2:Resource crid="CRID://frenchcourse.com/chapter1/chat" />
        </tva2:Component>
        <tva2:Component>
          <tva2:Resource crid="CRID://frenchcourse.com/chapter1/food" />
        </tva2:Component>
      </tva2:Item>
      <tva2:Item>
        <tva2:Component>
          <tva2:Resource crid="CRID://frenchcourse.com/chapter2/holiday"/>
        </tva2:Component>
        <tva2:Component>
          <tva2:Resource crid="CRID://frenchcourse.com/chapter2/timetable"/>
        </tva2:Component>
        <tva2:Component>
          <tva2:Resource crid="CRID://frenchcourse.com/chapter2/emergency"/>
        </tva2:Component>
      </tva2:Item>
    </tva2:Package>
  </tva2:PackageTable>
</TVAMain>
```

Appendix B

CRID Resolution Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<ContentReferencingTable
xmlns="http://www.tv-anytime.org/2001/04/ContentReferencing"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:schemaLocation="http://www.tv-anytime.org/2001/04/ContentReferencing
local_copy_of_schema.xsd" version="1.0">
<Result CRID="crid://frenchcourse.com/chapter1/shortbreak"
  status="resolved"complete="true" acquire="all">
  <CRIDResult>
    <Crid>crid://frenchcourse.com/chapter1/taxi/</Crid>
    <Crid>crid://frenchcourse.com/chapter1/askfordirection/</Crid>
    <Crid>crid://frenchcourse.com/chapter1/taxi/gettingasnack</Crid>
  </CRIDResult>
</Result>
<Result CRID="crid://frenchcourse.com/chapter1/taxi"
  status="resolved" complete="true" acquire="all">
  <CRIDResult>
    <Crid>crid://frenchcourse.com/chapter1/taxi/</Crid>
  </CRIDResult>
  <LocationsResult>
    <Locator>dvb://1.4ee2.3f4;4f5@2001-03-27T18:00:00.00?:00</Locator>
  </LocationsResult>
</Result>
<Result CRID="crid://frenchcourse.com/chapter1/askfordirection"
  status="resolved" complete="true" acquire="all">
  <LocationsResult>
    <Locator>
      http://frenchcourse.com/resources/chapter1/askfordirection/speech.mp3
    </Locator>
  </LocationsResult>
</Result>
<Result CRID="crid://frenchcourse.com/chapter1/gettingasnack"
  status="resolved" complete="true" acquire="all">
  <LocationsResult>
    <Locator>
      http://frenchcourse.com/resources/chapter1/askfordirection/speech.mp3
    </Locator>
    <Locator>
      dvb://1.4ee2.3f4;4f5@2001-03-27T18:00:00.00?:00
    </Locator>
  </LocationsResult>
</Result>
</ContentReferencingTable>
```

Appendix C

Interface Descriptions

This appendix lists all interfaces for all components of the Blu-IS system architecture.

C.1 Used Types

Action	Type representing a certain user action
BluisResult	Type representing a list of CRIDs with a prediction value for them.
Content	Type representing a certain content element (file)
ContentMetadata	Type representing the metadata of a content element
CRID	Type representing a String starting with “crid://”
CRIDResult	Type representing the result of a CRID resolution step
DeviceID	int
Locator	Type representing a locator
MDType	Type representing a certain metadata type (review, descriptive)
Metadata	Type representing metadata
Package	Type representing a package
PackageMetadata	Type representing the metadata of a certain package
Prediction	Type representing a prediction value for a certain item
RefinedQuery	Type representing a refined query, meaning a list of keywords with their semantic closeness value, possibly with denoted context.
SessionID	Type representing a session identification number
User	object representing a user with all of his properties
UserID	int
XMLCRIDResult	String containing the result of a CRID resolution step in XMLFormat, which must be conform the TV-Anytime CRIDResult specification
XMLPackage	String containing the package in XML format, which must be conform the TV-Anytime package specification

C.2 Interfaces

C.2.1 Package Handling

- $deletePackage(crid : CRID)$

  Pre: $cridistheCRIDofapublishedpackage.$
  Post: *the package has been deleted*

- $getLocalPackages() : listofCRID$

  Pre: *true*
  Return: *returns a list of CRIDs for all locally stored packages*

- $getPublishedMetadata() : listofMetadata$

  Pre: *true*
  Return: *a list of all published metadata elements*

- $getPublishedPackages() : listofCRID$

  Pre: *true*
  Return: *a list of all published packages*

- $publishMetadata(m : Metadata)$

  Pre: *true*
  Post: *the metadata has been published*

- $publishPackage(p : Package)$

  Pre: *true*
  Post: *the package has been published*

- $recordContent(c : CRID)$

  Pre: *true*
  Post: *the selected crid has been scheduled for recording.*

- $resolveCrid(c : CRID) : CRIDResult$

  Pre: *true*
  Return: *the resolved crid result*

- $retrieveMetadata(c : CRID) : Metadata$

  Pre: *true*
  Return: *metadata for the specified crid*

- $retrievePackage(c : CRID) : Package$

  Pre: *true*
  Return: *the requested package*

- $storePackage(p : Package)$

  Pre: *true*
  Post: *the package has been stored locally.*

- $unpublishMetadata(c : CRID)$

  Pre: $cisaCRIDforapieceofpublishedmetadata$
  Post: *the metadata has been unpublished*

- $publishPackage(p : Package)$

  Pre: $cisaCRIDforapublishedpackage$
  Post: *the package has been unpublished*

C.2.2 Personalization

PersonalizationClient

- $performQuery(users : listofUserID, keywords : listofString) : BluisResult$

  Pre: $users! = null \wedge keywords! = null$
  Return: *a list of metadata elements matching the specified keywords, together with the prediction value targeted at the group of users.*

  Performs a search for content items matching some set of keywords (Query 1 section 4.4.1). The *package handling* component is contacted to perform the search for packages and content. Before this search is executed, the *QueryRefiner* uses ontologies to give an extended set of keywords. After a set of results is returned from the *package handling* component, the resultlist is sorted and filtered with the use of the *UserModelHandler*, *LFClient*, *CFClient* and *GroupStrategies* classes. The filtered list (CRIDs + prediction value) is returned.

- $recommendContent(users : listofUserID) : BluisResult$

  Pre: $users! = null$
  Return: *a list of metadata elements with the top ten recommended items for each user, together with the prediction value targeted at the group of users.*

Recommends the top ten recommended items for each user (Query 2 section 4.4.1). The *CFClient* is contacted to perform this recommendation strategy. The *UserModelHandler* and *GroupStrategies* classes are then used to calculate group preferences. The filtered list (CRIDs + prediction value) is returned.

- $filterPackage(users : listofUserID, package : CRID) : Package$

  Pre: $users! = null$
  Return: *a package object personalized for the group of users.*

  Returns a personalized package for the user(s) (Query 3 section 1.4.1). The items of the package are sorted and filtered so that the user is presented with the most recommended items of the package.

- $registerNewUser(newuser : User) : userID$

  Pre: $newUser! = null$
  Return: *the ID of the newly created user or -1 if an error occurred during the registration process.*

  Registers a new user of the Blu-IS box at the *User Model service.* The User Model is initialized with stereotypical filtering.

