TECHNISCHE UNIVERSITEIT EINDHOVEN
Department of Mathematics and Computer Science

Dynamic Integration of Web and TV content for Personalized Information Retrieval in Interactive TV

By

C.J.J. Smeets

Supervisor:
Dr. Lora Aroyo (TU/e)

Mentor:
Ir. Pieter Bellekens (TU/e)

Eindhoven, November 2007

Abstract

This thesis concludes my work on the ITEA Passepartout project. The Passepartout project envisions an ambient interactive personalized connected home environment. In this thesis a semantics-based framework, called SenSee, for next generation television is extended. SenSee provides personalized access to TV content in a cross-media environment. It allows for an integrated view on data harvested from heterogeneous and distributed Web sources. The ultimate goal is to support individual and group TV viewers (operating multiple devices, e.g. PC, set-top box and a mobile) in finding programs that suit best their interests. In this paper, I focus on the integration of multiple data sources, such as BBC data from BBC backstage, XMLTV and IMDB. Moreover, these sources are interconnected and mapped to external vocabularies, like OWL Time, Geo Ontology, TV Anytime genre classifications and WordNet. The integration of these sources lead to a gigantic dataset, it is necessary to do some performance tests, with the purpose to guarantee a fast performance of operations on the dataset.

The resulting dataset is controlled by a client-application, called iFanzy, by offering the user a faceted browsing view on the data. Users can search and browse the data based on facets for time, location and genre.

Preface

This master thesis is the final project of my study at the Department of Mathematics and Computer Science at the Eindhoven University of Technology (TU/e). The work described in this thesis was carried out at the Information System group at the Department of Mathematics and Computer Science at the Eindhoven University of Technology.

Special thanks to Lora Aroyo and Pieter Bellekens for their guidance throughout this project, sharing their experience in the field and for making this work an enjoyable experience. Also thanks to all those who contributed to this work, especially Geert-Jan Houben en Kees van der Sluijs for their input and evaluation during the work. My final word of thanks goes to my family who supported me unconditionally throughout this entire thesis.

September 2007,
Chris Smeets

Presentations and Collaborations

The work described in this thesis is also used in the following papers:

- SenSee Framework for Personalized Access to TV Content
  Lora Aroyo, Pieter Bellekens, Martin Bjorkman, Geert-Jan Houben, Paul Akkermans, and Annelies Kaptein. In: EuroITV 2007. Pages 156-165.

- Engineering Semantic-Based Interactive Multi-device Web Applications
  Pieter Bellekens, Kees van der Sluijs, Lora Aroyo, Geert-Jan Houben. In: 7th International Conference on Web Engineering (ICWE 2007), 16-20 July 2007, Como, Italy, LNCS 4607, pp. 328 - 343

- Semantics-based Framework for Personalized Access to TV Content: Personalized TV Guide Use Case. Pieter Bellekens, Kees van der Sluijs, Lora Aroyo, Geert-Jan Houben and Annelies Kaptein. In: SWChallenge 2007.

The work described in this thesis also was used in the following presentations:

- EuroITV 2007 Conference
- Semantic Web Challenge 2007

A demo of the client application can be found at:

- http://wwwis.win.tue.nl:8888/SenSee/SenSeeWebClient.html

The work done in this thesis is also used in iFanzy.

- http://www.stoneroos.nl/solutions/productdetails/3101
- http://en.wikipedia.org/wiki/IFanzy

Table of Contents

Chapter 1

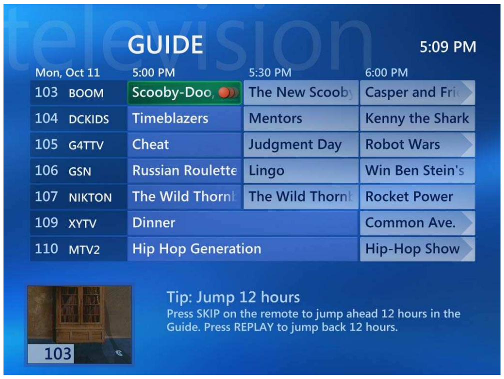
Introduction

The television (TV) is a widely used telecommunication system for broadcasting and receiving moving pictures and sound over a distance. The inventors had in mind at least a specific objective, if not a grand vision, but never as big as the reality now has become. First of all the TV became commercially available from the late 1930s. From that time the television set has become a common household device as a source of entertainment and news. Since the 1970s, VCR (VHS) tapes, later DVDs and currently also Blu-Ray and HD-DVD, have enabled the viewer to watch recorded movies and other kind of programmes.

The basic idea of the concept has not changed till the year 2000. Since that time, the area has started to digitalize. Cable companies started to upgrade their networks for digital broadcasting.
According to Ernst & Young[1], 35% of all Dutch households had a digital TV-connection (Cable, Satellite) in the beginning of 2007. The number will only increase due to technical, economical and political developments in the next years. For example, the Dutch government has accomplished the replacement of the analogue air-TV with the digital one, at the end of 2006. This digital switchover is also applied in other countries, or it will happen in the next few years.

Next to the digitalisation of the television, the DVD was introduced. The DVD replaced the popular VHS-tapes as media for watching on demand and recording purposes. Also the introduction of the hard disc recorders changed the manner of watching television. People can record their favourite programmes and watch them whenever they want.
There are three different types of these personal video recorders (PVR). Namely the Hard Drive-Recorder, the media-center (computers containing recording software) and the set-top box (receiver of digital TV).

One of the features of digital TV is an advanced Electronic Program Guide (EPG). The EPG software is included in the digital receiver. It is an on-screen guide to scheduled broadcast television programs (e.g. movie, news bulletin), allowing a viewer to navigate, select, and discover content by time, title, genre, channel etc, by use of their remote control or keyboard. The graphical user interface enables the display of programme titles, descriptive information such as a synopsis, collaborators, year of production, the channel name and the programmes on offer from subchannels such as pay-per-view or video-on-demand services, programme start times, genres and other descriptive metadata. An EPG allows the viewer to browse programme summaries, search by genre or channel, immediate access to the selected programme, reminders, and parental control functions. If the device is capable of it, an EPG can enable one-touch recording of programmes.



GUIDE

5:09 PM

Mon, Oct 11	5:00 PM	5:30 PM	6:00 PM
103 BOOM	Scooby-Doo,	The New Scooby	Casper and Fri
104 DCKIDS	Timeblazers	Mentors	Kenny the Shark
105 G4TTV	Cheat	Judgment Day	Robot Wars
106 GSN	Russian Roulette	Lingo	Win Ben Stein's
107 NIKTON	The Wild Thornb	The Wild Thornb	Rocket Power
109 XYTV	Dinner		Common Ave.
110 MTV2	Hip Hop Generation		Hip-Hop Show

103

Tip: Jump 12 hours

Press SKIP on the remote to jump ahead 12 hours in the Guide. Press REPLAY to jump back 12 hours.

Figure 1.1: IPTV EPG screenshot

Using the feature time-shifting the user is able to watch a movie and in the meanwhile the recorder is storing the movie content. Live television can be paused for a break at every moment. The user can record a huge amount of programmes and the probability that something is recorded the user likes is high; in other words, quality content at home that can be watched on-demand.

Another area of development is the switching from SD-TV (Standard Definition) to HD-TV, High Definition - Television. HD-TV is the offer of content in a higher resolution, better sound quality and all the devices which are capable to process it. To give a comparison, normal broadcasts (and DVDs) are about resolution 640x480. HDTV has currently a maximal resolution of 1920x1080 (Full HD, 1080p). In 2006, for the first time in history, a big event like the world-cup in Germany was broadcasted in High Definition format. For High Definition television it is necessary to have a television supporting this higher resolution, which implies that the customers need a LCD or Plasma television. The movies in HD can also be recorded in HD, having a more detailed image and enhancing the movie experience. There is one problem; these HD movies won't fit on a traditional DVD, because higher quality also demands bigger storage. Therefore there are two successors of the DVD. One is called Blu-Ray Disc and the other one is HD-DVD.

From the content point of view it can be summarized that there is a huge increase in the number of TV-channels, new services introduced like video-on-demand, the storage of recorded quality programmes and the emerge of (TV) content via IP (e.g. internet) which leads to an indefinite amount of content a viewer can watch. The capability to control, access and present this content is a challenge for every user.

Ernst & Young[1] indicates that 65% of the digital connected homes in the Netherlands use the EPG. Currently, it is one of the most popular features of digital television. 87% of the participants in the investigation state that their viewing behaviour has not changed since the introduction of digital television. Their assumption is that the EPG is one of the keys to activate this change in behaviour. An EPG should be more user-friendly and more advanced than the current ones and help with finding the needle in the haystack. The current situation is that a user can view and navigate TV-guides on the internet (like TVgids.nl) using it is PC and that a user can view the EPG using the TV. But the two very similar applications, in fact with the same function, work independently.

The Passepartout project will go beyond the current situation description. The project aims for changing the concept of watching television, by combining internet, disc and traditional broadcast. This is a new approach in ICT history.

1.1 Project overview

The Passepartout project is a European project, in co-operation with South-Korea, which is targeting the area of convergence of digital systems and applications in home media-centres, in compliance with the ITEA (Information Technology for European Advancement) roadmap "The Road towards Convergence". The full specification can be found in the ITEA Passepartout Full Project Proposal [1].

In the recent past the issue of adaptable HD video content technologies to provide a scalable service for "any content, any platform, anywhere and anytime" was identified. The Passepartout project has been launched to solve this issue. Four key technologies in the project are:

- HDTV and scalable content (from high definition television to lower definitions)
- Blu-ray and Media-center (PVR) technologies
- Reactive and interactive content flows
- Wireless Access network

These technologies must show how scalable content can be achieved for a broad class of terminals (e.g. TV-screens, portable displays, other media-players) in a home network. A number of new system components like Blu-ray, WIMaX, MPEG-4 and TV-Anytime, will be the basis for the project's platform.

The consumer plays the central role in the project. For that reason the new technologies will have to be simple, self-explanatory and easy to use, intelligent, context-aware and adaptive, seamless and interoperable. It is clear that with the availability of hundreds of media signals interesting challenges for applying user profiles, personalisation and adaptive user interfaces will be faced.

The system must provide content that is more closely adapted to the consumer's needs and desires (personalisation), think about matching languages, learning needs, opinions, lifestyles and habits, in a fashion that classical broadcast TV cannot offer. Also the classical broadcast TV is completely passive; the content is created and delivered to the viewer, there is no possibility to respond. Reactive and interactive principles enable the user to communicate with the system (human system interaction). Furthermore the system learns from continually and automatically updating the user profile during monitoring of the user's programme choices and interactions.

[1] http://www.ey.nl/?pag=4313

The technical innovation is the coupling of the four emerging technologies of the project into a package that can be exploited in a media-center (Mass-Customization box) in a home network supported by a wireless access network. From an applications point of view, four fields will be affected by the new technologies: content, infrastructure and basic services, human system interaction and engineering.

NL-Passepartout represents the Dutch consortium within the Passepartout team. The entire 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 media-center (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 and 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 Document overview

The main text of this thesis is organised as follows. The current chapter (Chapter 1) gives a general introduction to the thesis. This background information describes the main project, wherein the thesis is done. In Chapter 2 the goals and problem definition of this thesis are explained. The thesis work is applied in the SenSee framework. What SenSee is can be read in Chapter 3. The following chapters describe the design (Chapter 4) and the implementation (Chapter 5). Chapter 6 concludes the work. Finally a collection of appendices is given. These appendices are added to the document to clarify the used techniques and schemes.

Chapter 2

Goal and Problem Definition

Interaction with multiple devices (mobile phones, PC, TV) and multiple applications (web browsers, EPG) has become a major trend in ICT. From the development in the ICT landscape new methods emerge, like TV programmes via the Web [3]. Not to forget the mobile environment, accessing media content by mobile phone or PDA. In other words there is a desire is to **bridge the concepts of TV, PC and also of mobile**. These methods together with the urge of people to create **content** causes an enormous information **overload** and the presentation of the content is hardly adapted to the preferences, context and to the history of the user. To tackle these problems a more generic framework in a cross-media home environment, named **SenSee**, **providing adaptation, integration and personalisation** has been developed within the Passepartout project. In the coming paragraphs the various problems which this framework tackles are described. Real life situations are used for clarification purposes.

Nowadays, if somebody is **interested** in the TV schedules tonight, there are two possibilities of **finding** this information; by consulting a TV-guide on the Web (e.g. tvgids.nl) or the EPG. However an EPG on the TV is only local accessible, outside the home it is not possible and the online **TV-guide is not connected with the TV at home**. The later means that the user is able to search and view the content, but it does not allow communication with the TV to setup a schedule for tonight or to bookmark it.

A user might be interested in a certain movie (on-demand) and consults for example the Internet. The user can view very detailed content about the movie and related material and it is quite plausible that locally some personalisation is applied. This personalisation means that the application/service gathers information about the user and gives for example recommendations for movies in a certain genre. However it **does not share** the gathered interests, preferences or history with the TV at home. This is due to **isolation of the applications**. Everything is processed **locally**. There is not an overall dataset containing the preferences or actions of the user. One of the ideas behind the **bridging** is the **creation** of an overall (**integrated**) **dataset enabling a more accessible, personalised and synchronised system**. If a person discovers a nice movie at work (accidentally, extensively searched or perhaps a recommendation) the system should be able to remember the **interest** and to play it in the TV environment at home later on. From this interest and moment the system can deduce a certain personal behaviour and use it for personalisation purposes in the future. For example a recommendation of programmes of a certain genre where the person is interested in.

The main objective of this work is to **extend** the **SenSee** framework with **methods** to **organize, store** and to **present** information from the **different distributed sources**, taking into account the **dynamically and personalised environment**.

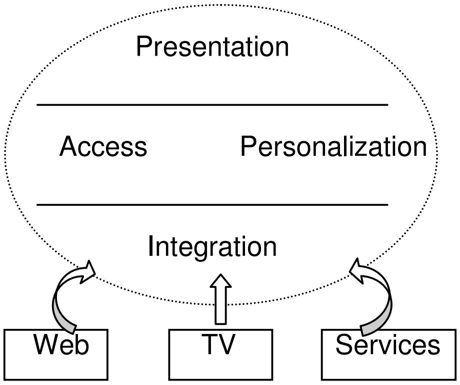


Figure 2.1: View of aspects

Integration of distributed data collections is one of the features of **SenSee**. Different heterogeneous content sources, like IP sources (IMDb, TV-Guides, BBC Backstage, Wikipedia), Blu-Ray disks (Operas, Movies stored on disk), locally stored and broadcasted content should be accessible and possibly be integrated. An important issue of this integration is the **difference** in **semantics**, layout and formatting of the different (TV) content sources. SenSee should organize and provide (semantically) **bridging of content** and disband the isolation.

Typically, a user wants to access the data (from the different sources), however SenSee itself is just a underlying framework. It allows an integrated view on data harvested from the different sources. Therefore an application on top of the framework (e.g. iFanzy) should provide access to the (semantically) integrated view in different environments. In this integration hierarchical structures are involved (e.g. genre, geography). The **presentation** is the main issue of this client application, especially the **hierarchical structures**. Another issue is the control of the application. In the current situation the remote control is the main device for searching content on **TV**. In a relatively small and simple set of content, navigation and searching is easy. Just a few clicks are necessary to reach the content the user is looking for. However for diverse, hierarchical and large sets the remote control is not an appropriate device. It is uncomfortable, the viewer wants to **relax** and only use the remote control for **quick navigation** and lookups, **not for complex search** commands. This is mainly due the lack of a more efficient interaction controller, like a keyboard or mouse. The **PC** setting is perfect for **searching** content, but is in general less suitable for relax and watch purposes. The TV is more sharing oriented and the PC is, like the definition says, personal. Each **environment differs** and the devices used have their limitations. A possible solution is multi-device support providing **adaptation**. The application adapts to the circumstances of the environment (TV, PC, mobile) in which the user resides.

Furthermore, SenSee focuses on the **personalised access** to content by adapting to user's preferences and abilities. Personalisation can be applied in alerts, reminders, filtering out content the user does not like and in ordering items by interest, time or genre. Finally, the presentation of content and services also plays an important role for the personalised **user's experience**.

From the description, research questions can be defined.

2.1 Research Questions

To tackle the problems defined above, several research questions need to be answered. The main questions discussed in this work are:

RQ1. How to prepare the content for SenSee? (Chapter 4)
The different content sources & vocabularies can differ in semantics, layout and formatting. (Semantic) mappings are necessary to realise integration.
The **starting point** is the already **existing SenSee framework** (Chapter 3) including some sources. Therefore the focus is broader than just **additions**, **refinement** and **extensions** are also possible. The three terms are caught in one word, identify.

Summarized, the tasks to create an integrated schema (common semantic schema) for SenSee are:

- Identify **content sources**
  What potential sources are available to provide content for the framework? For example schedules for TV or movie on-demand content. Identify the data schemes from these sources.
- Identify **vocabularies**
- Define **(semantic) mappings** between sources and vocabularies

RQ2. How is the performance of the integrated dataset (content)? (Chapter 5)
A slow performing application is not very useful for the end-user. The data should be accessible in a way that the user can easily view and search the content without too long waiting times. For a very large and diverse set it is more likely to encounter **performance** issues.

RQ3. How can the content collected by SenSee be presented? (Chapter 4)
For an end-user of the system the experience to access and search the content is important. The main issue is how the system can **present hierarchical structures** involved in the semantic integration to the user.

2.2 Goals and deliverables

The project delivers the following results:

1. Extended common semantic schema of the content in SenSee (from different sources and vocabularies)
    - New data sources and vocabularies
    - New mappings with others schemes, vocabularies
2. Implementation of the extended SenSee architecture
3. Implementation of the client-application (iFanzy)
4. Mid-term presentation on the research part of the project
5. Master's Thesis report describing the research and implementation parts of the project
6. Final presentation of the research, implementation and results of the project

Chapter 3

SenSee Framework

The SenSee framework is an embodiment of the Passepartout specifications [1]. It has been designed and implemented by developers at the TU/e and by Master students from the Computer Science department of the TU/e. The goal is to create a platform for personalisation and interaction in a cross-media environment. An important assumption is that the current development in digital media brings a content explosion that makes information filtering, search and personalisation increasingly important for home media-centres. The use of ontologies and semantic web technology will be very relevant in the project for doing things like user modelling, syndication and personalisation. In Chapter 4 this is explained in more detail.

SenSee is in general based in the following areas of the Passepartout project:

- "Creation of home media-centres that move beyond current players using MPEG-2 technology, to create true mass-customisation device for family entertainment."
- "Provide a step towards ambient intelligence through mass personalisation of reactive content."
- "Broaden the scope of Personal Video Recorder technology in Europe to include a new business model, based on packaging, as replacement for the TV channel."
- "Content packaging and personalisation to match the cultural and linguistic needs of the states of the EU and their economies."
- "Using W3C standards"

Basically, what these 5 points explain is that SenSee provides content that is more closely adapted to the consumer's needs and desires, thus matching languages, lifestyles and habits, in a fashion that conventional broadcast TV cannot offer. The user is able to control and to interact with the system, instead of only being a passive watcher. Besides that, it is a converged and personalised system; bringing together a number of content sources and offering a feature to easily find and watch the content based on the needs and desires. The ultimate goal is to support individual and group TV viewers (operating multiple devices, e.g. PC, set-top box and a mobile) in finding programmes that suit best their interests. To implement this, open specifications are used to enhance the interoperability. That is what W3C standards (XML, RDF/OWL, OWL-Time) provide.

The thesis work started with an already existing SenSee framework. This existing framework, like it was at the start of the thesis, was implemented after some basic steps; a requirements study, an architectural design and also the main parts of SenSee have been implemented. Thus, the thesis extends the framework, on basis of the results of the investigations and design of developers and students done before [2][4][6][7].

From the requirements study the main areas of interest have been extracted [5].

- Content retrieval: Support retrieval of content from distributed heterogeneous public collections of audio, video and textual material via IP and broadcast channels and from local storages and portable media. Integration of content according to a common specification should be supported.
- Content formats: TV-Anytime as the primary content metadata specification and the primary specification for content packaging providing flexible, extensible and topical organisation of content.
- Content conversions: Support dynamic (on-the-fly) conversion of content (e.g. XML, HTML pages, etc.) to a common content specification.
- Integrated semantic content model:
  - Support maintenance of multiple shared vocabularies (e.g. ontologies, thesauri)
  - Support a mapping infrastructure to connect concepts from different vocabularies
  - Support reasoning within this vocabulary pool
  - Enable the vocabulary pool to be publicly available
- Search: Support efficient search through all the content metadata
- Personalisation:
  - Support personalized content retrieval and presentation generation

    - Maintain a representation of the user and his context
    - Personalize the experience both for a single user as for a group of users
- User interface:
    - Support heterogeneous input of user and context data (e.g. sensors, web interfaces, TV remote control)
    - Generate different presentations for different user(s)-device(s) combinations
    - Support user interaction and content presentation with/on multiple devices
    - Multiple devices can interact independently of each other or can interact within the same interaction session
    - Support interaction of individual and group users

As can be seen in Figure 3.1 SenSee builds on a layered architecture. At the bottom the various heterogeneous content sources like IP, broadcast, media (e.g. DVD, Blu-Ray, HD-DVD) and local storage can be found, while at the top the client applications making use of the SenSee functionality. In between different stages, like retrieval and personalization, are implemented and taking the content as input and providing the desired functionality.

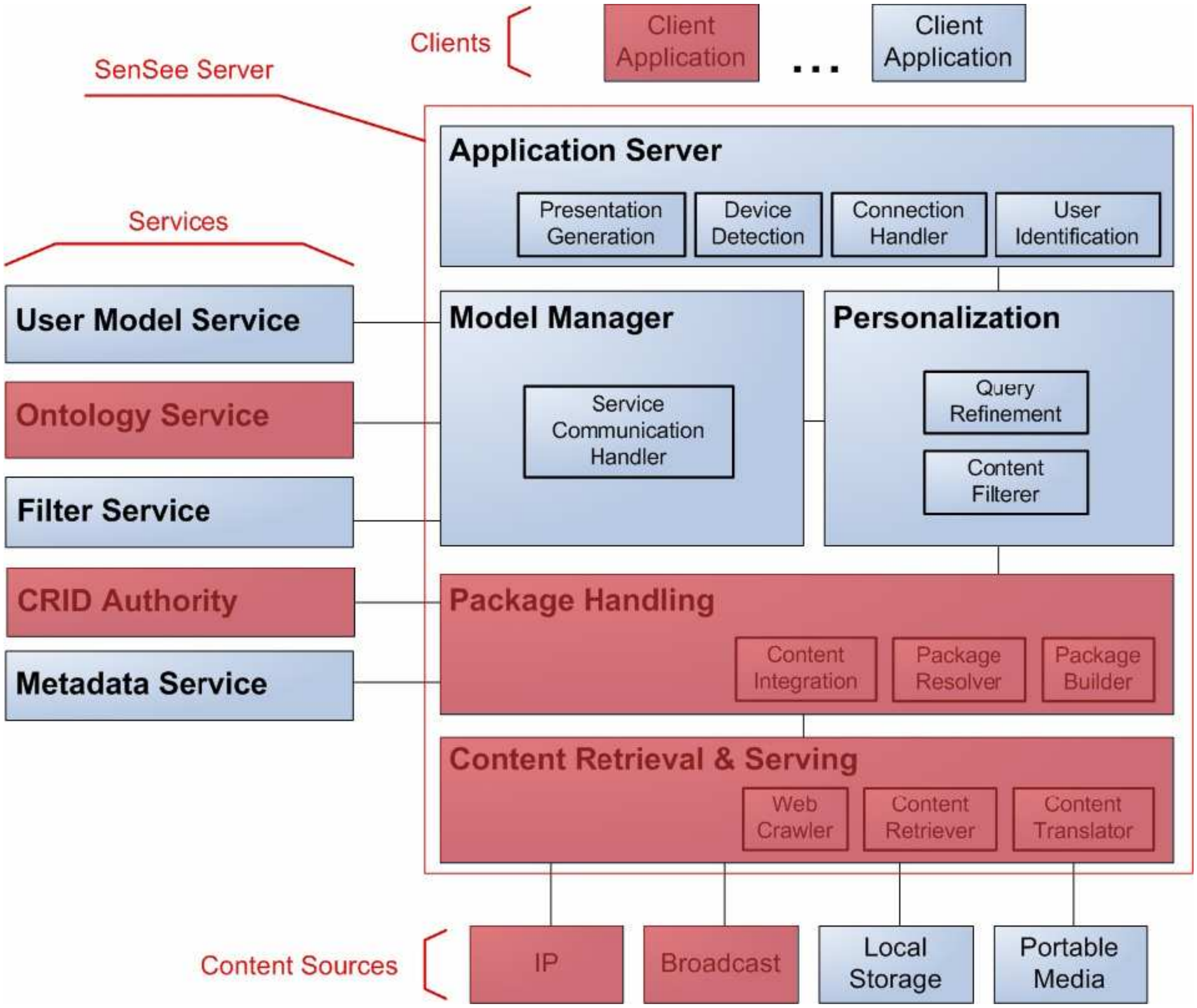


Figure 3.1: Architecture SenSee

The parts of the architecture of interest (in Figure 3.1 dark/red coloured) for this thesis will be explained globally; giving a general impression of the parts wherein the thesis work be implemented. The *server* is the part of the design which has the responsibility for integrating content from various sources and providing client applications access to the content, user data and adaptation functionality.

To separate responsibilities in the *server* design four interconnected layers have been designed. At the bottom the *Content Retrieval & Serving* for retrieval and integration of content, on top of that the *Package Handling* for transforming content and resolving packages, the *Personalization* layer which refines queries and filters content, followed by the *Application Server* providing access to the lower layers and identifying both users and devices.

The content input may be of various formats, sources like broadcast, CD, DVD and Blu-Ray, IP sources, local storage (e.g. personal content, saved content). The layer contains interfaces for retrieving content from each one of the specific sources that are supported. The retrieved content is sent upward to the *Package Handling* layer. To enable an integrated view on this content, the content must be modelled according to a content format, e.g. TV-Anytime (4.2.4).

In order to be able to understand different concepts like time, geographical places and content genres the Ontology Service (OS) is designed to be the main provider of semantics. It is the main storage for vocabularies and knowledge which is used extensively when searching and filtering content. For example, it can be used to extend search queries with synonyms, e.g. “bike” is extended with “bicycle”, and translates keywords to corresponding metadata, e.g. “morning” is translated to “`start 6:00:00, end 11:59:59`”. The personalization layer in the design makes extensive use of this service because it provides the knowledge to understand the descriptions of the content on one side and the user related data on the other side.

The storage and access of metadata (the data describing the content) is provided by the *Metadata Service*.
In Chapter 4 it will be explained that if the metadata about a TV programme or a TV broadcast schedule is collected, it may be in other formats (e.g. XML, HTML or plain text) then the format used in the service. In those cases the CRS layer is responsible for gathering and parsing such metadata, transform and pass it on to the *Package Handling* layer so that it can be stored in the *Metadata Service*.

In summary, the *Package Handling* layer provides the functionality to search and retrieve packaged content and manipulate the package structures. It has two main functional parts, a *package resolver* that is used for communicating with services, and a *package creator* used for creating packages by grouping content together or by converting other formats.

The top layer is the client applications. Various applications can use the SenSee platform. These are not only front-end interfaces but could also be sensored-devices that gather additional information about the user or the user’s environment. Applications can be made by different parties since the SenSee server interface is open for external connections. Applications can furthermore be running locally, remote on some device or on the Web. iFanzy is an front-end application operating on top of the framework.

3.1.1 iFanzy (deliverable 3)

iFanzy[2] is a personalised TV guide (client) application using the SenSee framework. The personalized EPG provides the viewer with an offer of programmes selected especially for this person, based on his or her personal interests or the mood he or she is in. iFanzy consists of Web-based front-end serving as a Web-based remote control point to the set-top front-end. The development is done by Stoneroos[3], a Dutch Interactive Television Developer, in cooperation with the Eindhoven University of Technology.

[2] http://www.stoneroos.nl/solutions/productdetails/3101

[3] http://www.stoneroos.nl

Chapter 4

Design aspects for the SenSee content modelling

Integration of distributed data collections is one of the features of SenSee. Different heterogeneous content sources, like IP sources (IMDb, TV-Guides, BBC Backstage, Wikipedia), Blu-Ray disks (Operas, Movies stored on disk), locally stored and broadcasted content must be accessible and be integrated. An important issue of the integration is the difference in semantics, layout and formatting of the different (TV) content sources. According to the requirements, SenSee must organize and provide (semantically) bridging of content.

In this chapter it will be explained how the content can be retrieved, how the different sources can be integrated and how it is possible to search and find content. At first sources are necessary to provide the framework with data (content, metadata). Thereby a question arises: What potential sources are available to provide data for the framework? Secondly, these sources differ in semantics, layout and formatting. The main question is how these sources can be integrated. Therefore a design is necessary which provides a common scheme to bring the sources together. It can be the case that such a common scheme lacks a little bit of power (expressiveness) in particular cases; for example to apply reasoning over a number of subjects, like time, geography and genre. Geographical vocabularies can be useful to provide a geographical structure and to enable SenSee to reason about locations or using a time ontology to be able to reason about time. Therefore a number of vocabularies (e.g. ontologies) are used to enrich and to provide structures/hierarchies. Structures are useful to make sense of the many and varied information involved. All these questions relate to the research question RQ1 and will be answered in the next paragraphs.

It is highly likely that a user wants to access the data, however SenSee itself only allows an integrated view on data harvested from the different sources. Therefore an application on top of the framework (e.g. iFanzy) should provide access to the integrated view. This integration needs special attention of the search and browser interface presented to the user. Lots of structures and hierarchies will be involved. The design of the client application will be explained in paragraph 4.4. This presentation issue relates to research question RQ3.

4.1 Data sources

In SenSee two main concepts of TV content delivery can be distinguished; the regular broadcasts and Video on demand (VOD). Broadcasted content distinguishes itself by being available at a certain point in time. For example if a person switches on the TV and selects BBC, there is a programme running. The programme starts at a specific time and ends also at a specific time. This type of content is a regular broadcast.

The other concept, Video on demand, has no time dimension. This content can be watched, like the definition says, on demand, at a moment a person wants. VOD services either stream content, allowing viewing in real time, or download it in which the programme is brought in and can be watched later on.
This VOD services are broad. It can be a service offered by a TV provider like pay-per-view, whereby a user selects a movie or television programme and has to pay for that particular item. It can also be something like the Dutch website “Uitzending gemist”, where content from the public Dutch channels can be found which was broadcasted in the recent past. The digitalisation is the main driver of the increasing interest of this concept and makes it possible to offer a wide range of VOD services. People can watch their favourite content whenever they want. In the current society the demand for these services is increasing and therefore it will be one of the basic features in the near future. It is one of the goals of SenSee, to move beyond the traditional concepts and offer new features; it offers the next generation of EPGs. This means that it has to support the concepts what we know now, like the regular broadcasts, but also implements new services, like VOD.

For the implementation of the concepts, commercial parties are needed, to deliver the content via the services they offer. Exclusive rights are necessary to use and show the content. However, there are no commercial parties in this project involved providing such content. This means that it is not possible to implement the concepts exactly how they should work in a production environment. To show that these concepts both work in SenSee a solution has been introduced. Metadata sources are proposed, which can mimic the working of the concepts. These metadata sources contain the information which describe the content but do not contain the real content (movie or programme itself).

Next to the search of the mentioned metadata sources, the investigation also focuses on enriching the metadata with for example photos and trailers and on providing structures/hierarchies in the metadata.

4.1.1 Broadcasts

The regular broadcasts are the heart of the traditional concept of TV. The metadata involved differentiates it from the VOD concept; namely start and end time. This time related metadata is needed to show the typical EPG style currently used on website like TVgids.nl and of course on your TV. What a person mostly gets is an overview of programme items ordered by time and channel (Figure 1.1). SenSee has to support this type of concept. Therefore an investigation is necessary to discover sources which fit best to provide SenSee with metadata mimicking the working of the concept.