- $startSession(uids : listofUserID)$

  Pre: $uids! = null\wedge$ *each UserID has been registered at the Blu-IS server*
  Post: *a session has been created on the Blu-IS server for the specified group of users.*

  Starts a session for a certain group of users. A session starts at the moment a user logs in to the system and ends when he logs out.

- $registerUserAction(uids : listofUserID, action : Action)$

  Pre: $uids! = null$
  Post: *logs the specified action for a certain group of users.*

  Registers an action that the user does. The actions that a user performs are not directly used to update the user model, but are stored locally as a sort of short-term memory of the user model.

- $updateUM()$

  Pre: $newUser! = null$
  Post: *all actions have been processed to update the corresponding user models.*

  Updates the user model for a certain user with all actions that occurred during the user's session.

- $getBoxUsers() : listofUser$

  Pre: *true*
  Return: *All registered users at the Blu-IS server.*

Retrieves all registered users at the Blu-IS server from the *user model handler*.

- $getUser(uid : UserID) : User$

  Pre: *uid has been registered at the Blu-IS server*
  Return: *The user with userId uid.*

UserModelHandler

This component contains a subset of the methods of *PersonalizationClient* class, which are the following:

- getBoxUsers
- getUser
- updateUM
- registerUserAction

The methods implemented in PersonalizationClient are simply forwarded to this class, as this class is responsible for *User Model Service* communication. The methods in this class perform the actual operations for the methods described in the *PersonalizationClient* Class.

CFClient

- $getPredictionsForItems(items : listofCRID, uids : listofUserID) : listofPrediction$

  Pre: $items! = null \wedge uids! = null$
  Return: *A list of prediction values for the list of items.*

- $getRecommendations(uid : UserID, n : int) : listofPrediction$

  Pre: $n > 0$
  Return: *A list of prediction values for the top n recommended items*

- $setRating(uid : UserID, rating : int)$

  Pre: $0 < rating < 6$
  Post: *rating has been registered for the user with userid uid*

LFClient

- $getPredictionsForItems(items : listofCRID, uids : listofUserID) : listofPrediction$

  Pre: $items! = null \wedge uids! = null$
  Return: *A list of prediction values for the list of items.*

- $setRating(uid : UserID, rating : int)$

  Pre: $0 < rating < 6$
  Post: *rating has been registered for the user with userid uid*

QueryRefiner

- $refineQuery(keywords : listofString) : RefinedQuery$

  Pre: *true*
  Return: *The refined query*

MetadataHandler

- $performQuery(keywords : listofString) : listofCRID$

  Pre: *true*
  Return: *a list of content items matching the specified query.*

GroupStrategies

- $calcGroupPreference(ratings : float[], strategy : int) : float$

  Pre: *true*
  Return: *The group rating calculated from the list of individual ratings.*

  The specified parameter *strategy* is an integer denoting which strategy to use:

  1. Average
  2. Average without misery
  3. Least misery

C.2.3 Application

The application class contains a number of interfaces which are triggered as events by the *UI Dispatcher* and which are forwarded to the responsible layer: *packaging* or *personalization.* The *Application* class other responsibility is to track the active sessions. The list of interfaces is basically the combined interfaces list of both the components *packaging* and *personalization.*

C.2.4 Presentation

- $generatePresentation(p : PresentationObject, u : UserID, d : DeviceID)$

  Generates a presentation for a certain User using a certain Device using some presentation template. The U*ser Model service* is contacted to be able to create a personalized presentation.

C.2.5 UI Dispatcher

Interfaces of the UI Dispatcher are the same as those of the Application class. Every method is received from the client application in the form of a HTTPRequest. The requested method and parameters are extracted from this request and the parsed request is forwarded to the *Application* class as an event. A HTTPResponse is sent back to the client after the event returns.

C.2.6 Client Application

The *client application* has no interfaces, as the User Interface of this *client application* is used to perform the different methods and functions the component consists of.

Appendix D

Use Cases: State-Transition Diagrams

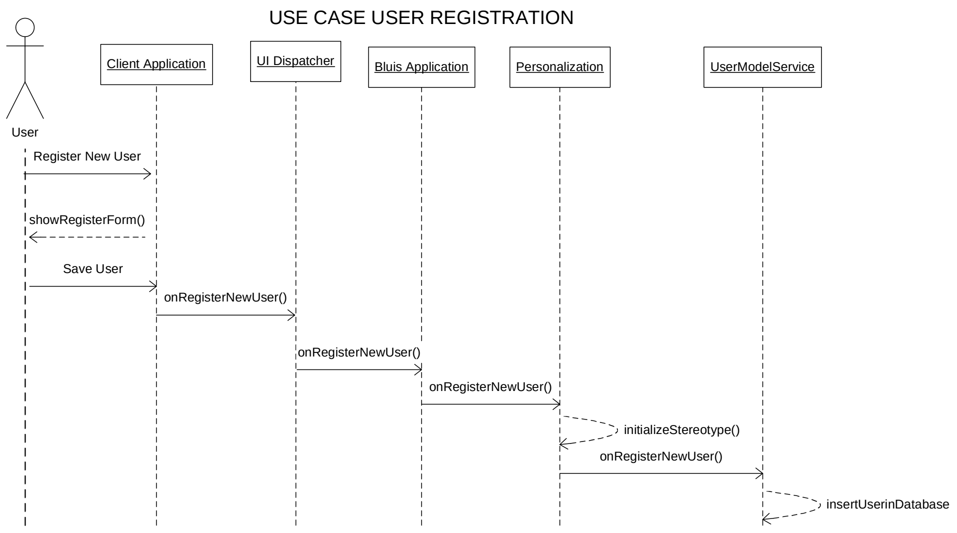


Figure D.1: Use Case: Registration of a new user

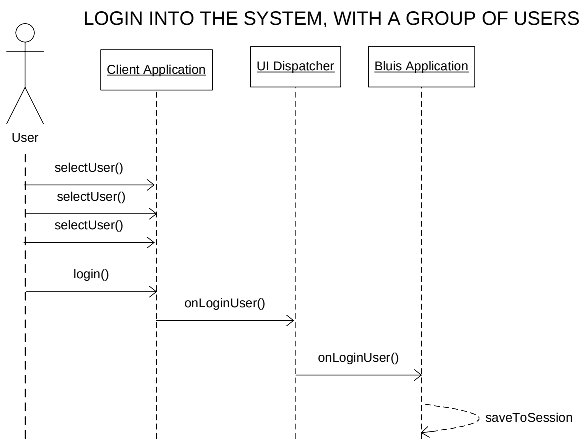


Figure D.2: Use Case: Login as a group of users

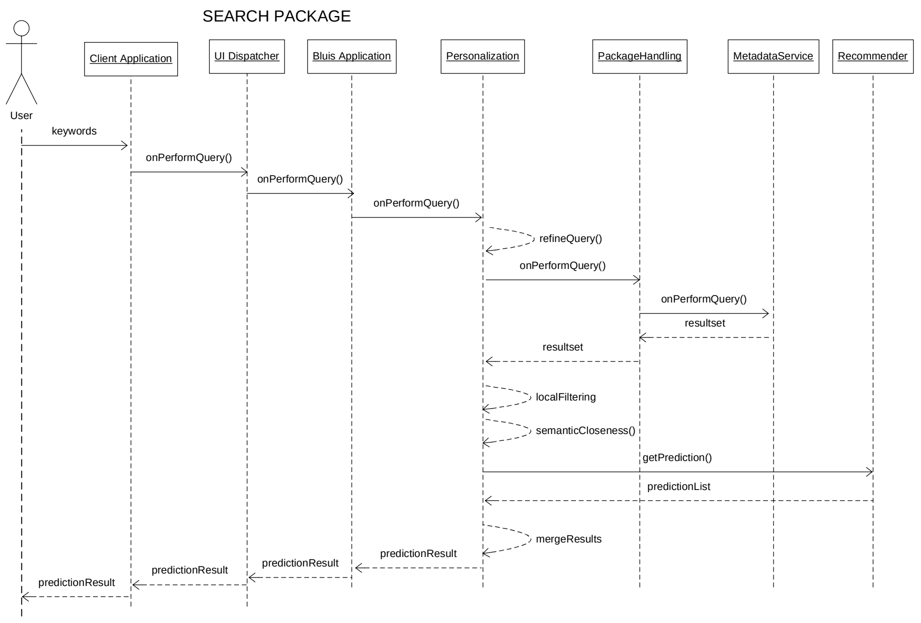


Figure D.3: Use Case: Search for a package

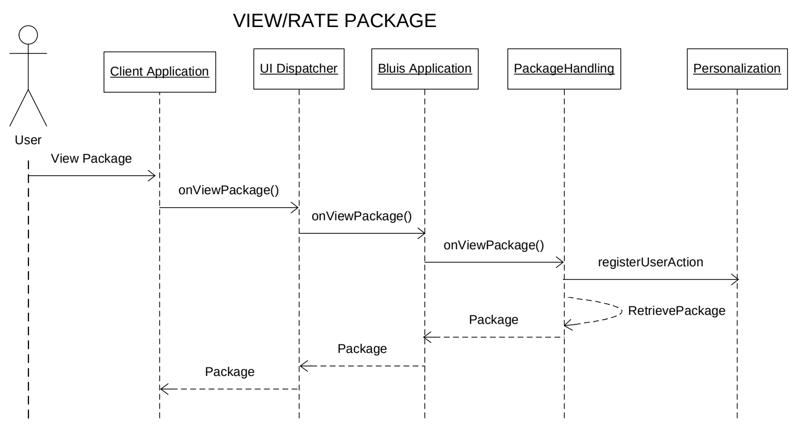


Figure D.4: Use Case: View a package

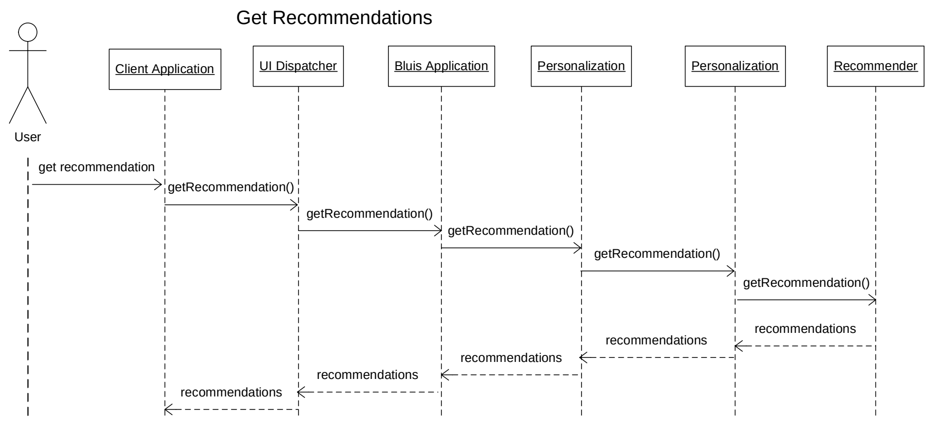


Figure D.5: Use Case: Ask for a recommendation

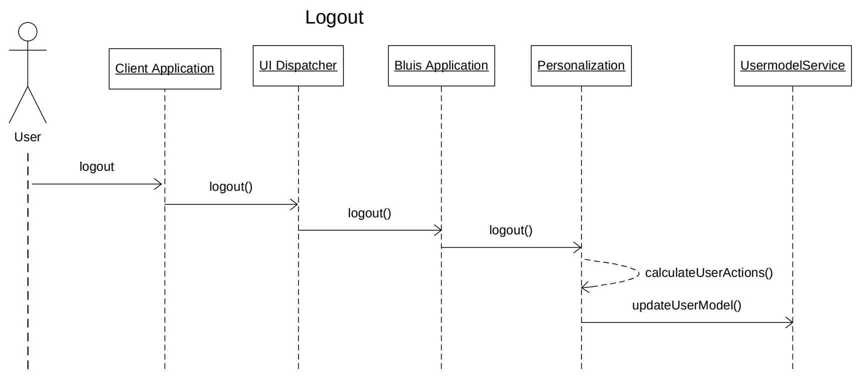


Figure D.6: Use Case: Logout

Appendix E

Published Paper

Personalised Home Media Centre Using Semantically Enriched TV-Anytime Content

Martin Björkman[1], Lora Aroyo[1], Pieter Bellekens[1], Tim Dekker[1], Erik Loef[1] and Rop Pulles[2]

[1]Eindhoven University of Technology, Faculty of Mathematics and Computer Science,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{m.bjorkman, l.m.aroyo, p.a.e.bellekens}@tue.nl, {t.j.dekker, e.loef}@student.tue.nl

[2]Philips Applied Technologies, P.O. Box 80002, 5600 JB Eindhoven, The Netherlands
rop.pulles@philips.com

Abstract
The paper presents a design and middleware implementation of a personalised home media centre. The proposed system aims to constitute the fundamentals to go beyond existing media centres and electronic programme guides, in the sense of offering a personalised experience for a single and multiple users in a connected home environment. User's characteristics, preferences and context are used to personalise the user's experience of viewing and interacting with multimedia content on different heterogeneous devices. The TV-Anytime specification is used as the underlying content and metadata format for handling content from IP, digital broadcast and Blu-ray disc sources. We show how semantics can enrich the TV-Anytime content classification in order to achieve intelligent personalised content search and recommendations.

1. Introduction

The information society is going digital to an ever greater extent in the field of media, e.g. in television, radio, music and news. These changes bring new possibilities and challenges which affect the whole media chain; content production, distribution and not least the end-user (the consumer). We describe in this paper our research focussing on the experience of the home user and the possibilities for connecting several digital media input channels and user devices into a connected media centre.

As an effect of digitalising, new forms of home media are emerging as digital systems are converging. Different content, e.g. from TV, radio, music, homemade images and videos, are no longer bound separate devices or to local storage, and the development of the Internet makes the media boundaries become less limited. As envisioned by for instance IBM (Berman, 2004) the future media may become more pervasive and offer a more ubiquitous and immersive experience, as increasing technological sophistication brings new media environments. The transfer to digital content along with technologies and standards like DVB[1], HDTV, voice over IP, Blu-ray[2] and TV-Anytime[3] create opportunities to bring new interactivity to the traditional TV concept and change it drastically. The television itself has not yet experienced any major revolution for the past fifty years, which constitutes a strong contrast to the Internet which has quickly evolved from mere textual information to multimedia content. Adding new technology to the TV concept may provide changes from a traditional one-way to a two-way communication where the user changes from a passive viewer to a more active participant, and programmes structures change from fixed to dynamic. Furthermore, the home users would, to a greater extent, also become content producers (Berman, 2004), thus breaking the traditional business model where companies and institutions are the sole content providers.