Two metadata sources have been selected for further investigation:

- XMLTV[4] [Appendix B.3]
  The XMLTV source grabs programme information from several programme information/TV-guide websites (e.g. tvgids.nl). It covers various countries worldwide. There is a set of grabbers in the XMLTV software included to grab the metadata from the different TV guide sources and a grabber which is able to grab IMDb [Appendix B.3.1] metadata.

  XMLTV offers metadata that is necessary to show an overview of channels and schedules in an EPG. Title, production year, channel, time, genre and a small synopsis are the basic attributes provided. The amount of available metadata depends on the implementation of the grabber and the website where the original information resides. The function of the additional IMDb grabber is to enrich the metadata grabbed by the other XMLTV grabbers. However this metadata grabbed by the IMDb grabber is still very limited compared with the metadata presented on the IMDb website.

  XMLTV is completely configurable; it can be determined what grabbers in the package will be used, which channels for a particular grabber will be retrieved, how many days of metadata it grabs etc.

  If for example XMLTV data for 7 days (today and next 6 days) from 129 channels would be grabbed, the result would contain about 20.389 programme information items. This means 16 items on average per day per channel. The total amount of attributes (such as genre, title, actors) of all the programme information items exceeds 250.000.

- BBC backstage[5] [Appendix B.4]
  The BBC backstage is the BBC's developer network to encourage innovation in TV. Metadata feeds are available for developers to use on a non-commercial basis. The programme information from BBC broadcasts can be retrieved directly from BBC Backstage in TV-Anytime XML format.
  Note that the BBC source only provides metadata for the BBC channels. The metadata from this source in comparison to XMLTV is much richer in terms of how many fields are filled.

  It contains minimal 8 days (today + next 7 days) of 20 channels BBC TV and Radio schedule information. A random sample contained 1575 programme information items. If the attributes describing the programme information are included the total number of items is around 30.000.

[4] http://xmltv.org/wiki/

[5] http://backstage.bbc.co.uk

4.1.2 Video on demand

VOD is the concept of getting programmes on demand and it will be one of the basic services in the future. For this concept there is currently no commercial party involved in the project to, for example, offer a service on a pay-per-view basis. The solution is to find a metadata source which mimics the behaviour, showing that it is possible to integrate such type of concept in SenSee. There are some options to solve this issue; a lot of sources can act as a stand-in. It can be a movie database or one of the earlier found sources (XMLTV, BBC backstage) by omitting the time property. Remember, VOD, programmes without start and end time.

During the investigation the main focus was to search for movie, TV-series databases. Because these sources provide richly annotated metadata, such as titles, genres, filming locations, actors, biographies and photos. A lot of these sources have been investigated. For example the All Movie Guide, Movies.com and TV.com. The All Movie Guide is a commercial database of information about movie stars, movies and television shows. The metadata can be accessed on the web and is also available via the AMG LASSO media recognition service, which can automatically recognize DVDs etc. However this service is not for free. Movies.com is the second-most visited movie-related website on the Web and TV.com focuses on TV-series. None of them really complied with the demands; the metadata is too difficult to obtain and/or the metadata may not be used.

The Internet Movie Database (IMDb)[6] [Appendix B.1] contains a catalogue of every pertinent detail about movies and series, such as the cast, director, filming locations, languages, soundtracks, etc. Note that their metadata and software required to construct a copy of the database can be used for non-commercial purposes. Currently the metadata can be constructed and accessed locally through various interfaces like plain text files (text dump of the IMDB database) [Appendix B.1.2] or accessed via the website itself (HTML) [Appendix B.1.1]. The text dump is divided over a number of files. In the text dump the data is string formatted and separated by tabs. The disadvantage of getting the metadata from the website is the format in which it is delivered, namely HTML. It is a mix of content and presentation styles. Therefore it is more difficult to parse in comparison with a more content structured format. If the presentation layout changes it is plausible that a parser is not able to parse the content anymore.

The text dump of IMDb does not contain photos, movie trailer URLs. The HTML version does. Besides it, the text dump does not contain the unique identifier of a movie or person. These identifiers are used internally, but also by external sources to refer to a movie or person within IMDb.

Just some numbers, to imagine how large this set is:

- 1 million movies, whereof 379.871 theatrically released, 483.832 TV episodes, 61.119 TV movies etc. Having 8 million attributes describing a movie. Thus, on average 8 attributes a movie.
- 2.34 million persons good for 16.5 million collaborations

4.1.2.1 IMDb photos & posters [Appendix B.1.3]

Nowadays if a user consults a TV-guide or the EPG on the TV they are often very simple. To make an EPG more vivid and attractive, features like photo galleries and/or trailers are desirable. Users can easily recognize movies or persons only by viewing such a feature, or are even more attracted to see the movie or related material. The IMDb website contains photos and posters of movies and photos of persons. The photos can be found easily, they are part of the presented movie or person detail page. The gallery containing the photos is a sub page of such a detail page. Thus, if a user enters “Gladiator” and selects this movie from the result list then IMDb redirects to the movie details. At the movie detail page there is a link to the photo gallery and such sub page contains that gallery.

IMDb contains 14.000 posters, 42.000 movie images, 110.000 headshots, 109.000 person images, 38.000 headshots. Thus, in total around 300.000 items.

[6] http://www.imdb.com

4.1.2.2 IMDb trailers

Trailers can be seen as an attractor (a kind of commercial), offering the users (potential customers) a feature giving a better impression of a programme. It also makes an EPG more attractive, like the before mentioned photo gallery also does.

IMDb beautifies the movie information with trailers. They can be found via the same procedure as described in 4.1.2.1 (photos). Only the sub page differs; photo has become trailer.

These trailers are hosted by IMDb (internal) and provided by external sources (e.g. movie-list.com, rottentomatoes.com and apple.com). The external sources, is a link to a page which contains the trailer itself or to a page containing a link to another (sub) page where the trailer resides.
The internal service ceased to exist during the thesis. From the external sources only movie-list.com is a suitable candidate. The others sources contain too much commercials, disappear after a short time and do not offer older movie trailers (e.g. classic movies) or it is simply to difficult or time consuming to parse.

IMDb contains 37.000 trailer URLs.

Later on in the document, in paragraph 4.1.3.1, another solution to obtain trailers is proposed.

4.1.2.3 IMDb locations [Appendix B.7.1]

Geographical related metadata lend itself to a hierarchy; if we think about geography, the most likely idea is something like world, continent, country. In the investigation there is focus on this kind of hierarchies and structures; to be able to structure metadata in such a way that it can be searched or navigated easily or to make searching more powerful. It can enable applications to reason about locations.

A simple example: assume a user is interested in all movies shot in the Netherlands and a movie is only labelled with a location "Eindhoven". Without additional geographical information/hierarchy the system is never be able to figure out that Eindhoven is part of the Netherlands and that movies shot there are also shot in the Netherlands. Replacing this simple type with a reference in a geographical hierarchy, which means that Eindhoven is connected with for example the province Noord-Brabant and Noord-Brabant with the Netherlands, enables the application to find also the movies shot in Eindhoven.

IMDb locations[7] provides such a hierarchy, it is a collection of all locations which can be found in the metadata of IMDb. Each property, such as filming location, country of release, contains a value from this collection. The children of the root element are countries in the world, seas and oceans and some locations which cannot be defined properly, like a marine ship. Below these elements there is again a structure. The Netherlands for example is divided into provinces. These provinces are split up into cities and villages. Furthermore these cities and villages can have streets or other objects, like houses.
The USA has a similar structure; instead of provinces they have states.

A small example of how such a structure looks like:

```
Rancho Palos Verdes, California, USA
Grote Kerkplein, Rotterdam, Zuid-Holland, Netherlands
```

The IMDb set contains 48.612 unique filming locations divided over 217 top level elements (e.g. countries). Another geographical related movie property, country of origin, contains 218 unique countries.

Note that this paragraph explains how the data is structured, how it is, at IMDb locations. See 4.2.1 for the geographical vocabularies.

[7] http://www.imdb.com/LocationTree

4.1.3 Additional sources

During the investigation, additional sources were discovered which could be used to both enrich metadata and to solve certain issues with the possible sources which will be selected to implement the concepts. Videodetective and movie-list are possible sources to enrich the metadata with trailers. Wikipedia can be used to enrich the metadata with additional descriptions about persons, movies etc. The IMDb set also contains literature about movies and persons, however it is difficult to parse because it cannot be distinguished from the text, if some string describes a title or an author. It is not possible to distinguish "To Kill a Mockingbird" from "Harper Lee" without any knowledge that the former is the title of the book and the latter the author. Therefore some solutions have been investigated to solve the issue and to make it possible to unravel the attributes such as a title or author from the text.

4.1.3.1 Videodetective [Appendix B.2.1]

The reason of investigating this source is the same as for the IMDb trailers [4.1.2.2]; to enrich metadata, to beautify an EPG. Videodetective[8] claims to be the biggest provider of movie trailers. It contains trailers from theatrical releases, TV and DVD releases. It the investigation, only Videodetective was found as the only provider offering a free interface to get access to their database. The absence of commercials in their trailers is an advantage, especially for demonstration purposes.

The open interface is a Web service returning a XML document. The service is just a URL with some predefined variables to be set (e.g. search term, number of results). The returned XML document contains the metadata describing the trailer (and in fact the movie also), such as title, actors, release date and director.

A disadvantage of the service is that it does not allow a great amount of requests in a short time (e.g. 20 search commands per second). The interface also does not provide methods to retrieve all the items at once in a simple way. For example it is not possible to retrieve all the trailer links from the movies starting with a letter (e.g. A). Therefore it is quite complicated to copy the entire database and to convert it completely in another format. The request of a complete movie title is not a problem at all, that's what the service basically provides.

The exact size of the database is unknown, but the roughly estimated number is between 100.000 and 200.000 items.

4.1.3.2 Movie-list

Movie-list[9] is also a relative large trailer provider. It is one of external sites supported by IMDb. From a particular movie page within IMDb there is a reference to a movie-list trailer page. Disadvantages of this source are the size of these trailers (sometimes around 100MB, downloadable), relative short lifetime of the trailers and the manual added references within IMDb (there are more trailers available, but IMDb doesn't link to them).

However the determining factor is the format of the source. Movie-list is just a website without Web service. Thus the trailer link must be parsed from HTML. Also there is no additional check to control that the correct trailer has been found. This is due to the minimal amount of metadata, such as title and actors, available.

The roughly estimated number of items is between 10.000 and 25.000.

4.1.3.3 Wikipedia

Wikipedia[10] is a multilingual, free content encyclopaedia. It is written collaboratively by volunteers from all around the world. 75.000 active contributors work on 5.300.000 articles in more than 100 languages. There are almost 2 million articles in English.

[8] http://www.videodetective.com

[9] http://www.movie-list.com

[10] http://www.wikipedia.org

The articles are never complete; they are continually edited and improved over time. This in general results in an increase of quality. Users should be aware that not all the articles are of encyclopaedic quality from scratch, and sometimes may contain false or debatable information.

The idea of using Wikipedia is to offer additional information about movies or persons. The advantage of using Wikipedia is the large and very detailed multilingual description of a movie of person and with many references to other domains. Keep in mind that not all articles are extensively described. The page of the movie "Gladiator" is an example of a very detailed description, with references to the actors, locations, keywords, other movies etc. The disadvantage is the structure of an article; the articles are written by volunteers and they can determine how the article looks like. Just a simple example: one person uses a header title "Cast", the other one "Credits. It is all free text based, therefore anybody can write what and how he wants. There are some exceptions; some parts of an article contain a short overview about the movie. This is more or less a structured overview of short facts (called an infobox). Also the references to other articles can be obtained easily. However unravelling free text can be a difficult job.

Wikipedia vocabularies can be found in 4.2.3. Note that Wikipedia is not included in the final design, motivated in 4.3.1.1

4.1.3.4 Literature [Appendix B.9]

Movies and literature are closely related, many written stories are adapted to the screen and vice versa. Perhaps users could also be interested to read the book or to watch the movie. A literature data source can help in gaining more information about the movie literature.

The IMDb source contains metadata about books and articles related to a movie or person, however the data describing it is not structured; it is impossible to correctly identify what the author, title, publisher of a book is. This is caused by free text (string) without mark-up. It is not possible to distinguish "To Kill a Mockingbird" from "Harper Lee" without any knowledge that the former is the title of the book and the latter the author. Therefore a solution for the issue has been investigated; in order to find repositories providing a system to match the metadata from the IMDb set and to discover the book metadata.

ISBNdb.com[11] [Appendix B.9.1] project is a database of books in different languages. They provide online and remote research tools which can be used by individuals and scientists. They provide a free and easy accessible Web service. It is based on standard HTTP queries and XML based responses. A maximum of 500 requests can be submitted per day. Keywords, book title, author, publisher, topic or ISBN keywords can be feed to the service, resulting in a XML document containing the found items.

The database contains almost 3 million books, 900.000 authors and 188.000 publishers.

The WorldCat[12] library [Appendix B.9.2] is another repository. It offers a service of XML-formatted metadata for published content. It contains over 4 million records representing many popular titles and a selection of unique items. The set can only be used with permission from the copyright holder. The set contains metadata fields such as title, language, author, publisher name and year, edition, genre and ISBN.

Note that the literature sources are not implemented in the final design, motivated in 4.3.1.1

4.2 Vocabularies

What vocabularies can do in the project is to provide a common scheme in which the metadata can be modelled, give more meaning to certain terms, build/use hierarchies for classification purposes etc. If it is not clear what vocabularies are and what their added value is, then the introduction of Appendix B can help in understanding the subject.

[11] http://www.isbndb.com

[12] http://www.worldcat.org

Geographical vocabularies are useful for providing geographical structures and to enable applications to reason about locations. In 4.1.2.3 it is explained that a geographically vocabulary can help in finding movies, for example if somebody is interested in all actors born or movies shot in the Netherlands. Probably in most metadata sources it is only described that a movie has been shot, for example, in "Eindhoven". Without any additional geographical information the application is never able to figure out that "Eindhoven" is part of "Noord-Brabant" and "Noord-Brabant" part of the "Netherlands". Thereby resulting in zero movies if the geographical knowledge is not used and by using it a number of hits will be returned.

WordNet can be interpreted and used as a lexical ontology in ICT. A prominent example of using WordNet, as it is as an ontology, is to determine the similarity between words. This can help a user in finding items. For example, the user enters the keyword "car". The application receives this keyword and starts searching for all items containing the keyword "car". Unfortunately, the creator of the metadata has entered "automobile". The system delivers no results at all. By using WordNet the system will find the item containing the keyword "automobile". Automobile and car are synonyms and WordNet was created to discover such relationships between words.

Wikipedia is one on the central knowledge sources of mankind and is maintained by thousands of contributors. Wikipedia currently only supports keyword-based search and does not allow more expressive queries like "Give me all movies shot in the USA and having the genre Action". This restricts the overall utility of Wikipedia. That's why some organisations tried to create a custom ontology for it.
What Wikipedia can add to the project is to enrich for example movie and person metadata.

TV-Anytime is a content metadata specification. It provides a very extensive metadata specification which allows for a very rich and detailed content classification and description. The content classification for example contains a very rich classification of Genres (Action, Thriller) and Collaborators (Actor, Director). TV-Anytime is a candidate for the common scheme to model the metadata in.

OWL-Time is an ontology of time that expresses temporal concepts and properties common to any formalization of time. The lack of interpretation constrains a search command to a simple string matching between the string and the content metadata. For example, take the string "2007-08-06". This string can be identified as a time concept by using OWL-Time, which enables a search for items between 00:00 and 23:59 on this date.

4.2.1 Geographical [Appendix B.7]

The Getty Thesaurus of Geographic Names[13] (TGN) [Appendix B.7.2] is a structured vocabulary providing geographical information. It contains around 1.1 million names, place types, coordinates, and descriptive notes, focusing on places important for the study of art and architecture. TGN is also applied in another project running at the Technical University, the E-culture project.

A disadvantage of this vocabulary is the lack of an API/Web service which can be used to access the repository easily. This means that the complete repository must be copied to the project, to enable access to the data. For example if movie metadata contains the filming location "Eindhoven" and the system has to do some reasoning about it, somehow it is necessary to access TGN, to enable the reasoning. For example to discover that Eindhoven is a city in the Netherlands.
Another disadvantage is a too strong focus on important places in art and architecture. This means that less interesting locations for the area of art and architecture are not included.