In this paper we try to identify requirements, opportunities and problems in home media centres, and we propose an approach to address them by describing an intelligent home media environment. The major issues investigated are coping with the *information overflow* and *need for personalisation* by adapting to the user with respect to age, interests, language abilities, current context etc. The research presented in this paper is a collaboration between Eindhoven University of Technology and Philips Applied Technologies. The work has been carried out within the ITEA funded Passepartout European project, which also includes the partners like Thomson, INRIA and ETRI.

In section 2 we describe the motivation and problem in detail, followed by an illustrative scenario in section 3. The TV-Anytime packaging concept is described in section 4 and serves as the

[1] http://www.dvb.org
[2] http://www.blu-ray.com
[3] http://www.tv-anytime.org

background for understanding our proposed system architecture described in section 5. This latter section elaborates on an interoperable design and semantic techniques for enabling intelligent context-aware personalisation. Section 6 describes the implementation, followed by related and future work in sections 7 and 8.

2. Motivation and Problem Statement

The main problems we investigate are how to design a home media system that can be accessed through various devices in the home and be connected to different media sources. It should provide access to a wide range of content, yet avoid an overflow of information for the user.

A problem for home media centres is how to go beyond the traditional limited solution of a single TV screen and simple remote control. In our demonstrator called Blu-ray Interactive System (Blu-IS), we aim to enable the connection of various devices for interaction, such as shared (large) screens, personal (small) handheld devices and new hand gesture recognition and biosensor-based interfaces. Our anticipation is to create the foundation for an ambient connected home environment.

Regarding an *information overflow* aspect, we assume that the amount of available content will increase enormously with the digital development and bring an explosion of digital content (Murugesan and Deshpande, 2001). Simple electronic programme guides are thus likely to turn inefficient in terms of helping the user in choosing from an overwhelming amount of content (Chorianopoulos, 2004; O' Sullivan et al, 2004). This creates a challenge for media systems to support the user in finding the most relevant and interesting programs. Applying intelligent filtering of the content will in our perspective be crucial to avoid such an overflow.

Various research already shows that there is a need for *personalisation* in dealing with such vast amount of TV content (Ardissono et al, 2004). We believe that a personalization approach in home media centres is also needed in order to capture the user's preferences and use this as basis for the interaction, both regarding content and devices. Since users differ in ages, interests, abilities and language preferences, it is important that this can be represented in the system. For instance, an eight year old person will have very different favourite programs than an adult, and some user might want the movies to always be displayed on the biggest screen, but private content to be shown only on his or her handheld device. By creating a *user model* (Kobsa, 1990) for each user of the system, such personal preferences may be stored. The user model needs to both capture a *user profile*, with the user's preferences, and a *user context*, which describes the current situation that the user is in, e.g. if the user is alone or with a group, available devices at the moment, the time and location etc. The user models furthermore constitute one of the necessary requirements for enabling intelligent filtering of content (the other being content models) to make content *recommendations* (Setten, 2005). By this we mean finding and suggesting content that should be interesting for the user, while filtering out unwanted or uninteresting content.

Apart from supplying semantic models of the user, it is also necessary to have well described content. *Metadata* describing the content needs to follow a standard and be suitable for parsing and searching. Since the amount of content will be huge, this needs to be done with high performance. The metadata should also be the basis for content classification, i.e. sorting the content into different types like fiction, non-fiction, news, sports etc. Further requirements on content descriptions are on *flexibility*, to enable heterogeneous content sources, on *readability*, to enable users to add their own descriptions of homemade content, and *interoperability* so that different applications and environments can process the same metadata. Intelligent search and filtering of content moreover benefit from metadata descriptions that are suitable for reasoning, to deduce new information and enrich content search.

3. Application Scenario

This section describes a scenario to illustrate the goal functionality of our demonstrator. The setting is a family home somewhere in the EU, in the year 2010. The family is assumed to be financially well off, living in a region outside their original parental background. While they wish for the children to integrate with the local community and live and learn from their neighbours, they also value their heritage; linguistic, cultural and religious, to effectively communicate with distant relatives and friends. The family consists of the mother, father, a child about four years old, a deaf nine year old and a teenager. The parents are determined that the children should be effectively multi-lingual and multi-cultural, and will invest time to adapt the multimedia content in the home. They therefore act as media guides and to some extent teachers for their children, by selecting and adapting the content. Since the parents have immigrated to the region, they will have a different preference of content than the default local selection and they use their home media centre to include also programmes from their original home area, e.g. for news, music and movies. They may also choose to alter the language or subtitles of the content.

As the family gathers for a movie night together, the home media centre has suggested a movie that suits each family member's preference and interest. The mother has briefly scanned the story of the movie and discovered that the ending is in her opinion not suitable for the children. She therefore changes to an alternative ending. As they start the movie, they all together use a shared big screen. However, this evening the deaf child also includes additional sign language on his own small screen, although sometimes they use subtitles on the shared screen. The teenager needs to practice her second language and has therefore been asked by her parents to listen to an alternative language version with her headphones. Although they enjoy the movie together, the father also wants to follow a live soccer game broadcast, and therefore uses his own handheld screen to view this private video stream. The media devices in the home are all connected to the ambient connected home media environment.

4. TV-Anytime Packaging

A content structure which goes beyond a fixed linear time structure and allows multiple languages, alternative versions etc. put high demands on the content model. As mentioned, it needs to have a dynamic structure, rich metadata and suitable for various media. We believe that the TV-Anytime standard may serve as the basis in such requirements and we have built our demonstrator upon the TV-Anytime concepts. The standard focuses mainly on broadcast, though we apply it here for content from IP and local disc sources. It organises content into an interconnected structure, where each piece of content is referred to by a Content Reference Identifier, CRID, which is a RFC standard (Earnshaw et al, 2005). This could be used for several purposes; to define locators, which give the actual location where the content is stored, to refer to the content's metadata, or for referring to some other CRID. The TV-Anytime Package is a collection of related CRIDs. The data model of a package adopts the multi-level structure of the MPEG-21 Digital Item Declaration Language (Bormans and Hill, 2002), i.e. *container-item-component* structure, with some extensions.

For example, a language course structured as a package could be organised and divided into chapters and sections where each are identified by a CRID. Figure 1 shows a part consisting of sections with video and audio fragments. Each content element is not stored in the package itself, but is referenced using the CRID. Thus some part can be distributed on a disc and other via IP. The main content of the language course could for instance be on a disc that the user has bought, extra interactive content and trailer for the next course may be distributed via IP. This packaging structure is very dynamic since parts can easily be modified or extended, e.g. the course could be extended with a new chapter by simply adding a CRID reference. Since packages are complex collections of CRIDs, they need to be resolved to discover which items are contained, as well as to get the locator(s) when about to view the actual content. This resolving process is performed by a *CRID Authority*. The response of such resolving request is an XML document containing a list of all CRIDs and locators that it resolves into.

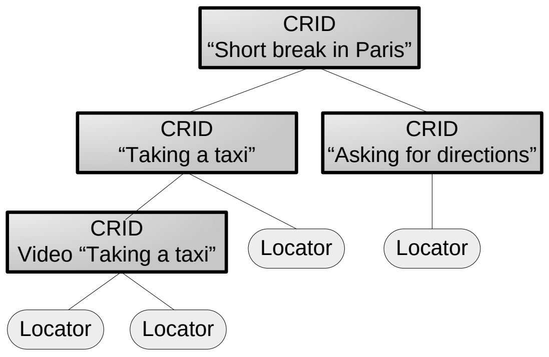


Figure 1. Example CRID representation.

The standard's metadata specification defines how content is described and classified – a fundamental feature for search and filtering. The specification uses XML syntax for capturing different concepts. Apart from technical descriptions such as screen aspect ratio and number of audio channels there are possibilities of describing genre, synopsis, topic keywords and language etc. The genre description is a fine graded taxonomy structure, going from general concepts of fiction / non-fiction, down to specific categories in the very leaf nodes. These are typical well known genres like comedy, drama, daily news, weather forecast etc. used for content classification to sort programmes into categories. The TV-Anytime also describes the basic functions of how to access it via a *Metadata Service.* Our demonstrator uses these features and further maps additional semantics to the metadata in order to improve search for and within packages.

5. Personalised Home Media Centre

5.1 Design of Personalised Home Media Centre

We propose in this section the architecture of the Blu-IS system and connected components which can be seen in Figure 2. We first begin with describing the services and end-user environment before going into details of the central point of connection and control of TV-Anytime content.

Content Services represents in our design the different content distribution channels, where each has its own specific properties. Content can be retrieved from an optical disc. Initially, this will be the system's primary input channel of High Definition content due to its high storage capacity, and we have chosen the Blu-ray disc as technology for this task. Furthermore, the IP channel can offer any type of content and has naturally the advantage of two-way communication, which makes the IP well suited for interactive applications and distributing home made content to others. The Blu-IS demonstrator supports the most commonly used IP protocols of HTTP, FTP, streaming media and peer-to-peer technology. The hard disk drive (HDD) is used for local storage of content.

IP Services are external services that provide content metadata and descriptions of where content is located. This comprise the CRID Authority and the Metadata Service (see Section 4). However, in our system we have extended the Metadata Service to also handle more elaborate requests. The basic functionality to retrieve metadata for a CRID is useful when just the metadata, and not the whole package, needs to be retrieved, e.g. when browsing content, or to show some parts like title and synopsis to the user. However, our extension also enables searching for metadata which contains a set of keywords. This means searching for e.g. "French course" in all CRIDs and retrieving a list of all which have metadata that contain these keywords. This significantly improves the search functionality.

System Services refer to the components that the system uses to handle the available content in an intelligent way. The *User Model Service* (UMS) is responsible for storing and updating information about the users, and the *Recommender Service* (RS) for using a set of filters to recommend content based on the user information. In other words, these are the main components to enable a personalised user experience. The user information can be accessed by sending queries to the UMS, to retrieve e.g. a user's age, how a particular movie was ranked or on a higher abstraction level an estimation of the user's interest in a topic or genre like "sports". Such information is used by the RS to filter large amount of information down to smaller result sets based on the user's interest perspective. Our content model, user model and personalisation process involved in this are described in 5.2 to 5.4.

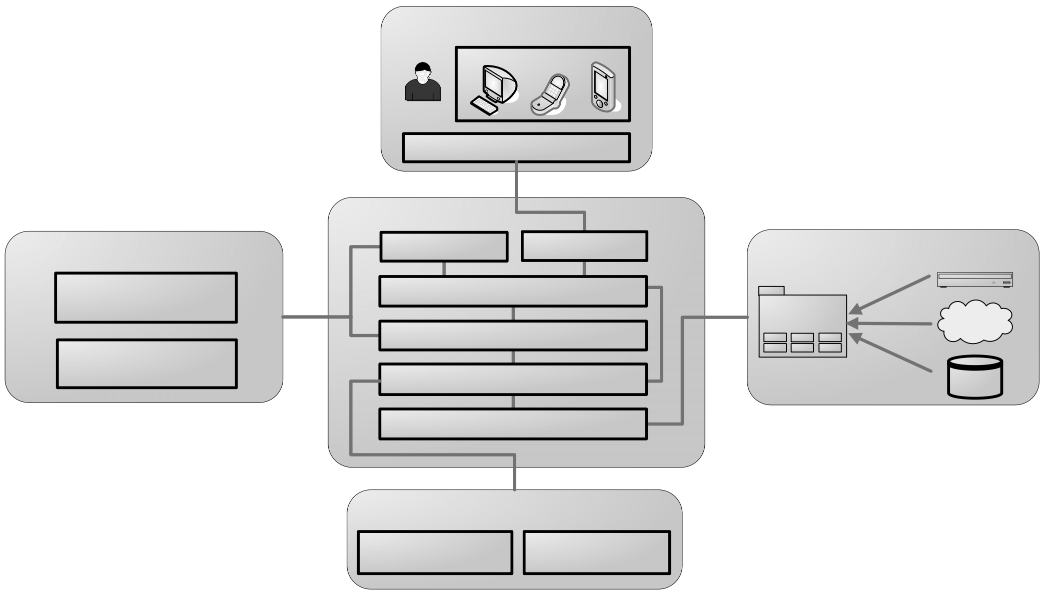


Figure 2. Overview of central unit and external connections and services.

The *end-user environment* is the point where the user interacts with the system. Various devices, such as TV screens, PC, handheld mobile or PDA are used. The central Blu-IS component identifies and keeps track of the connected devices and directs content streams simultaneously to multiple devices, according to the settings and user requests. Each user is furthermore identified and logged in to the system, in order to realise personalisation. Several users can be logged in and using the system in parallel, where each user may use one or multiple devices to interact with the system. The actual interface and functionality presented to the users are thus adapted for the devices used, with their own limitations and possibilities, and the running applications.

Blu-IS is the component which connects to all services and user devices. Its inner design is shown in the middle block in Figure 2. Starting at the bottom, the first two components provide the main functionality to handle content and content metadata. *The Content Retrieval & Serving* (CRS) is responsible for the communication via the different connections of the Content Services. Retrieved content may be directly streamed through the system, cached or stored locally. When downloading content, it communicates upwards in the component structure by events, to signal when the content has become available. As the name indicates, apart from retrieval, it also acts as a server for distributing local content, created by the users. Incoming external requests are handled and if accepted, the data transfer will be handled.

The *Package Handling* (PH) component communicates with the external IP Services concerning metadata descriptions and queries of package CRIDs. It enables the system to work with and parse packaged content as well as editing or making new. Naturally, the PH keeps track of a CRID for locally stored or cached content to reduce communication. Furthermore, for any content that a user wants to publish, the PH communicates with the CRID Authority to receive a unique CRID and make this public. Thereafter, other systems can retrieve it via the CRS.

Personalisation is described in detail in section 5.4 and we therefore only mention here that it communicates with the PH, System Services and the *Blu-IS Application* component. The latter is the session manager for keeping track of all active users, applications and devices. This means linking the applications to the content, as well as personalisation and presentation functions. The communication with client applications running on connected devices is based on a request-response model via a *User Interface Dispatcher*. In this process the interaction is directed via the personalisation, but there is also an option to skip this step as indicated in the figure, e.g. for a simple retrieval of a selected movie.