Geonames[14] [Appendix B.7.3] is another geographical vocabulary. This large and public geographical database contains a world-wide list of geo-administrative and geo-physical names. It provides data such as geographical names and postal codes. The database contains over 6.5 million entries for geographical names whereof 2.2 million cities and villages.

[13] http://www.getty.edu/research/tools/vocabulary/tgn/index.html
[14] http://www.geonames.org

Geonames offers a (RDF) Web service to query the repository and to process the results. The repository is very large, containing hierarchical structures of data. It also extends the data with some useful additional information, like coordinates, neighbouring places and references to Wikipedia pages describing the location in more detail. Furthermore, there is RDF dump of the repository (containing 94 million triples) available. Another feature is the availability of a Java library helping you to easily access the Geonames Web services with Java. Geonames is a better solution as a more general repository, because TGN focuses on art and architecture.

A third option is the before mentioned IMDb locations (4.1.2.3). From this location tree a straightforward ontology can be designed. One of the definitions of a tree: *is a way of representing the hierarchical nature of a structure in a graphical form.* The definition already suggests that there is a structure involved; this can be used to build an ontology. The tree contains the important locations in the area of movie and related material. Countries are the broader terms at the top and in the narrower parts more specified locations can be found, such as houses, streets, cities, villages. The strings from the location tree are use to build a hierarchy, from that a structured model similar to the RDFS subclass structure has been created, i.e. Eindhoven is subComponentOf Noord-Brabant, is subComponentOf Netherlands.

4.2.2 Wordnet [Appendix B.8]

WordNet[15] is a lexical database of the English language. Nouns, verbs, adjectives and adverbs are grouped into sets of synonyms called *synsets*. WordNet was created at the Princeton University in 1985. In 2006 the database contained about 150,000 words organized in over 115,000 *synsets*. The goal of WordNet is to present a combination of a dictionary and thesaurus and to support automatic analysis of text and artificial intelligence. The database and software tools can be downloaded and used freely.

W3C also created WordNet[16] in an RDF/OWL [Appendices A.2 and A.3] representation of the entire structure of WordNet. By doing so, WordNet data can be accessed via RDF APIs and query languages, and to be mixed with non-WordNet data, as well as with other lexically-oriented material, such as extensions to, and derivatives of, WordNet.

An example of using WordNet is if a movie is related to a keyword, e.g. car. WordNet can help extending it with synonyms such as auto, automobile, machine and motorcar. It is possible to use WordNet in different layers of SenSee. For example at query time or at the retrieval; at query time the keywords the user has entered can be extended and at retrieval, the metadata keywords can be extended.

4.2.3 Wikipedia

A large body of information and knowledge is often already available in structured form, yet not always accessible as such on the Web. Wikipedia is one of the better examples. Wikipedia (4.1.3.3) is the largest publicly available encyclopaedia on the Web. Integrating open data saves time and effort to re-enter data that is already available and leaving the data and editing where it belongs: at its origin. Wikipedia has the problem that its search capabilities are limited to full-text search, which only allows very limited access to the database. In other words it currently only supports keyword-based search and does not allow more expressive queries like “Give me all cities in the Netherlands with more than 10.000 inhabitants” or “Give me all Dutch painters from the 18th century.” This restricts the overall utility of Wikipedia.

Wikipedia articles consist mostly of free text, but also contain types of structured information, such as:

- Infobox templates. An example is the movie infobox containing metadata such as actor, title, writer, director, producer, releasedate and runtime.
- Categorisation information. Category films: Films by country, Films by director, Films by genre, Films by producer, Films by year. 1999 films, 2000 films are subcategories from Films by year. Thus, this categorisation is an ordering of items in categories on different levels.
- Images, sometimes metadata is included (e.g. summary containing the description, source and creator of the image)

[15] http://wordnet.princeton.edu/

[16] http://www.w3.org/TR/wordnet-rdf/

- Links to external Web pages.
- Internal links, to other articles

This structured information can be extracted from Wikipedia and can serve as a basis for enabling sophisticated queries against Wikipedia content.

Wikipedia3 [17] is a conversion of the English Wikipedia into RDF [Appendix A.2]. It was described using a custom designed ontology and enriched with elements from SKOS[18]. The dataset currently combines structural information like link and category relationships with basic per-page metadata (title, contributor). In other words all the links of a Wikipedia article are recognized and stored as a link to a subject or article. Currently the dataset consists of roughly 47 million triples and comes in all major RDF serialization formats, such as RDF/XML. The *Wikipedia*^{3} dataset looks like this:

```
<rdf:Description rdf:about="http://en.wikipedia.org/wiki/Gladiator_%28film%29">
<rdf:type rdf:resource="http://www.systemone.at/2006/03/wikipedia#Article"/>
<dc:title>Gladiator_(film)</dc:title>
<dc:contributor>SmackBot</dc:contributor>
<dc:modified>2006-03-24 02:03:51.0</dc:modified>
<skos:subject rdf:resource="http://en.wikipedia.org/wiki/Category%3A2000_films"/>
<skos:subject rdf:resource="http://en.wikipedia.org/wiki/Category%3AAmerican_films"/>
<wiki:internalLink rdf:resource="http://en.wikipedia.org/wiki/2000"/>
<wiki:internalLink rdf:resource="http://en.wikipedia.org/wiki/Academy_Award_for_Sound"/>
<wiki:internalLink rdf:resource="http://en.wikipedia.org/wiki/Colosseum"/>
<wiki:internalLink rdf:resource="http://en.wikipedia.org/wiki/Russel_Crowe"/>
<wiki:internalLink rdf:resource="http://en.wikipedia.org/wiki/Hans_Zimmer"/>
<wiki:internalLink rdf:resource="http://en.wikipedia.org/wiki/Roman_emperor"/>
</rdf:Description>
```

The *DBpedia*[19] dataset is a large multi-domain ontology which has been derived from Wikipedia. It provides a system which extracts structured information from Wikipedia and makes this information available on the Web. It allows you to ask sophisticated queries against Wikipedia and to link other datasets on the Web to Wikipedia data. The RDF is used a flexible data model for representing the extracted information. It can be downloaded in an RDF triple store or the data set can be queried via an endpoint on the web.

The dataset currently consists of around 103 million triples, extracted from the English, German, French, Spanish, Italian, Portuguese, Polish, Swedish, Dutch, Japanese, Russian, Finnish, Norwegian and Chinese version of Wikipedia. It describes 1.950.000 concepts, including at least 80.000 persons, 70.000 places, 35.000 music albums, 12.000 films. It contains 657.000 links to images, 1.600.000 links to relevant external web pages and 207.000 Wikipedia categories.

A small overview of some concepts in the dataset:

- City: Berlin, Amsterdam
- Country: Netherlands, Spain, Italy
- Politician: Angela Merkel
- Music album: Thriller, Like a Virgin
- Director: Oliver Stone
- Film: The Lord of the Rings, Pulp Fiction
- Technical Standard: HTML, RDF

Each of the 1.95 million resources described in the dataset is identified by a URI reference of the form `http://dbpedia.org/resource/Name`, where Name is taken from the URL of the source Wikipedia article, which has the form `http://en.wikipedia.org/wiki/Name`. Thus, each resource is tied directly to an English-language Wikipedia article.

Every DBpedia resource is described by a label, short and long English abstract, a link to the corresponding Wikipedia page and a link to an image depicting the concept. If a concept is multilingual then the short and

[17] http://labs.systemone.at/wikipedia3

[18] http://www.w3.org/2004/02/skos/

[19] http://dbpedia.org

long abstracts within these languages and links to the different language Wikipedia pages are added to the description. There are almost 2 million English abstracts and 164.000 Dutch abstracts. A movie description in DBPedia looks as follows:

rdf:type	dbpedia:class/yago/MotionPictureFilm103789400
rdfs:comment	":This article is about the 2000 film. For the 1992 film, see Gladiator (1992 film)."@en
rdfs:comment	"Gladiator is een Amerikaanse film uit 2000 over de Romeinse tijd. … "@nl
rdfs:label	"Gladiator (2000 film)"@en
rdfs:label	"Gladiator (film)"@nl
dbpedia2:name	"Gladiator"@en
dbpedia2:image	<http://upload.wikimedia.org/wikipedia/en/8/8d/Gladiator_ver1.jpg>
dbpedia2:caption	"Promotional movie poster for the film"@en
dbpedia2:director	:Ridley_Scott
dbpedia2:producer	:Douglas_Wick
dbpedia2:writer	:David_Franzoni
dbpedia2:starring	:Russell_Crowe
dbpedia2:music	:Hans_Zimmer
dbpedia2:editing	:Pietro_Scalia
dbpedia2:runtime	"154 min. "@en
dbpedia2:language	:English_language
dbpedia2:distributor	:DreamWorks
dbpedia2:imdbId	"0172495"^^xsd:Integer
dbpedia2:amgId	"1:184587"@en
dbpedia2:reference	<http://www.americanrhetoric.com/moviespeeches.htm>
dbpedia2:budget	"103000000"^^dbpedia:units/Dollar
dbpedia2:abstract	":This article is about the 2000 film. For the 1992 film, see Gladiator (1992 film). Gladiator is a 2000 historical action drama film. It is directed by Ridley Scott and stars … "@en
dbpedia2:abstract	"Gladiator is een Amerikaanse film uit 2000 over de Romeinse tijd. De film werd geschreven door David Franzoni en geregisseerd door Ridley Scott…""@nl
skos:subject	:Category:2000_films
skos:subject	:Category:American_films
skos:subject	:Category:Drama_films
skos:subject	:Category:DreamWorks_films
skos:subject	:Category:English-language_films
skos:subject	:Category:Films_directed_by_Ridley_Scott
skos:subject	:Category:Universal_Pictures_films
foaf:page	<http://en.wikipedia.org/wiki/Gladiator_%282000_film%29>
dbpedia2:wikipage-nl	<http://nl.wikipedia.org/wiki/Gladiator_%28film%29>
dbpedia2:wordnet_type	<http://www.w3.org/2006/03/wn/wn20/instances/synset-movie-noun-1>

From the two excerpts it can seen that DBPedia goes far beyond categorisation and linking to other articles. Both projects try to approach the problems by extracting structured information from Wikipedia and by making this information available using Semantic Web technologies. DBpedia also allows to ask sophisticated queries against Wikipedia and to link other datasets on the Web to the DBpedia data.

The DBpedia and Wikipedia[3] datasets are available for download. Note that neither of the vocabularies is used in the final design.

4.2.4 TV-Anytime [Appendix B.5]

The TV-Anytime[20] forum is an organization established in 1999 to develop a specification of persistent audio and video or another kind of services on a consumer media platform. 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 [Appendix A.1] to represent the metadata. XML schema is used to represent the data model. A metadata schema is the formal definition

[20] http://www.tv-anytime.org

of the structure and type of metadata. TV-Anytime uses the MPEG-7 Description Definition Language (DDL) 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 [Appendix B.5.3].

In the context of TV-Anytime the metadata is descriptive data about content such as a programme title or synopsis. This metadata is called an attractor because on the basis of this information the end-user will decide whether or not to acquire a particular piece of content. TV-Anytime metadata is divided into two phases mainly Phase-1 and Phase-2.

TV-Anytime Phase-1 [Appendix B.5.1] is the set of specifications that enable various operations on the content from both broadcast and online media. Metadata specification addresses a data model that allows a broadcaster or consumer to describe the content available within the broadcast system. An example ("Gladiator") of the metadata is described below:

```
<tva:ProgramDescription>
    <tva:ProgramInformationTable>
      <tva:ProgramInformation programId="crid://tue.nl/Gladiator">
       <tva:BasicDescription>
         <tva:Title type="main"><![CDATA[Gladiator]]></tva:Title>
        <tva:Synopsis><![CDATA[ When a Roman general is betrayed and his…]]></tva:Synopsis>
        <tva:Genre href="urn:tva:metadata:cs:ContentCS:2002:3.4.6" type="main"></tva:Genre>
        <tva:Language><![CDATA[EN]]></tva:Language>
        <tva:CaptionLanguage closed="true" supplemental="false">nl</tva:CaptionLanguage>
        <tva:CreditsList>
          <tva:CreditsItem role="urn:mpeg:mpeg7:cs:RoleCS:2001:ACTOR">
           <tva:PersonName>
             <mpeg7:GivenName>Russell Crowe</mpeg7:GivenName>
           </tva:PersonName>
          </tva:CreditsItem>
          <tva:CreditsItem role="urn:mpeg:mpeg7:cs:RoleCS:2001:DIRECTOR">
           <tva:PersonName>
             <mpeg7:GivenName>Ridley Scott</mpeg7:GivenName>
           </tva:PersonName>
          </tva:CreditsItem>
        </tva:CreditsList>
        <tva:ProductionDate>
          <tva:TimePoint><![CDATA[2000]]></tva:TimePoint>
        </tva:ProductionDate>
        <tva:Duration>PT2H35M</tva:Duration>
      </tva:BasicDescription>
      <tva:AVAttributes><tva:VideoAttributes><tva:Color type="color"/></tva:VideoAttributes>
     </tva:AVAttributes>
    </tva:ProgramInformation>
   </tva:ProgramInformationTable>
  </tva:ProgramDescription>
```

TV-Anytime Phase-2 [Appendix B.5.2] is the extension of Phase-1. Few new concepts like Packages, Targeting (Automatically matching and delivering the content to the profiled consumer), and Re-distribution (sharing the content in and around the system) were added to Phase-2. Phase-2 specification is backward compatible with that of Phase-1 so all the elements in Phase-1 are still present in the Phase-2 but new elements were added.

4.2.5 OWL-Time [Appendix B.6]

In the RDF and OWL communities there are at least two different options to represent time. One way is to use XSD data types (e.g. xsd:datetime) and another one is to use OWL-Time[21].

To illustrate more clearly the difference between using OWL-Time and using the XSD data types, let's look at a concrete example: Suppose someone wants to watch a TV programme at 14:00 on 18 September 2007. But the little child wants also the watch "Telekids" at 10:00 on the same day, and the programme duration is 120 minutes (2 hours). Will there be an overlap? It can be expressed as follows:

[21] http://www.w3.org/TR/owl-time/

```
  <ex:programme1DateTime>
    2007-09-18T14:00:00+1:00
  </ex:programme1DateTime>
  <ex:programme2DateTime>
    2007-09-18T10:00:00+1:00
  </ex:programme2DateTime>
  <ex:programme2Duration>
    PT120M
  </ex:programme2Duration>
```

XSD data type only

```
<time:Interval rdf:ID="programme1DateTime">
  <time:begins rdf:resource="#p1Start"/>
</time:Interval>

<time:Interval rdf:ID="programme2DateTime">
  <time:begins rdf:resource="#p2Start" />
  <time:durationDescriptionOf rdf:resource="#p2DurationDescription" />
  <time:durationDescriptionDataType rdf:datatype="&xsd;duration">
    PT120M
  </time:durationDescriptionDataType>
</time:Interval>

<time:Instant rdf:ID="p1Start">
  <time:inCalendarClock rdf:resource="#p1StartDescription" />
  <time:inCalendarClockDataType rdf:datatype="&xsd;dateTime">
    2007-09-18T14:00:00+1:00
  </time:inCalendarClockDataType>
</time:Instant>

<time:Instant rdf:ID="p2Start">
  <time:inCalendarClock rdf:resource="#p2StartDescription" />
  <time:inCalendarClockDataType rdf:datatype="&xsd;dateTime">
    2007-09-18T10:00:00+1:00
  </time:inCalendarClockDataType>
</time:Instant>

<time:CalendarClockDescription  rdf:ID="p1StartDescription">
  <time:unitType rdf:resource="&time;unitMinute" />
  <time:year rdf:datatype="&xsd;gYear">2007</time:year>
  <time:month rdf:datatype="&xsd;gMonth">9</time:month>
  <time:day rdf:datatype="&xsd;gDay">18</time:day>
  <time:dayOfWeekField rdf:datatype="&xsd;nonNegativeInteger">3
  </time:dayOfWeekField>
  <time:dayOfYearField rdf:datatype="&xsd;nonNegativeInteger">261
  </time:dayOfYearField>
  <time:week rdf:datatype="&xsd;nonNegativeInteger">38</time:week>
  <time:hour rdf:datatype="&xsd;nonNegativeInteger">14</time:hour>
  <time:minute rdf:datatype="&xsd;nonNegativeInteger">0</time:minute>
  <time:timeZone rdf:resource="&tz-world;ATZ" />
</time:CalendarClockDescription>

<time:CalendarClockDescription  rdf:ID="p2StartDescription">
  <time:unitType rdf:resource="&time;unitMinute" />
  <time:year rdf:datatype="&xsd;gYear">2007</time:year>
  <time:month rdf:datatype="&xsd;gMonth">9</time:month>
  <time:day rdf:datatype="&xsd;gDay">18</time:day>
  <time:dayOfWeekField rdf:datatype="&xsd;nonNegativeInteger">3
  </time:dayOfWeekField>
  <time:dayOfYearField rdf:datatype="&xsd;nonNegativeInteger">261
  </time:dayOfYearField>
  <time:week rdf:datatype="&xsd;nonNegativeInteger">38</time:week>
  <time:hour rdf:datatype="&xsd;nonNegativeInteger">10</time:hour>
  <time:minute rdf:datatype="&xsd;nonNegativeInteger">0</time:minute>
  <time:timeZone rdf:resource="&tz-world;ATZ" />
</time:CalendarClockDescription>

<time:DurationDescription  rdf:ID="p2DurationDescription">
  <time:minutes rdf:datatype="&xsd;decimal">120</time:minutes>
</time:DurationDescription>
```

OWL-Time DateTimeDescription

Programme1 and programme2 are defined as intervals, *begins* is used for specifying the start times of the meetings. The datetimes are specified using *inCalendarClockDataType*. The duration of the meeting is specified using the duration description class.