The last component, the *Presentation*, provides functionality to influence how the interaction or the content should be presented for a particular set of devices. As it is linked to the System Services, the user model can be used in this process to retrieve a user's preferences of which device to use or the look-and-feel using templates.

5.2 Semantically Enriched Content Model

The content metadata previously described, is fundamental for search and filtering of content. However, we imagine that due to the potentially vast amount of content, it is not enough to simply describe and classify the content, there must also be more intelligent ways of handling it. We therefore propose semantic knowledge models in addition to the TV-Anytime content model, which add possibilities for reasoning and deducing information about the content. The techniques we have used originate from the research area of Semantic Web (Berners-Lee et al, 2001) where ontologies are used for modelling semantic relations between concepts. We have used OWL, the Web Ontology Language (McGuinnes and Harmelen, 2004) to make a s*emantically enriched content model* that serves two main purposes – being able to:

- *reason and query the TV content*
- *add semantic knowledge about the application domain to achieve intelligent behaviour of applications*

To begin with, we have translated the TV-Anytime genre classification into OWL as a means of incorporating it into the system and enable querying. When translated into a *TV-Anytime content classification ontology* it is possible to use the structure and for example deduce that archery and climbing are both types of a sports genre. Unless the linkage between the genre classification concepts can be used, the applications will not "know" any semantic difference or connection between them. This is important for being able to group content into semantically related collections, which is useful when presenting and navigating available content.

Furthermore, we have defined mappings from the TV-Anytime annotation elements to existing ontologies for time, geography and lexical concepts which improve our possibilities to reason and query. Mapping time concepts to a time ontology

(Hobbs and Pan, 2004), e.g. the TV-Anytime annotation `<PublishedTime>` to corresponding time ontology concepts of `year`, `month`, `day`, `timezone` etc. enables temporal reasoning over the data. By this we mean handling time intervals and translations of expressions like “noon”, or “evening”. Apart of mappings from XML tags to ontology concepts, it is also useful when searching the free-text in e.g. the synopsis. For example, a user might look for movies that take place in the 1970’s. A simple search would only find those movies which metadata explicitly mention some year in this decade (a number 197*). However, by using concepts from time ontology we can also find those that instead wrote “the seventies”. A geographical ontology, as the Teknowledge Ontology of Geography[4], may likewise be used when searching for programs from “Europe”, where we extend the search with all the member countries to improve the results. When searching for programs about the region the user is located in, a geographical ontology gives us the area to use and possibly neighbouring cities. A lexical ontology furthermore helps to find synonyms of terms. We have incorporated the WordNet (Miller, 1995) linguistic ontology in OWL.

Our approach is not limited to use one ontology per domain. Geographical ontologies sometimes focus on listing countries while another on defining orientations (like ’westOf’ or ’isPartOf’). One time ontology may focus on time-zones and another on day, month, year, hour, etc. By combining the strength of different ontologies we can obtain a rich ontological structure. Furthermore, we intend to use *content ontologies* that model typical topics, with one ontology per each major genre. For example, sports ontology can model knowledge about sport equipment, famous players and well known competitions. Such content ontologies are an additional source that can be used when searching for content to semantically enrich the search. Our basic idea is thereby to go beyond keyword matching, which often is limited to finding only results which contain the exact keyword.

5.3 User Modelling

For the purpose of personalisation we have defined a user model (UM) to capture various concepts such as user’s age, location, possible physical disabilities, interests, dislikes etc. The model is divided into three parts; a model for user profile, one for user context and third for user history.

User profile (UP) is used for capturing personal information of the user and his or her media preferences. The personal data comprises gender, age, home address, native language, other spoken languages and level of advancement, occupation, possible hearing or visual capacities etc. Knowing this basic information enables content to be selected within the correct age limit of the user, finding local TV for the user’s home area and in a language that the user understands. Preferred devices per type of content can also be stored. Over time the UP furthermore stores the user’s content preferences, by linking a rating of like/dislike to the watched content. Since content is described with TV-Anytime metadata, there is enough information to identify the programme or movie in order to be exact in remembering the programme when storing the ranking preference. Remembering how a user liked a particular movie like “Braveheart” for instance is valuable. Moreover it can be used for drawing conclusions about the user’s preference of its genre of war/drama or the main actor Mel Gibson. As the content model has a searchable semantically model of the genres, it possible to combine all ratings for some genre by the ratings of semantically related (sub) genres.

User context (UC) describes the present situation of the user. It comprises concepts of the time, location, devices, audience and even the mood of the user. The purpose is to enable the system to achieve a more dynamic behaviour and adapt to the current situation. The time of the day may for instance be used in content filtering to affect the choice of device and content, e.g. one user may prefer a selection of news programs to be shown on the personal screen each morning. Furthermore, the currently available devices are a part of the user context. So is also the fact if the user is alone, or part of a group audience that share a device.

User history (UH) contains information of the previous behaviour of the user. It stores every action that the user takes and maintains a program list of the watched programs in combination of the time of watching. The purpose of this information is for attempts to discover patterns in the user’s behaviour. The program list is structured with a clear focus on time and comprises concepts of the week number, the day of the week and exact start and end time. The model can thus be used to find which programs, or types of programs, that are watched e.g. on a Tuesday evening. The UH becomes rather extensive after a lot of user interaction with the system. The time structure will

4 http://reliant.teknowledge.com/DAML/Geography.owl

probably also contain a lot of useless, redundant and duplicate information after some time of use. Therefore it is necessary to have a process which can filter the structure periodically and keep only the valuable information. We have for this sake chosen to use two instances of the user history repository; a short term user history which keeps all information, and a selective long term user history. The latter keeps all information which proved to be valuable after filtering the short term history, which occurs at the end of a session when the user logs out. This process also updates the UP. The UM thus develops and grows over time as the user interacts with the system. In an initial state it may however likely suffer from a cold start problem, due to few stored preferences, or that the first interactions influence the behaviour of the system too much. We attempt to avoid this by employing simple stereotypical user templates which are selected at user registration by letting the user fill out a brief form with questions of user preferences.

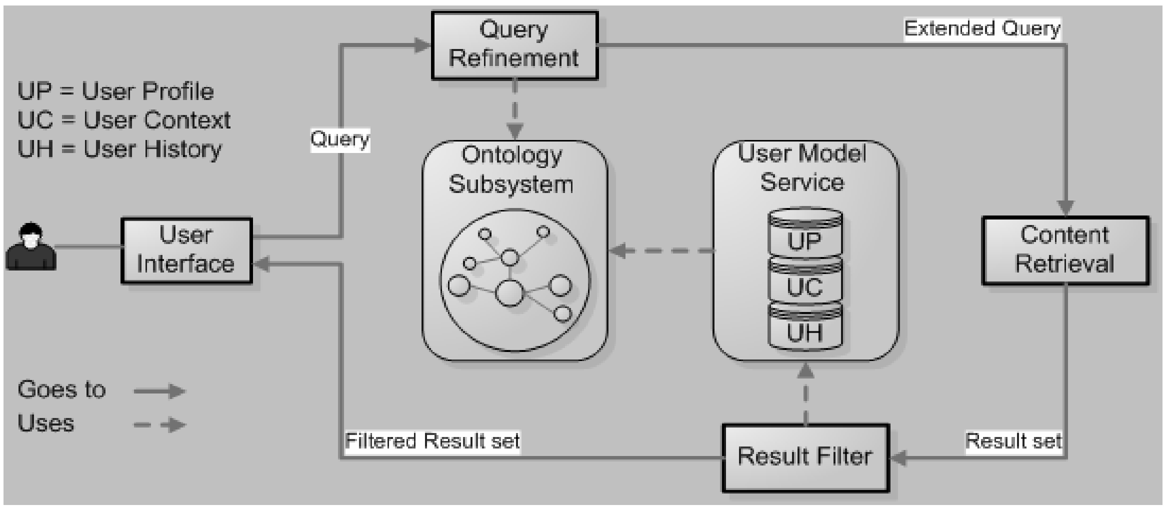


Figure 3. Personalised package search process with semantic refinement.