From this example it can be seen that it is much simpler to describe date and time using the XML Schema data type. However, an advantage of using OWL-Time is that it can express more information than the XML data type only, such as "week", "day of week" and "day of year", so in the above example, it is also know that 18/09/2007 is Tuesday, on day 261 of the year, and in week 38 of the year. Moreover, each field is separate; this means that it is easier to extract the value of some fields for later use and easier to reason about. The unitType is added to define the precision; this date description leaves out the seconds, therefore the minutes are the most specific time unit.

Why OWL-time is preferred in such situation is that it facilitates a structure for a reasoner to determine whether if there is a conflict. Such a reasoner, can test and deduce the information. Only defining simple XML date types is not sufficient to reason about time.

An example of using OWL-Time in SenSee is to map a search string to a specific time. If a user enters “sport on Tuesday” the system must be able to map the string “Tuesday” to a time concept. The lack of interpretation constrains the search to a simple string matching between the string and the content metadata. Using the time ontology, the string “Tuesday” will be identified as a time concept. OWL-Time allows defining time concepts like “Tuesday”. What Tuesday describes is an interval of two time instances, starting at day 3 at 00:00:00 (begins) and ending at 23:59:59 (ends). With that knowledge and the separation of fields it is a lot easier to find all the items of that particular Tuesday. For example instead of defining “Tuesday”, it is also possible to define “evening”, “morning”, “weekend”, “Sunday Afternoon” etc. Important relations are, *before*, describing that “Monday” comes before “Tuesday” or “noon” before “afternoon” and *contains*, describing that “Weekend” contains “Saturday” and “Sunday”. These are also good examples of defining concepts to reason about. Items broadcasted on “Sunday afternoon, are broadcasted on “Sunday“ and therefore also in the “Weekend”.

The conclusion is that OWL-Time can define complex date and time representations and enables reasoning by deducting information.

4.3 Extended common scheme

In the previous paragraphs a number of sources have been described to provide the framework with metadata (for broadcasts, VOD and the enrichment of them). From these a number of sources have been selected. These sources differ in semantics, layout and formatting. To achieve content retrieval and integration, a data description is required that allows TV-content to be described, linked and associated with other available data sets. A key element in SenSee is the use of Semantic Web languages RDF(S)/OWL to model all input data and its relationships, to be able to reason and navigate over it in a uniform connected way. It can be the case that a common scheme lacks a little bit of power (expressiveness) in particular cases; for example to apply reasoning over a number of subjects, like time, geography and genre. Therefore a number of vocabularies (e.g. ontologies) are selected to enrich, provide structures/hierarchies and to apply reasoning.

Thus the process is done in two steps:

1. Identify sources and vocabularies
2. Define mappings

4.3.1 Sources to be integrated

To integrate the different sources a common scheme is necessary to be able to create a uniform representation. It must allow TV-content to be described, linked and associated with other available data sets. Before the start of the thesis, during the architectural design it was decided to use the TV-Anytime specification as main domain model to describe the content internally.

For the integration of next-generation TV-content on the Web the translation of the TV-Anytime was crucial [4]. It was necessary in order to share the common structure easily with other applications. Therefore the original TV-Anytime classification files and the TV-Anytime XML Schema, which expresses all the data types and semantics have been transformed into OWL/RDF. Also SKOS[22] was used to describe relations between concepts in the loosely structured vocabulary (i.e. by adding broader/narrower/related relations).

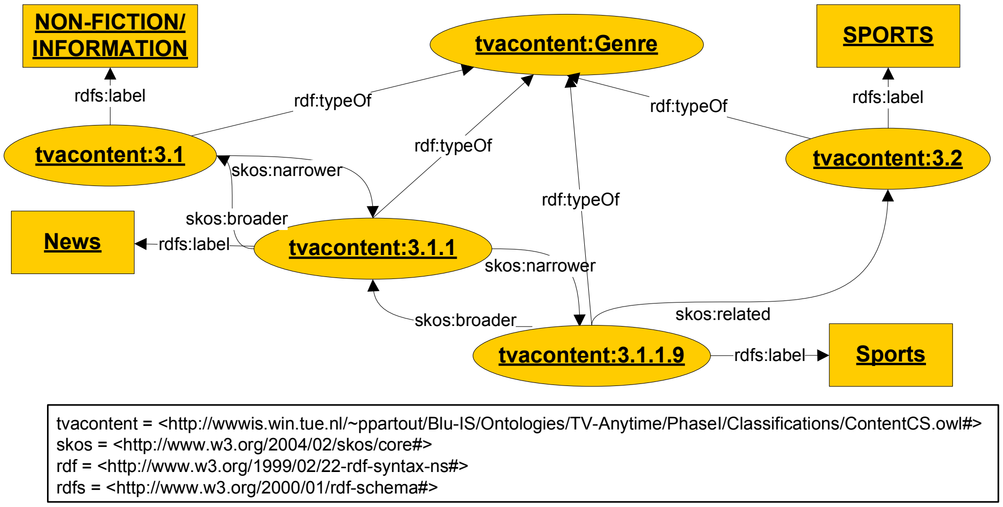


Figure 4.1: TV-Anytime Genre classification

In Figure 4.1 a small part of the transformed content (genre) classification file is depicted. SKOS was added to define relations between the genres. In the original scheme sports news and sports are not related at all, by adding *skos:related*, the two genres are related. For example, for the *tvacontent:3.2* (Sports) concept also *tvacontent:3.1.1.9* (Sports News) can be found. The *skos:broader* results in all genres that are more general than the current genre and *skos:narrower* in more specific genres.

Furthermore the TV-Anytime time definition was mapped to the OWL-TIME notion of time. OWL-Time was extended with a set of instances like 'afternoon', 'Wednesday' and 'weekend'; This to enable a user to enter "sport on Wednesday" resulting in items related to the Wednesday in the current week. The last addition is a Geographical vocabulary, to enable reasoning over locations.

When adding new sources with content that is not TV-Anytime compliant, mappings must be defined to obtain a consistent TV-Anytime data set. However, first a number of sources must be selected.

4.3.1.1 Source selection

For the broadcasts concept two metadata sources have been selected:

- XMLTV
- BBC Backstage

XMLTV can deliver programme information from several programme information and TV-guide websites worldwide. It is a good source to provide the system with metadata for the broadcast concept. It returns a XML document and it can easily be transformed to the adjusted TV-Anytime format.

BBC Backstage is an alternative source, which delivers original TV-Anytime content. This means that it just needs a small transformation to the adjusted TV-Anytime format. Thereby the advantage of having a richly described metadata source, in terms of how many fields are filled. Also it is good addition to show that integration of different sources, with different semantics, layout and formatting works.

[22] http://www.w3.org/2004/02/skos/

IMDb is the best option to mimic Video on demand. It contains a large and rich set of movie, series and related material. Many metadata fields are involved, which means that the items are extensively described. Furthermore it supports the possibility to access the content for non-commercial purposes. The interface to access and retrieve the metadata is quite acceptable. Last but not least is the big support of the community for this website/metadata. Alternatively the source can also be used to enrich the metadata from the sources selected for the broadcast concept, in particular the movie items. The IMDb metadata is much richer than both broadcast sources, in number of metadata fields.

The TV-Anytime specification is a good candidate to model the IMDb movie metadata. However the IMDb (movie, person) metadata contains fields than not exist in the TV-Anytime specification. For example the person metadata fields; TV-anytime only allows storing person names and their roles in the credit list. There is no field to store biographically information about a person available.

A solution is to use or create an IMDb scheme which can be mapped to the TV-anytime format.
First the possibility to use an existing IMDb scheme was investigated.
Three schemes have been found, namely:

- IMDb Mapping Movie Ontology[23]
  This schema is rather simple and small. A lot of metadata cannot be described (e.g. filming locations) or can only be modelled as a string. For example, date and time atrtibutes should be modelled according to a date time type. For movie genres, there is no structure involved to describe a genre classification. Also there is no distinction made between movies and series and their episodes. If the IMDb data is investigated carefully, it is clear that movies and series contain the same attributes to describe the data. However, episodes are a little bit richer; some additional attributes (season, number and airdate) are involved. Also series refer to their episodes and this episode data can differ from the series (e.g. other/more actors involved). To model this, an episode should inherit the attributes from a more general movie/series class. Besides that a series object should also refer to its episode objects.
- RDFSuite IMDb scheme[24]
  This scheme is already much better modelled in comparison with the first one. Data types are used how they should be used. The scheme contains a very extensive structure of roles a person can fulfil. However other parts of the scheme are simply modelled and also some metadata cannot be described. For example, filming locations are missing. There is no distinction made between movies and series and their episodes.
- UMDB IMDb Vocabulary Specification[25]
  This scheme is the best of the three. The creators studied the IMDb text dump and the website of IMDb in order to describe the data. Still, attributes are missing. The filming locations are modelled, but not via a hierarchical scheme. Also they assume United States locations, because the attributes involved describing the locations are modelled like US standards. From the data it is clear, that IMDb locations are much richer.
  Again there is no distinction between movies and series and their episodes.

Concluding that none of the three schemes really fitted, the last one was the best option of the three. All schemes provided in the fundamental concepts, but omit some useful attributes or model it differently. Therefore the choice was made to create a new extensive scheme which models time, locations, genres, series-episode relations better. Additionally relations between movies and between persons (e.g. father of, son of) are modelled. The choice of how to model the fields is completely free. For example to define a genre, the IMDb genre can be mapped to a TV-Anytime genre classification, instead of storing the plain keyword. However for the most metadata fields a plain text string suffices, think about a synopsis or title, or an integer, e.g. the episode number.

[23] http://www.schemaweb.info/schema/SchemaInfo.aspx?id=284
[24] http://139.91.183.30:9090/RDF/VRP/Examples/imdb.rdf
[25] http://ebiquity.umbc.edu/ontologies/imdb0.9/imdb.rdf

During this phase also the necessary vocabularies, concepts and supporting sources have been chosen.

- OWL-Time: for time conceptualizations extended with a set of instances like 'afternoon', 'Wednesday' and 'weekend'. It enables reasoning with time concepts.
- TV-Anytime content (genre) classification is a direct result of choosing TV-Anytime; enabling an extensive and hierarchical set of TV content genres.
- Geographical: IMDb locations to provide a hierarchical set of geographically locations
- IMDb photos, provider of photos of movie and movie related material.
- Videodetective as provider of movie trailers. Note that movie-list was the first choice, because the Videodetective service started during the design. However, this change does not make a difference, both sources provide trailer URLs.
- Literature

Wordnet is considered to extend movie keywords or possible other string related attributes. In SenSee this extension is applied at query time (if the user enters a keyword), in the query refinement. The query keywords are extended with synonyms. However it is also possible to extend the keywords from the metadata sources at the retrieval of the content. The first mentioned is preferred; it is better to keep the original keywords and to apply to extension at the query refinement. As consequence, less storage and the moment of choice using WordNet can be postponed (in the refinement). Thus, WordNet is only used to extend keywords at query time.

In the case of the geographically vocabulary three options have been introduced. Namely, TGN, Geonames and the original location tree of IMDb. The IMDb hierarchy is preferred because it is a relative easy step to get a hierarchy without transformations and without the loss of data. To illustrate this, take the example: `Grote Kerkplein, Rotterdam, Zuid-Holland, Netherlands`. *Netherlands* is the most general term and *Grote Kerkplein* the most specific. Each term defines a level in the location tree. If TGN or Geonames are used, the IMDb location string must be mapped to a location in TGN or Geonames. The issue hereby is that the IMDb location string is used as input and that the output relies on that string; a one to one transformation from IMDb locations to Geonames or TGN concepts is not present/possible. Some intermediate steps are necessary to identify the location. The input string "Malta" is not a problem, each vocabulary, can identify the country name. For more specific locations it starts to be a problem. *Grote Kerkplein* cannot be found, neither of the two is able to return a result. A possible solution is to accept a more general term, in this case, Rotterdam. However some data is lost. Another issue to take care of the uniqueness of locations; for example there exists a city called Ontario in Canada but also Ontario in California-USA exists. The difference between those two must be preserved to guarantee uniqueness.

The locations in IMDb are very detailed (sometimes on street/house level) and it focuses on locations in the domain of movie content. Movies shot in a dynamically environment, like (Marine) ships, are also defined. TGN focuses too much on important art and architectural locations, thus locations without any interesting art or architectural objects are not present in the repository. Geonames is a general repository, also returning small villages in the world. On the other hand Geonames will cause a lot of overhead. The set contains 6.5 million items. IMDb uses just a small fragment of the set (around 50.000 unique locations). This could be very unsuitable by building a location structure in the interface.

Thus the advantage of using the IMDb tree is that all available locations are related to items in the IMDb data set (no overhead) and secondly there is not any transformation needed to another set. From the IMDb locations a straightforward ontology is built, on basis of the hierarchy it provides.

As mentioned before, IMDb also provides literature about movie and movie related information (e.g. persons). Because the difficulties of unravelling the literature metadata the idea of matching a vocabulary was introduced. ISBNdb and WorldCat are two possible candidates. For future usage it is recommended to investigate these candidates profoundly. During the design it was decided to postpone it, because it did not belong to the key features of the system. The IMDb scheme provides some basic metadata about literature (e.g. ISBN).

For the integration of Wikipedia, DBPedia is the best option. The set of Wikipedia³ contains links between Wikipedia articles only. For example if there is a link to the actor Russell Crowe in the movie description of The Gladiator then Wikipedia³ collects this link. DBPedia does something similar but also collects other data, such as the introduction of the article in multiple languages. DBPedia is much richer, it goes far beyond categorisation and linking to other articles like Wikipedia³ does, therefore more useful to implement in SenSee. Unfortunately, the scheme of DBPedia was not available during the thesis. The investigation was done on basis of the RDF dataset. The conclusion of the investigation on basis of the RDF dataset is well-considered and correct, but for implementation/integration purposes the complete scheme should be available. The scheme will be available in the future. Anyhow the Wikipedia integration has been postponed.

4.3.2 Integration process

After the identification, the mix of source schemes, vocabularies and ontologies have been shaped in an integrated common scheme. The central vocabularies used in SenSee are: the TV-Anytime [Appendix B.5.4] (including MPEG-7) ontology for description of incoming content; the OWL-Time [Appendix B.6] for time conceptualizations extended with a set of instances like 'afternoon', 'Wednesday' and 'weekend'. WordNet [Appendix B.8] to provide synonyms, hyponyms and some other lexical relations. Furthermore a geo vocabulary, based on IMDb locations [Appendix B.7.1] is added to enable reasoning about locations.

A set of heterogeneous (publicly) available metadata sources which serve as the searchable content have been chosen. If metadata is non TV-Anytime compliant, transformations are performed to obtain a consistent TV-Anytime data set. The sources are:

- BBC Backstage: metadata in the TV-Anytime format
- XMLTV: manage and retrieve TV listings stored in the XMLTV format
- IMDb: provider of the metadata mimicking the VOD concept, photos and location hierarchy
- Videodetective: provider of movie trailers
- Movie-list: alternative provider of movie trailers.

Figure 4.2 depicts a general overview of vocabularies and data sources used; giving a general idea of the mappings which will be described later on. Note that Wikipedia (DBPedia) and literature have been postponed.

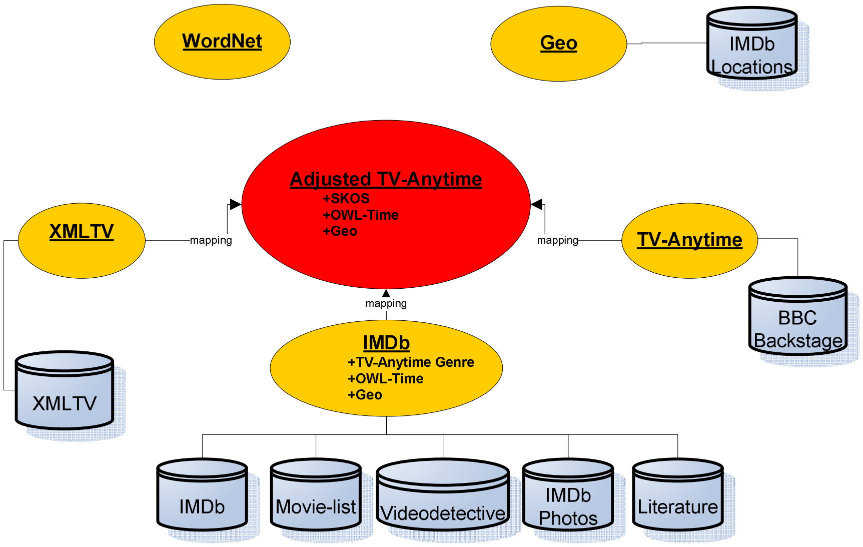


Figure 4.2: General overview

The central role in the content integration is an adjusted TV-Anytime ontology. SKOS is used to describe relations between concepts in the loosely structured term hierarchies (i.e. describing genres, formats, roles etc.) by adding broader/narrower/related relations. The TV-Anytime time definitions are mapped to the

OWL-TIME notion of time. OWL-Time was extended with a set of instances like 'afternoon', 'Wednesday' and 'weekend'; The last change is to map the geographical definition to a Geo vocabulary, to enable reasoning over locations. This vocabulary is created using the IMDb location tree.

Programme information from BBC broadcasts is retrieved directly from BBC Backstage in (original) TV-Anytime format. This information must be mapped to the adjusted version of TV-Anytime. Another broadcast source is XMLTV, this XML based document [Appendix B.3.2] must also be mapped to TV-Anytime.

The IMDb metadata is modelled according to the created IMDb scheme [Appendix B.1.4]. This scheme uses some vocabularies to enable the same kind of reasoning and to find content easily for end-users. Attributes like genre, time and location lend themselves to describe them in a structure/hierarchy. Therefore IMDb includes the TV-Anytime genre classification, time attributes map to OWL-time and locations are modelled according to the IMDb location scheme. During the creation of the IMDb scheme, a choice was to map the attributes at query time or to link them directly with these vocabularies at retrieval time. The big advantage of doing it at retrieval time is a faster performance at query time, no additional mappings are necessary. Because the choice was completely free, it did not depend on an already existing IMDb scheme, better performance was preferred.

The IMDb scheme also provides the integration of trailers, photos and literature. A mapping from IMDb to TV-Anytime enables SenSee to integrate it with the metadata from the other sources.

WordNet is separated from the rest, because it extends the keywords at query time and does not directly operate on the metadata set.

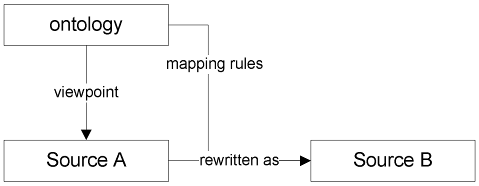


Figure 4.3: Mapping procedure

In general, the ontology and the mapping enables the translation from one scheme (source a) to another (source b). Thus, all the different sources with a variety of schemes are mapped to the adjusted TV-anytime specification, to create a coherent set of metadata; regular broadcasts (BBC, XMLTV) and Video on-demand (IMDb) are merged into one set.

This general overview gives already a good idea of how the different sources and schemes are related, however the mappings from one scheme to another makes the integration possible. These mappings will be discussed now. The mappings cannot be presented in on scheme, because it would not fit on one page. The schemes on which these mappings are applied can be found in the appendices:

- TV-Anytime and BBC backstage [Appendix B.5.4]
- XMLTV [Appendix B.3.2]
- IMDb [Appendix B.1.4]
- Geo (IMDb locations) [Appendix B.7.1]
- OWL-Time [Appendix B.6]

The difference between broadcast and on demand programmes in TV-Anytime is made via the ProgramLocation. A broadcast programme is an instance of *tva:BroadCastEventType*, such a broadcast type is a subclass of *tva:ScheduleEventType,* which is a subclass of *tva:ProgramLocationType*. A on-demand programme is an instance of *tva:OnDemandProgramType*. This type is also a subclass of *tva:ProgramLocationType*. Subclassing means that the subclass inherits all the properties from the superclass.

In practice this can mean that one programme, with one programmeInformation instance, which describes the informational metadata of the programme, can have more locations, it can be available on-demand but also via the broadcast.

One of the most applied mappings is the mapping from a xsd:dateTime (or something similar) to an OWL-Time Instant (Figure 4.4). The dateTime is mapped to an OWL-Time Instant. The separate fields in OWL-Time are filled with the individual date components in xsd:dateTime. Thus, 2007 is a year within OWL-Time. This mapping can also be applied for other date time definitions used in IMDb for example.

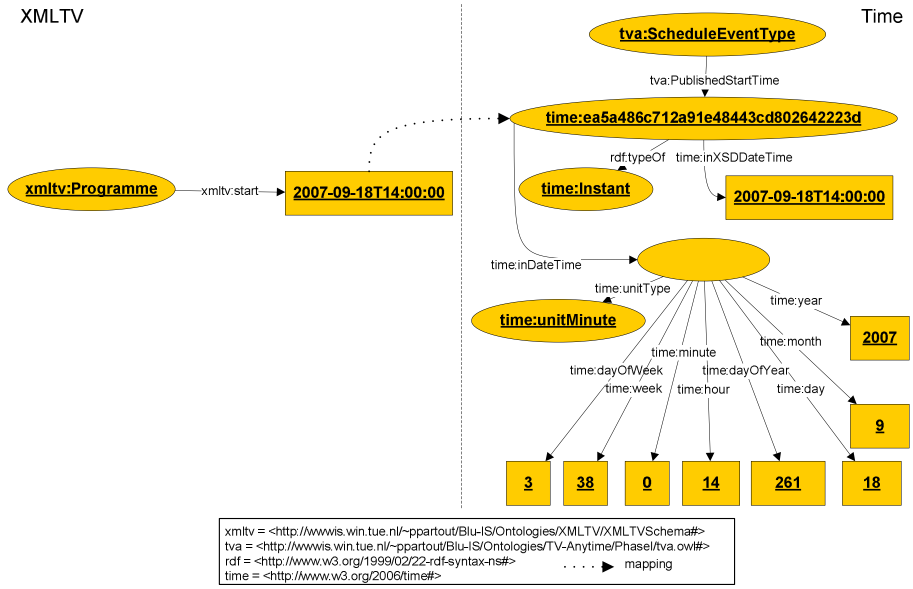


Figure 4.4: Mapping a xsd:dateTime to OWL-Time

4.3.2.1 XMLTV

For XMLTV an extensive mapping scheme is created to map the content from one scheme, to another, TV-Anytime. The most important mappings will be discussed here, but are all depicted in Figure 4.5.

Date-time attributes are mapped to OWL-Time and geographical related attributes are mapped to IMDb location. The only geographical property in XMLTV is the country. Thus country is mapped to an instance of a Location in the Geo vocabulary. In general, the XMLTV *Programme* attributes are mapped to the attributes of *ProgramInformationType*, except some attributes such as channel, start and end time. These attributes are part of a *BroadcastEventType*. The attributes are inherited from the superclasses.

Another mapping is the *xmltv:category* property (genre) to the TV-Anytime GenreType. The GenreType refers to a content classification containing terminology like the genres Sports, Action and Horror. How the (adjusted) structure looks like is depicted in Figure 4.1. What happens is that the XMLTV category *action* is mapped to *tvacontent:3.4.6* in TV-Anytime. This is the Action genre within the TV-Anytime genre classification. An excerpt of the mapping is shown below.

```
tvacontent = http://wwwis.win.tue.nl/~ppartout/Blu-IS/Ontologies/TV-Anytime/PhaseI/Classifications/ContentCS.owl#
```

XMLTV category	TV-Anytime Genre	XMLTV category	TV-Anytime Genre
action	tvacontent:3.4.6	sci-fi	tvacontent:3.4.6.7
adventure	tvacontent:3.4.6.1	short	tvacontent:3.4.17

Table 4.1: XMLTV category TV-Anytime Genre mapping

The Credits are another mapping implemented. The various properties of *xmltv:Credits* are actor, director etc. These properties contain a value: the name of the person, who fulfils the role. The value of the property is mapped to *tva:FullName*, which is connected via *tva:PersonName*, a property of an instance of *tva:CreditsItemType*. The property (actor, director) itself is mapped to a *tva:termReferenceType* of the property *tva:role*. For example, the property *xmltv:actor* in XMLTV is mapped to the term ACTOR in the TV-Anytime role classification. In Figure 4.5 this is not depicted clearly. Figure 4.6 gives a more detailed overview, by using an example.

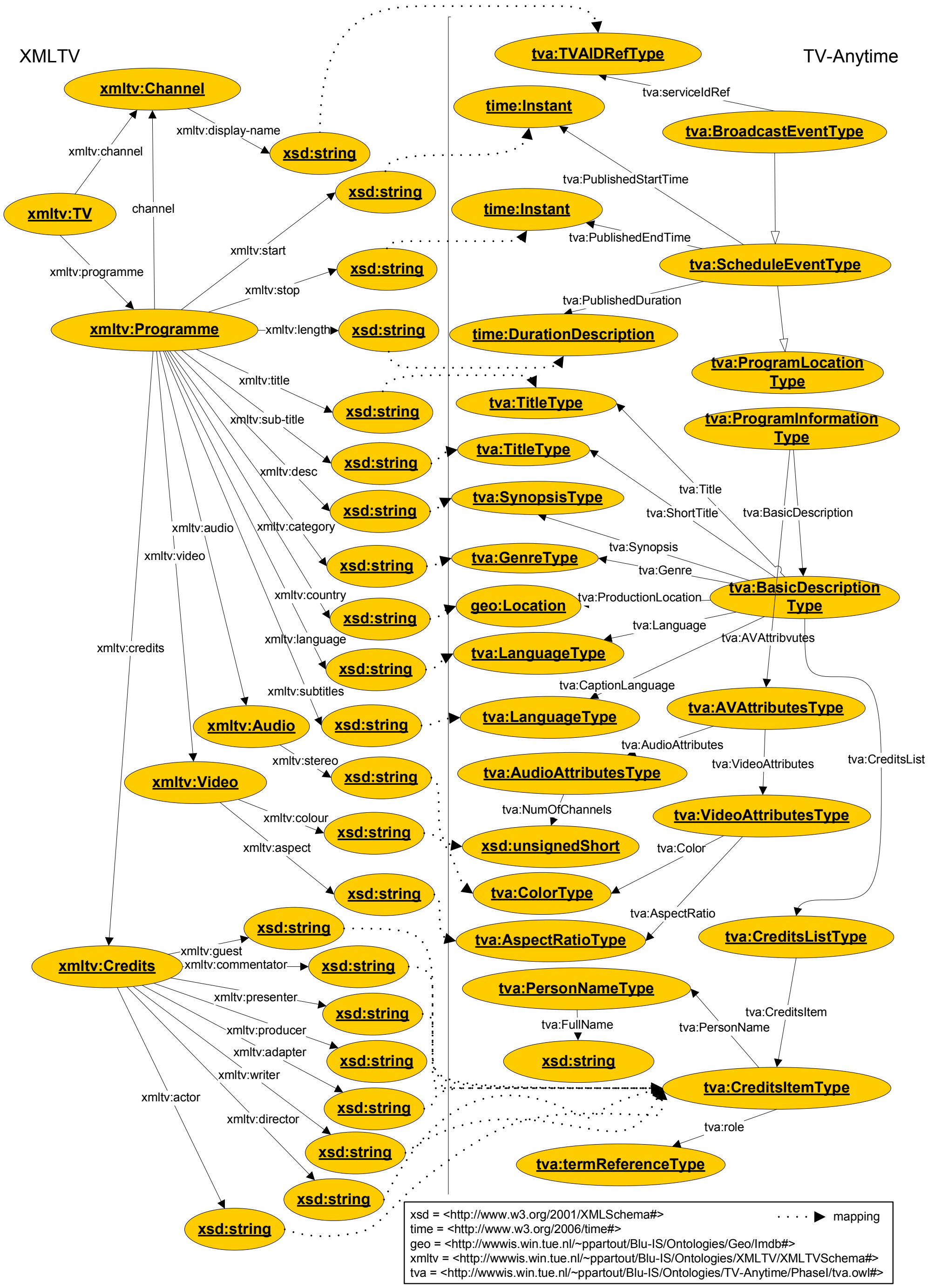


Figure 4.5: Mapping XMLTV to TV-Anytime

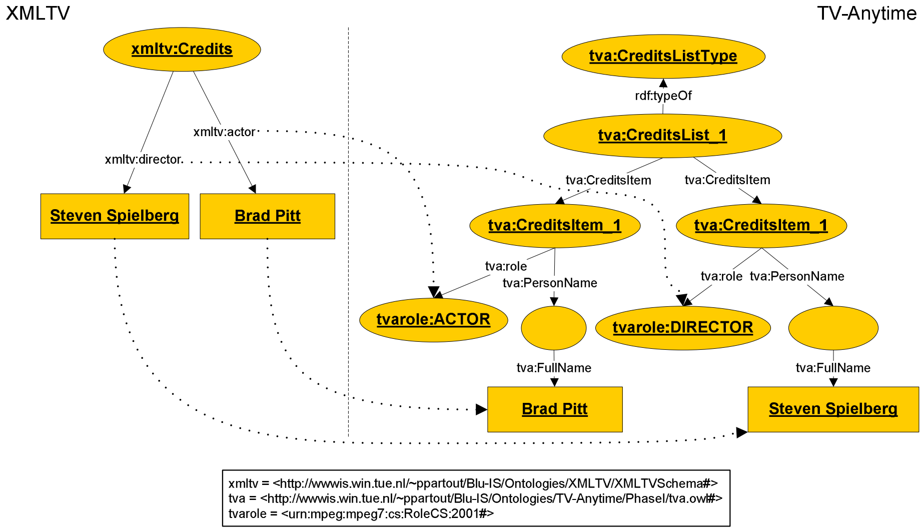


Figure 4.6: Mapping XMLTV credits to TV-Anytime CreditsList

4.3.2.2 BBC Backstage

The BBC set contains a number of files for each channel per day. The *ProgramLocation* and the P*rogramInformation* files provide the searchable content. The data in these files are already TV-Anytime formatted. Due to the adjusted model of TV-Anytime used in the project, some mappings are necessary.
Time related attributes are mapped to OWL-Time, an example is depicted in Figure 4.4. Furthermore geographical related attributes are mapped to the IMDb location scheme. These adjustments correspond with the (blue) dark coloured nodes in the TV-Anytime scheme [Appendix B.5.4].