5.4 Personalised Content Search

This section describes the personalised content search functionality of the Blu-IS Personalisation component. Our basic idea is to add personalisation in the step between the Blu-IS Application and Package Handling when searching for content. This may occur when navigating through available content, when searching for something specific by entering keywords, or when asking the system to make a suggestion. In all cases, we aim at supporting the user by filtering the information based on the user's own perspective. The process affects the results found in the search in the following aspects:

- A smaller, more narrow result set
- Results contain the items ranked as most interesting for the user
- Results contain the items most semantically related to any given keywords
- Goes beyond word matching search and also considers semantic related concepts
- Results are categorised with links to semantic concepts
- Semantic links can be used to show the path from search query to results

We illustrate this by stepwise walking through content search process, depicted in Figure 3. Let's imagine as example that the user via the user application interface enters the keywords "*army 1940's*" and asks the system to search. This initial query expression of keywords **(*k_1, …, k_n*)** is analysed in a query refinement process which aims at adding extra semantic knowledge. By using the ontologies of the content model, we first search for modelled concepts with the same name as the keywords. We can in this case get hits in history and time ontologies, where "army" and "1940's" are found and thereby now known to belong to a history and time context. Second, since it is not sure that content metadata will use the exact same keywords, we add synonyms from the WordNet ontology, as well as semantically close concepts from the content ontologies. E.g. apart from direct synonyms, a closely related concept such as "World War II" is found through a semantic link of "army" – "war" and "1940's" – "1945", which furthermore links to the geographical concept "West Europe" which furthermore links to "Great Britain", "Germany" etc. However, this leads to a problem that the original keyword should be valued higher than related concepts. We solve this by adding a numerical value of semantic closeness, σ. In our initial algorithm the original keywords and synonyms receive σ value of 1.0, related ontology concepts within one node distance 0.75 and two nodes away 0.5. Third, we enrich the search query by adding every occurrence we found together with a link to the corresponding ontology concept. The query is thereby refined to new query expression of keywords $(k_1, \ldots, k_m)$ $(m \geq n)$, links from keywords to ontology concepts $(\varsigma_1, \ldots, \varsigma_m)$ and corresponding semantic closeness $(\sigma_1, \ldots, \sigma_m)$. Thereafter, the keywords in the query are mapped to TV-Anytime metadata items, in order to make a search request to the Metadata Service. From this content retrieval process the result is a collection of CRID references to packages which has matching metadata.

The next step in the process is result filtering, which aims at producing rankings of the search result in order to present them in an ordered list with the most interesting at the top. Furthermore is performs deletion of items in the list that are unsuitable e.g. content with an 18 years age limit for younger users. The deletion is a straight forward task of retrieving data on the user's parental guide limit or unwanted content types. The ranking consists of a number of techniques that estimate rankings from different perspectives:

- *keyword matching*
- *content based filtering*
- *collaborative filtering*
- *group filtering*
- *context filtering*

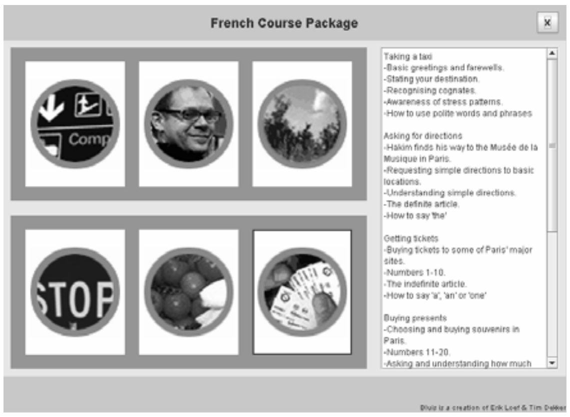
To begin with, packages are sorted based on a keyword matching value i.e. to what extent their metadata matched the keywords in the query. This can be calculated as average sum of matching keywords multiplied with corresponding σ value, in order to adjust for semantic closeness. Content based filtering (Pazzani, 1999) is furthermore used to predict ranking of items that the UM not yet have any ranking of. This technique compares the metadata of a particular item and searches for similarities among the contents that already have a ranking, i.e. that the user has already seen. Collaborative filtering (Shardanand and Maes, 1995) is used to predict the ranking based on how other similar users ranked it. Furthermore, context based filtering can be used to calculate predictions based on the user's context, as mentioned in section 5.3. If there is a group of users interacting with the system together, the result needs to be adapted for them as a group. This can be done by combining each persons filtering to create group filtering (Masthoff, 2004). Finally, the ranking value from each technique is combined by summarising the products of each filter's ranking value and a filter weight.

5.5 *Personalised Presentations*

In order to make the personalisation more transparent to the user, the path from original keyword to resulting packages is shown when the results are presented. The synonyms and other semantically related terms are also made explicit to the user as a feedback in order to avoid confusion when presenting the recommendation (e.g. when a movie with a different title is recommended than the original keyword given by the user). Since the links from keyword to related ontology concepts are kept we can present then in the user interface. Furthermore, we use them to group the result set, as well as in an earlier stage in the search process, when used to consult the user to find the appropriate context.

6. Implementation

A prototype of the system described has been developed and implemented in cooperation with Philips Applied Technologies. The fundamental parts of the IP and system services, content retrieval, packaging and personalisation are covered in this implementation. Our initial focus has been on realising the underlying TV-Anytime packaging concepts and personalisation and not so much on the Blu-ray. Currently, geographical and synonym ontologies have been incorporated in the prototype. The integration of content ontologies is still under development. A simple user application for searching and viewing packages has also been made, which is exemplified in Figure 4.



Taking a taxi
-Basic greetings and farewells.
-Stating your destination.
-Recognising cognates.
-Awareness of stress patterns.
-How to use polite words and phrases

Asking for directions
-Hakim finds his way to the Musée de la Musique in Paris.
-Requesting simple directions to basic locations.
-Understanding simple directions.
-The definite article.
-How to say 'the'

Getting tickets
-Buying tickets to some of Paris' major sites.
-Numbers 1-10.
-The indefinite article.
-How to say 'a', 'an' or 'one'

Buying presents
-Choosing and buying souvenirs in Paris.
-Numbers 11-20.
-Asking and understanding how much

Figure 4. Example view of package.

For test purpose a database with metadata on 500.000 movies from the Internet Movie Database (IMDB[5]) has been created. This has been converted into TV-Anytime phase 2 metadata. Our database also contains 1 million ratings of 4000 movies from 3000 users which have been imported from the EachMovie[6] dataset and are used by the prototype's first implementation of content recommendations. Our system currently allows a single user as well as multiple users to log in, where the system adapts to make recommendations for a single and a group of users correspondingly.

The User Model Service is an external service. This enables the use of multiple user models for collaborative filtering, improves the performance of the process and allows for an analysis of user behaviours in large. Furthermore, the use of an external User Model Service implies possibilities for the user to access the UM from other locations than in the home. A positive effect is achieved for the cold start problem, when the users use it on occasions, like visiting friends or on vacation, and get the same personalisation as at home. However, it may be argued that it can lead to privacy or integrity problems if users are uncomfortable with the thought of having information about their behaviour stored somewhere outside their home system, no matter how encrypted or detached from the person's identity it can be done. These issues are currently outside the scope of the reported research.

The user devices that can be connected are currently a HDTV screen and a LIRC remote control which communicates through a JLIRC interface. Content Services can furthermore handle both local content as well as streaming content via IP. The implementation has mainly been made in Java, where connections of external services are realised by the Tomcat Web Server, Java Web Start, SOAP and RMI. The tools used for the semantic models are Jena[7] and Protégé[8].

7. Related Work

Similar research as presented in this paper has been performed by Hong and Lim (2005), who also propose using TV-Anytime for handling content in a personalised way. However, they focus on broadcasted content, whereas we consider also content from IP and removable media like Blu-ray. Furthermore, their solution for content search also uses keywords and user history to recommend content, though the architecture differs in that all processing occurs at a metadata server, to which the user history, genre and keywords are sent.

Research focused on interactive TV systems in a home environment, where typical users are family members, have been done by Goren-Bar and Glinansky (2004). Here content filtering and user stereotypes are used for capturing and using user preferences. Furthermore, various techniques for filtering content to produce recommendations of movies have been explored by Masthoff (2004), where several user models are combined to create group filtering. Another related work is the PTVPlus online recommendation system for a television domain (O' Sullivan et al, 2004), which is a personalised EPG system.

As we propose TV content modelling with the use of ontologies, related work can be found in the domain of the Semantic Web. Necib and Freytag (2005) have focused on query processing with ontologies, which aims at refining search queries with synonyms (and yet avoid homonyms). However, we intend to take this one step further in our process as we also use other semantically related concepts and a measurement for semantic closeness.

8. Conclusions

In this paper we describe a scenario and an approach for a connected ambient home media management that enables connections from IP and Blu-ray, where users can view and interact via multiple rendering devices like TV screens, PDA,

[5] http://www.imdb.com
[6] http://research.compaq.com/SRC/eachmovie
[7] http://jena.sourceforge.net
[8] http://protege.stanford.edu

mobile telephone or other personal devices. The interaction, especially in content search, is supported by a semantics and context-aware process which aims to provide a personalised user experience. This is important since users have different preferences and capabilities and to prevent an information overflow. We have presented a component architecture which covers content retrieval, content metadata, user modelling, recommendations and an end-user environment. Furthermore we have presented a semantically enriched content search process using TV-Anytime content classification and metadata. Our goal is to propose a fundamental platform that can be used further by applications and personalisation services. Our current research focuses on the exploration and design of appropriate example applications (e.g. personalized programme guide).

References

Ardissono, L., Kobsa, A., & Maybury, M. (Ed.). (2004) *Personalized Digital Television : Targeting Programs to Individual Viewers.* Boston Kluwer Academic Publishers.

Berman, S. J., (2004). *Media and entertainment 2010. Open on the inside, open on the outside: The open media company of the future*. Retrieved November 24, 2005, from http://www-03.ibm.com/industries/media/doc/content/bin/ME2010.pdf

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). *The Semantic Web*. Scientific American.

Bormans, J., & Hill, K. (2002). *MPEG-21 Overview v.5*. Retrieved November 24, 2005, from http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm

Chorianopoulos, K. (2004). *What is Wrong with the Electronic Program Guide.* Retrieved November 24, 2005, from http://uitv.info/articles/2004/04chorianopoulos

Earnshaw, N., Aoki, S., Ashley, A., & Kameyama, W. (2005) *The TV-Anytime Content Reference Identifier (CRID).* Retrieved November 24, 2005, from http://www.rfc-archive.org/getrfc.php?rfc=4078

Goren-Bar, D., & Glinansky, O. (2004). FIT-recommending TV programs to family members. *Computers & Graphics. Vol. 28. (pp. 149-156).* Elsevier.

Hobbs, J., & Pan, F. (2004). An Ontology of Time for the Semantic Web. In *ACM Transactions on Asian Language Information Processing (TALIP), Vol. 3. Issue 1.* ACM Press.

Hong, B., & Lim, J. (2005). Design and Implementation of Home Media Server Using TV-Anytime for Personalized Broadcasting Service. In O. Gervasi, M. L. Gavrilova, V. Kumar, A. Laganà, H. P. Lee, Y. Mun, D. Taniar, C. J. K. Tan (Eds.), *Computational Science and Its Applications – ICCSA 2005: Conference Proceedings, Part IV. Vol. 3483. LNCS (pp. 138-147).* Springer.

Kobsa, A. (1990). User modeling in dialog systems: Potentials and hazards. *AI & Society. Vol. 4. Number 3. (pp. 214-231).* Springer-Verlag London Ltd.

Masthoff, J. (2004). Group Modeling: Selecting a Sequence of Television Items to Suit a Group of Viewers. *User Modeling & User-Adapted Interaction. Vol. 14. (pp. 37-85).* Kluwer.

McGuinnes, D.L., & Harmelen, van, F. (2004). *OWL Web Ontology Language*. Retrieved November 24, 2005, from http://www.w3.org/TR/owl-features

Miller, G.A., (1995). *WordNet: a lexical database for English.* Communications of the ACM, Vol. 38 Issue 11. ACM Press.

Murugesan, S., & Deshpande, Y. (2001). Web Engineering, Software Engineering and Web Application Development. In Murugesan, S., & Deshpande, Y. (Eds.), *Web Engineering. Vol. 2016. Lecture Notes in Computer Science.* Springer.

Necib, C. B., & Freytag, J-C. (2005). Query Processing Using Ontologies. In O. Pastor, J. F. e Cunha (Eds.), *Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005. Vol. 3520. Lecture Notes in Computer Science (pp. 167-186).* Springer.

O' Sullivan, D., Smith, B., Wilson, D., McDonald, K., & Smeaton, A. (2004). Improving the Quality of Personalized Electronic Program Guide. *User Modeling & User-Adapted Interaction. Vol. 14. (pp. 5-36).* Kluwer Academic Publishers.

Pazzani, M.J. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review, 13, (pp. 393–408).*

Setten, van, M. (2005). *Supporting People in Finding Information: Hybrid Recommender*

Systems and Goal-Based Structuring. Telematica Instituut Fundamental Research Series, No. 016 (TI/FRS/016). Universal Press.

Shardanand, U., & Maes, P. (1995). Social Information Filtering: Algorithms for Automating "Word of Mouth". In *CHI '95 Proceedings: Conference on Human Factors in Computing Systems (pp. 210–217).*