In the *ProgramInformationTable* the following property is mapped:

- ProductionLocation (RegionCode to IMDb location)

In the *ProgramLocationTable* the following attributes are mapped:

- PublishedStartTime (XSD:dateTime to OWL-Time, Instant)
- PublishedEndTime (XSD:dateTime to OWL-Time, Instant)
- PublishedDuration (XSD:duration to OWL-Time, DurationDescription)

4.3.2.3 IMDb

The IMDb scheme [Appendix B.1.4] was created after studying many web pages describing movies on the IMDb website and studying the text dump of the database. In this thesis it was attempted to design a detailed ontology for IMDb. The scheme provides the most relevant information from IMDb; it describes the fundamental concepts involved in IMDb, including classes such as movies, persons, actors, directors, genres, languages, countries, keywords, etc., and the relations among them. Also the episode, series relation and subclasses are modelled like they should be. This scheme uses some vocabularies to apply reasoning or to present content in a logical way, like explained earlier.

Important integrations:

- IMDb and trailers (video detective)
  Trailers are integrated in the scheme of IMDb. First of all IMDb provides all movie and series items available. The trailer sources, like Videodetective or Movie-list do not contain items which are not included in the IMDb set. The Videodetective cannot be copied completely, because the interface does not allow this (Paragraph 4.1.3.1). Also it is often possible that such a trailer source does not contain metadata, but that it has been found by a redirection from IMDb. Therefore IMDb is the main driver of collecting the trailers; the IMDb metadata is used to query the Videodetective interface and to check if the correct movie trailers are returned.
- IMDb and photos

Important mappings at retrieval time involved:

- IMDb and OWL Time

  All date-time attributes are mapped to OWL-Time. The attributes in the dataset of IMDb can be categorized in three sets: year, date and duration.

  Set of year attributes:

  - releaseBeginYear
  - releaseEndYear
  - distributionYear

  Set of date attributes:

  - productionStartDate
  - productionEndDate
  - filmingStartDate
  - filmingEndDate
  - hasReleaseDate
  - airdate
  - startDate
  - endDate
  - dateOfDeath
  - dateOfBirth

  Set of duration attributes:

  - versionDuration

  The year attributes defined in the IMDb set look like “2000” or “????” (year is not known) or are not defined. The undefined or unknown years are omitted. The defined year attributes are mapped to an OWL-Time *Instant*. The date attributes are also mapped to an OWL-Time *Instant*. Such an Instant is described like the example below. For a year only the year and the unitType is described. A duration attribute is mapped to an OWL-Time *DurationDescription*.

```
<time:DateTimeDescription rdf:ID="instantExampleDescription">
  <time:unitType rdf:resource="&time;unitDay" />
  <time:year rdf:datatype="&xsd;gYear">2007</time:year>
  <time:month rdf:datatype="&xsd;gMonth">9</time:month>
  <time:week rdf:datatype="&xsd;nonNegativeInteger">38</time:week>
  <time:day rdf:datatype="&xsd;gDay">18</time:day>
  <time:dayOfWeekField rdf:datatype="&xsd;nonNegativeInteger">3</time:dayOfWeekField>
  <time:dayOfYearField rdf:datatype="&xsd;nonNegativeInteger">261</time:dayOfYearField>
</time: DateTimeDescription>
```

- IMDb and IMDb locations

  IMDb distinguishes countries (Netherlands, USA), locations (Eindhoven, Noord-Brabant, Netherlands) and country abbreviations described in ISO-3166 (NL, UK).

  The following attributes are defined as a country string:

  - countryTitle
  - countryOfOrigin
  - countryDuration
  - countryRating
  - countryOfRelease

  The following attributes are hierarchical location strings:

  - filmingLocation
  - deathPlace
  - birthPlace

  The property companyLocation defined as ISO-3166 country abbreviation is mapped to the full country name. This because to be consistent with the location structure.

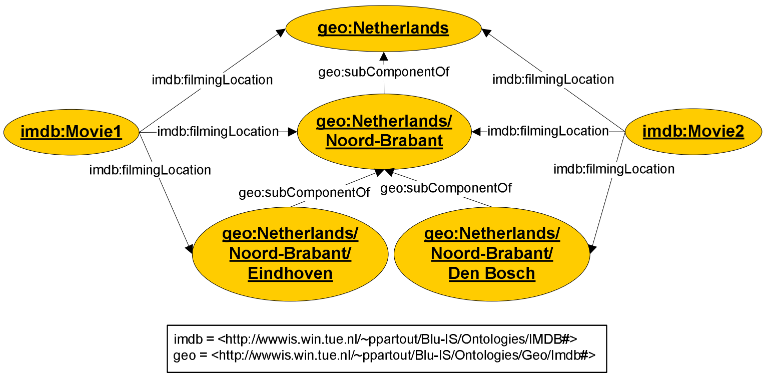

geo:Netherlands
imdb:filmingLocation
geo:subComponentOf
imdb:filmingLocation
imdb:Movie1
imdb:filmingLocation
geo:Netherlands/ Noord-Brabant
imdb:filmingLocation
imdb:Movie2
imdb:filmingLocation
imdb:filmingLocation
geo:subComponentOf
geo:subComponentOf
geo:Netherlands/ Noord-Brabant/ Eindhoven
geo:Netherlands/ Noord-Brabant/ Den Bosch
imdb = <http://wwwis.win.tue.nl/~ppartout/Blu-IS/Ontologies/IMDB#>
geo = <http://wwwis.win.tue.nl/~ppartout/Blu-IS/Ontologies/Geo/Imdb#>

Figure 4.7: IMDb location hierarchy

Figure 4.7 depicts the items "Eindhoven, Noord-Brabant, Netherlands" and "Den Bosch, Noord-Brabant, Netherlands". Location attributes in the dataset defined like this are related to the Netherlands, Noord-Brabant and Eindhoven/Den Bosch. If a user asks for all movies shot in the Netherlands, both Movie1 and Movie2 show up. A country string, "Netherlands" will only map to one node in the tree.

- IMDb and TV-Anytime Content classification
  In the IMDb data set a list of genre items per movie can be found. This set is just a collection of strings. The strings can be mapped to a term specified in the TV-Anytime genre classification scheme. All the genre keywords of IMDb are mapped to a Genre in TV-Anytime.

```
tvacontent = http://wwwis.win.tue.nl/~ppartout/Blu-IS/Ontologies/TV-Anytime/PhaseI/Classifications/ContentCS.owl#
```

IMDb genre	**TV-Anytime Genre**	**IMDb genre**	**TV-Anytime Genre**
action	tvacontent:3.4.6	film-noir	tvacontent:3.4.13
adult	tvacontent:3.9	horror	tvacontent:3.4.6.6

Table 4.2: Excerpt of IMDb – TV-Anytime genre mapping

The mappings during query time are discussed next. In the IMDb mapping scheme (in Figure 4.9) the mappings are quite identical. An IMDb movieTitle maps to a TitleType, a plot to a synopsis. The only mapping which is less obvious is the role. A role has a relation with a unique Person, this Person has a *fullName*. This *fullName* maps to *FullName* in TV-Anytime. The role itself is mapped from a role instance to a *termReferenceType* in TV-Anytime. Thus a imdb:Actor is mapped to ACTOR in the TV-Anytime role classification. The character an actor can play is defined in imdb:playsCharacter and is mapped to the tva:FullName of tva:Character. This is depicted in Figure 4.8.

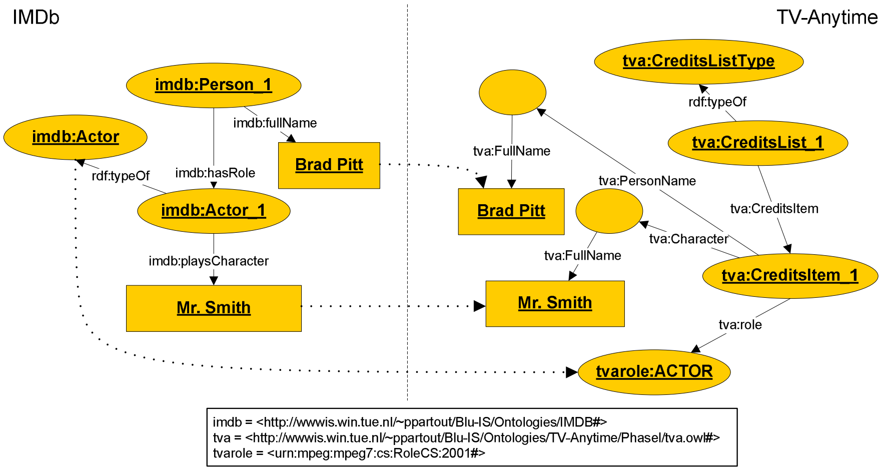


Figure 4.8: Mapping IMDb Actor to TV-Anytime Credits

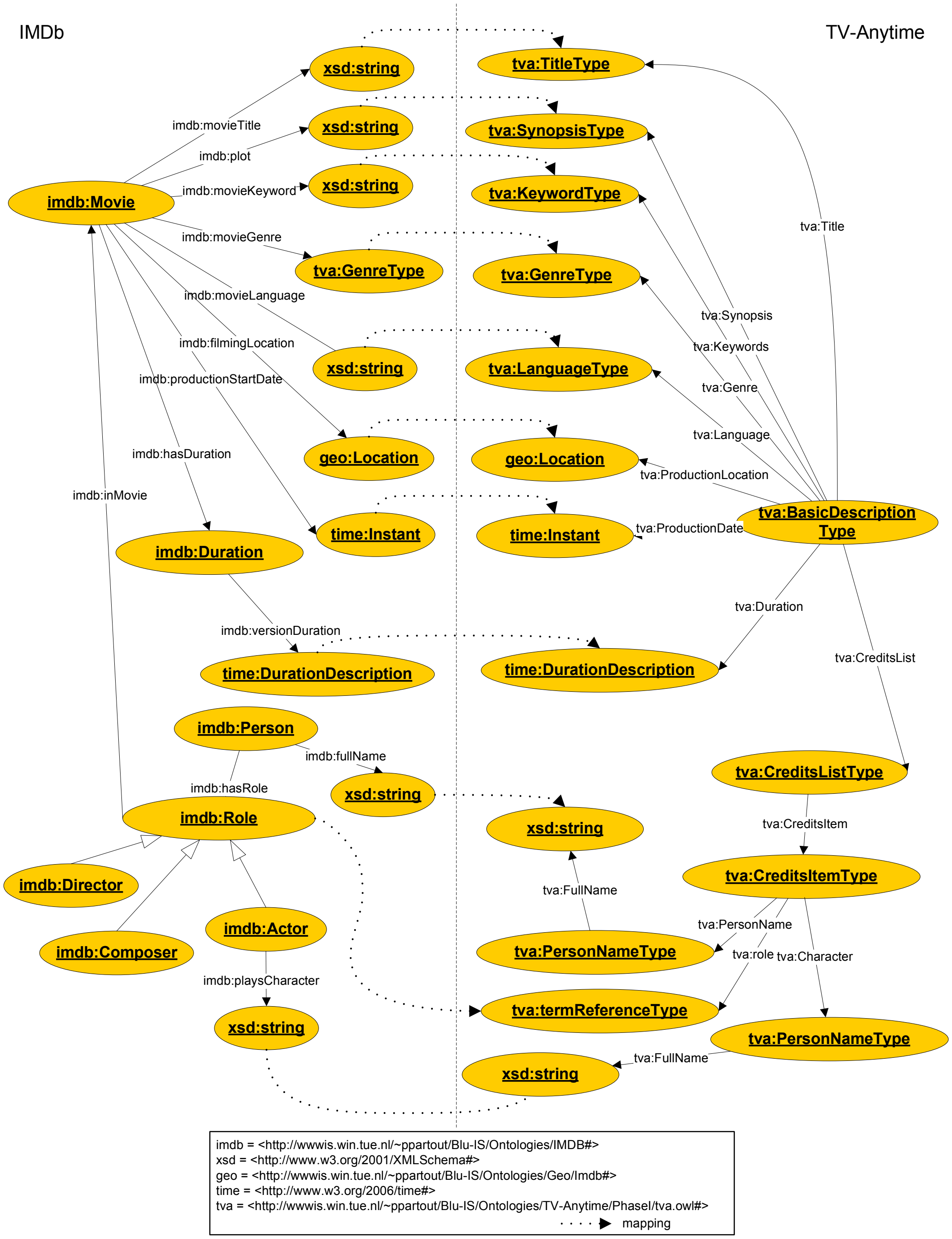


Figure 4.9: Mapping IMDb to TV-Anytime (1)

In the next scheme (Figure 4.10), the mapping from audio/visual properties is depicted. An *imdb:movieColor* (e.g. colour or black and white) in XMLTV maps to a *tva:ColorType* in TV-Anytime and an *imdb:soundMix* (e.g. dolby) to a *tva:MixType*.

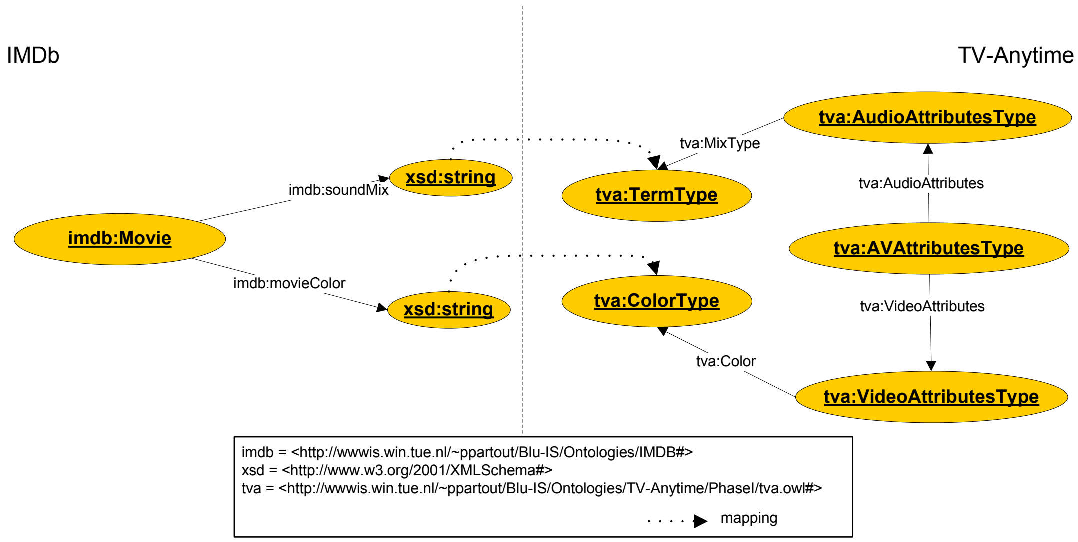


Figure 4.10: Mapping IMDb to TV-Anytime (2)

Each programme in TV-Anytime (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). The CRIDType defines how an identifier within TV-Anytime must be described. The unique id of a Movie (namespace and hash of movie title) used in IMDb is mapped to the CRID (Figure 4.11).

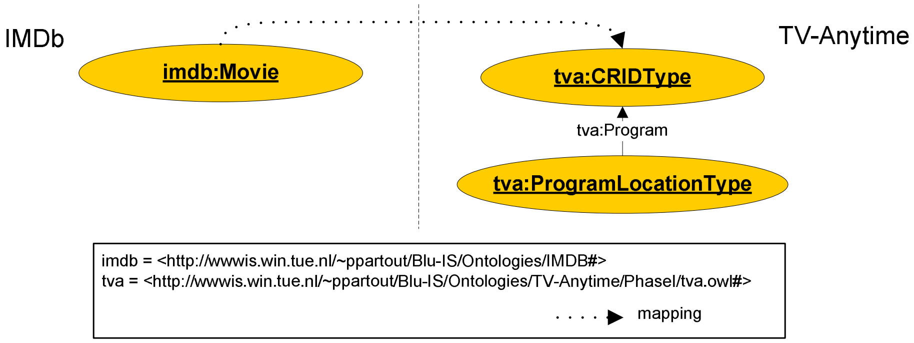

IMDb
TV-Anytime
imdb:Movie
tva:CRIDType
tva:Program
tva:ProgramLocationType
imdb = <http://wwwis.win.tue.nl/~ppartout/Blu-IS/Ontologies/IMDB#>
tva = <http://wwwis.win.tue.nl/~ppartout/Blu-IS/Ontologies/TV-Anytime/PhaseI/tva.owl#>
mapping

Figure 4.11: Mapping IMDb to TV-Anytime (3)

4.3.3 Conclusion

Research question RQ1 is answered. In the previous paragraphs it was explained what different sources, are available and how they differ from each other. XMLTV and BBC Backstage are chosen as metadata source for broadcasts and IMDb is the rich metadata source mimicking the working of the VOD concept.
For IMDb a scheme was created on basis of the original IMDb dataset. The scheme uses some vocabularies to enable reasoning and categorisation. Attributes like genre, time and location lend themselves to describe them in a structure/hierarchy. XMLTV metadata is described in XML and BBC in TV-Anytime.

The sources and schemas have been integrated to be able to reason and navigate over it in a uniform connected way. The mapping to TV-Anytime enables this uniformity. OWL-Time, WordNet and IMDb locations are vocabularies enabling reasoning about time, extending keywords and reasoning about locations.

The integration is complete, now the end-user must be able to access all the content.

4.4 Presentation & Faceted browsing: Client aspects

Bringing together TV content, available from high number of broadcasting channels, Video on demand and a number of shared vocabularies for its representation, requires a special attention of the search and browser interface presented to the user. In the metadata a lot of hierarchies/structures are involved. To present such kind of content an effective way of presenting and navigating through these hierarchies and grouping of content has been investigated. In this paragraph research question RQ3 is answered.

The result of the SenSee framework will be realised in two applications for a personalized programme TV guide; a Web-based and a set top-box-based (TV) [26]. In this thesis the Web-based version is implemented. This client, called iFanzy, is implemented on top of the SenSee framework.

If an end-user is searching for content the client application generates queries retrieving the content. These queries act on the integrated set. These queries can be refined or not, that is not of interest for this part. The result of the query is in TV-Anytime format. A user is often interested in search by keywords (free-text) or by facets. Each facet typically corresponds to the possible values of a property. For example, a title, genre, location, length, director, actors and other uniform properties that can be turned into facets to filter on. A user interface which makes use of faceted classification allows the user to explore by filtering available information. The most prominent use of faceted classification is in faceted navigation systems that enable a user to navigate information hierarchically, going from a category to its sub-categories, but choosing the order in which the categories are presented. This contrasts with hierarchies of categories being fixed and unchanging. For example, a traditional TV guide might group programmes first by time, then by genre, and location. In a faceted system, a user might decide first to divide the programmes by genre, and then by location and then by time, while another user could first sort the programmes by locations and then by time. Thus, faceted navigation, guides users by showing them available categories (or facets), but does not require them to browse through a hierarchy that may not precisely suit their needs or way of thinking.

Therefore two ways to access and navigate TV content have been introduced, by search and by facet. Both ways of navigation can also be combined, but both are not mandatory. Faceted-based presentation of search results is a good demonstration of combining multiple perspectives in one view. Besides that the set of metadata is multidimensional and facet based browsing is very appropriate on such data sets.

This two way navigation is also implemented for example in Apple iTunes; it has also faceted browsing features. When the browser is activated, the user can choose a genre and also filter music files by a search term: this combines the genre facet with the search field to narrow the results.

In iFanzy, genre, time and location have been selected as the three main facets. The choice for these facets is very straightforward; it is less convenient to represent metadata fields like title and synopsis as facets, because they do not contain structures to navigate over. Genre, time and location are the most prominent structures involved.

The used facets are:

- Time
  Time is visualized as a calendar showing the current day, and the following six days. The choice for this presentation is simple; the minimal bound of the dates in the gathered metadata is a week (XMLTV). Users can select a day and further specify the time period. Only broadcast information is time-bound as opposed to Video on-demand movies. A time selection can only filter the broadcasted results (XMLTV and BBC).
- Genre
  Is visualized as a TV-anytime classification tree. Users can select one or more genres for the programme they are searching for.
- Location

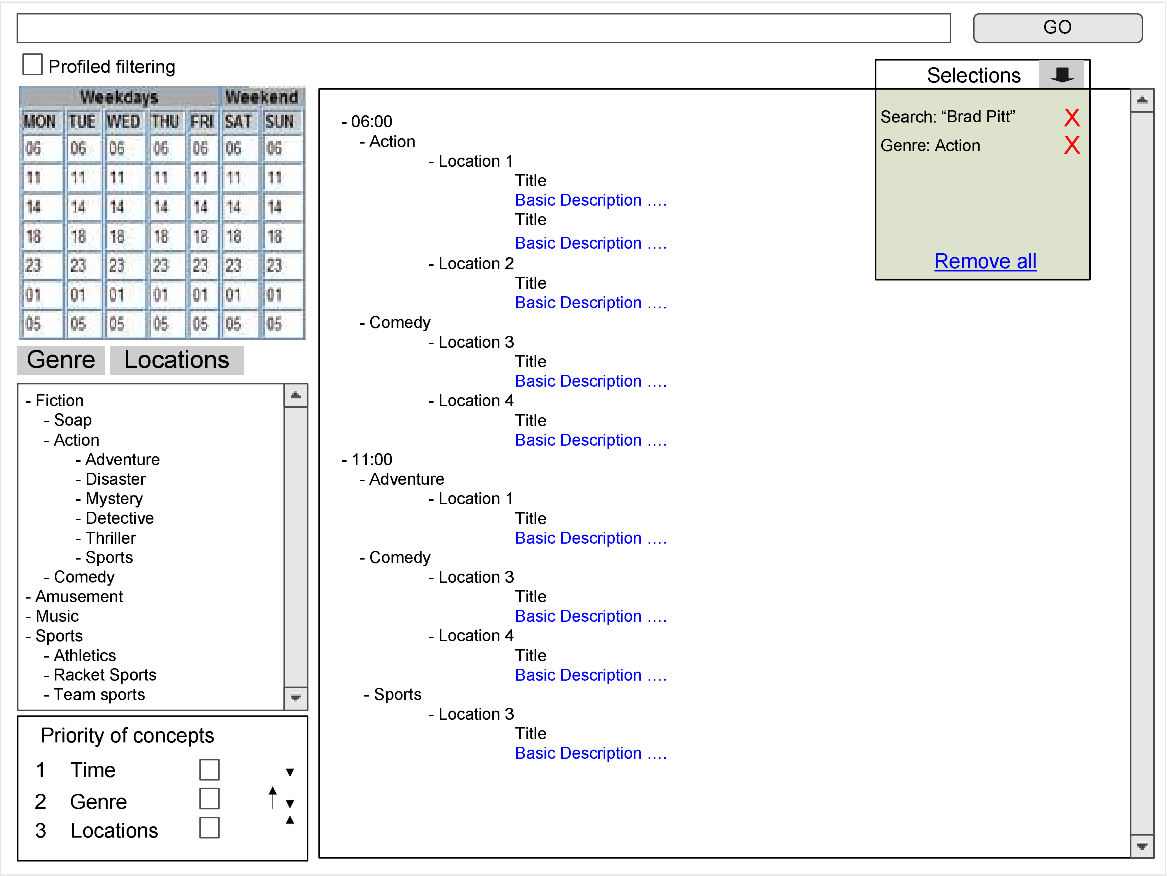
[26] http://www.stoneroos.nl/portfolio/case-study/Passepartout-personalised-EPG

Is visualized as an IMDb location tree. In the IMDb dataset every movie has filming locations. In other sets only countries are defined. Users can select one or more locations in the tree which limits movies that were shot in the selected locations. If a user selects a location that has sub-components, those will also be returned as a result. Noord-Brabant gives also the movies shot in Eindhoven. The interface tree is a direct implementation from the tree in the data set; at the top a root element, "the world", after the world the countries in the world are presented (218 in the IMDb set), and in the leaves the most specific locations.

The basis keyword search is similar to the Google search input. By default the input is matched to title, keywords, and actor names in all sources available.

The presentation of the results will be dynamically clustered in the 3 facets (the dimensions time, genre and location). Dynamically clustering enables the user to select certain facets and give an ordering priority to them. A user is free to choose a selection and ordering. Without a selection the result is just a simple list of items.

The description leads to an interface which is depicted in Figure 4.12. At the top a free text search, enabling the users to enter keywords. The profiled filtering option enables a user to search for items on basis of his preferences. At the top-left a calendar time to select days and time. Below the calendar component a tree structure is implemented, to select the genre items or location items. At the bottom-left, a component for changing priorities and selection of facets is implemented. At the right the content frame is located. In this frame the metadata will be presented; the result lists of the search or a window containing the metadata from a movie or person etc. Another component, the selection of facets and search queries, is located on top of the frame. This selection window makes it possible to remove the filters. All these filters can be seen as a refinement of a query (in query language, AND).



Weekdays					Weekend	
MON	TUE	WED	THU	FRI	SAT	SUN
06	06	06	06	06	06	06
11	11	11	11	11	11	11
14	14	14	14	14	14	14
18	18	18	18	18	18	18
23	23	23	23	23	23	23
01	01	01	01	01	01	01
05	05	05	05	05	05	05

```
- Fiction
   - Soap
   - Action
        - Adventure
        - Disaster
        - Mystery
        - Detective
        - Thriller
        - Sports
   - Comedy
- Amusement
- Music
- Sports
   - Athletics
   - Racket Sports
   - Team sports
```

1 Time □
2 Genre □
3 Locations □

Search: "Brad Pitt" X
Genre: Action X

```
- 06:00
   - Action
          - Location 1
                   Title
                   Basic Description ....
                   Title
                   Basic Description ....
          - Location 2
                   Title
                   Basic Description ....
   - Comedy
          - Location 3
                   Title
                   Basic Description ....
          - Location 4
                   Title
                   Basic Description ....
- 11:00
   - Adventure
          - Location 1
                   Title
                   Basic Description ....
   - Comedy
          - Location 3
                   Title
                   Basic Description ....
          - Location 4
                   Title
                   Basic Description ....
    - Sports
          - Location 3
                   Title
                   Basic Description ....
```

Figure 4.12: Mockup of interface

Chapter 5

Implementation

In the design an integrated common scheme has been created to facilitate a bridging of various content sources. Attributes like genre, time and location lend themselves to describe them in a structure/hierarchy. The sources and schemas have been integrated to be able to reason and navigate over it in a uniform connected way. Actually, to provide the system with data and to realise the design, the parts must be implemented. This means, that the metadata from XMLTV, BBC, IMDb and Videodetective must be retrieved and must be prepared to use it. Also a client (iFanzy) is implemented to enable users to get access to the (integrated) data set and to search for items. To implement all the parts described, some tools (a programming language, storage of data etc.) are necessary.

5.1.1 Implementation tools

The SenSee server has been implemented entirely using Sun Java 1.5. The choice for the Java programming language is based on the requirement that SenSee must be able to run on different operating systems (Windows, Linux).

For the scalability Sesame [27] [Appendix A.5] has been chosen as a framework for RDF storage, inferencing and querying (data delivery storage layer). All data handled within the SenSee platform is stored and retrieved by Sesame, including all metadata sources as well as the ontological sources. Sesame has an extensive API through which it becomes easy to embed it into other applications. Since the SeRQL [Appendix A.5.1] project is very alive (they are constantly working on extensions for it) and works flawlessly with Sesame, it was decided to support only this one. Several Sesame versions are used during the project. The reason is that Sesame is constantly in development, version 2 was launched last year and several Alpha and Bèta versions have been released.
Version 2 was introduced because it provides a better scalable storage and querying with reliable persistence, furthermore a revised Repository API that is much more targeted at embedded use of Sesame. Sesame supports a number of stores, one of them is only suitable for this project; the native store. A lot of metadata is involved, in terms of RDF, millions of triples. This cannot be stored in memory. A native store has a fast performance, high scalability and is persistent.

Web-based client application runs on the user's machine inside a web browser and connects to the TU/e public web server running Apache Tomcat.
This in turn connects to the SenSee Server on a back-end server machine, also running Tomcat. Likewise, external applications from other project partners connects via the Internet to the SenSee Server and the Services which are all deployed on the back-end server, though running as different processes (separate Java Virtual Machines).
The additional metadata is fetched by the SenSee Server by connecting via the Internet to external parties such as the Internet Movie Database (IMDb), XMLTV and BBC.

IFanzy is implemented as an AJAX application (Asynchronous JavaScript and XML). AJAX allows web services to retrieve information from the server by means of asynchronous calls. This means that when the client needs to retrieve a (calculated) value from the server this value can be obtained without having to reload the entire webpage. This enables us to provide a fluent web experience without high waiting times. To implement Ajax, a suite of programs developed by Google are used. These tools are put together in the Google Web Toolkit[28] (GWT) [Appendix A.4]. This toolkit put restrictions on the use of the Java version, 1.4 is the latest version it supports.

[27] http://www.openrdf.org

[28] http://code.google.com/webtoolkit/

5.1.2 XMLTV Parser

The XMLTV parser is configured to receive the broadcast metadata from the UK and Netherlands only (129 channels). This choice does not put any restrictions in the generality of the framework. The source is crawled on daily basis. The mapping from the XMLTV format to the TV-Anytime format happens at the *Content Retrieval* & *Serving* layer.

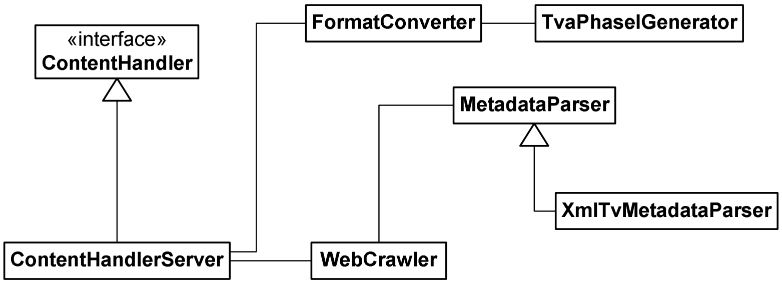


Figure 5.1: XMLTV Implementation

The XML-parser (*XMLTVMetadataParser*) parses the XMLTV formatted metadata into single attributes. These properties are received by the *TvaPhase1Generator*, which translates these single properties into TV-Anytime Phase 1. This mapping is done according to the mapping scheme (Figure 4.5). For example, genres are simple strings in XMLTV. With the mapping scheme these genre strings are mapped to a genre in the content classification scheme (CS).

5.1.3 IMDb Parser

The IMDb parser is responsible for reading, identifying, transforming and storing the IMDb metadata in the correct format, according to the IMDb scheme [Appendix B.1.4]. The *Content Retrieval* & *Serving* layer was partly implemented at the start of the project. The idea was to retrieve the IMDb metadata dynamically from the website, on the fly; a query from the user boils down to the lower levels of the framework and if the metadata from IMDb is not available, for that particular query, IMDb will be consulted to get the metadata. To realise that, the WebCrawler has been extended with an IMDb parser to get the metadata from the website. During the project this choice, caused problems and therefore the HTML parser was replaced by a parser which fetches the data from the IMDb text files (database dump).

The main reasons for this:

- The design of the IMDb website changed during the thesis. Like said in the design (Chapter 4), a change in the layout, can result in malfunctioning of the parser. To avoid this it is better to use a parser which operates on content structured files, without a mix of styles and content.
- IMDb does not allow for a great amount of request in a short time. This means that just a limited of queries can be asked in a certain time span. This also affects query time, because the system has to implement a time out.
- Query takes too long time. As a result of the time out and connections, performance of the queries at the IMDb server and parsing.

5.1.3.1 HTML Parser

The disadvantage of browsing and parsing the HTML structure is that the grabber has to submit queries and parse all the HTML pages to get access to the metadata. The HTML parser relies on the structure of an HTML page and is very sensitive for changes in the HTML. This means, if the design changes, also the grabber might need a redesign. This is due the mix of styles and tags. If one tag disappears from the page, it might be possible that the page cannot be parsed anymore. The first implementation of an IMDb grabber in the project was an HTML parser. The design is depicted in figure 5.2.

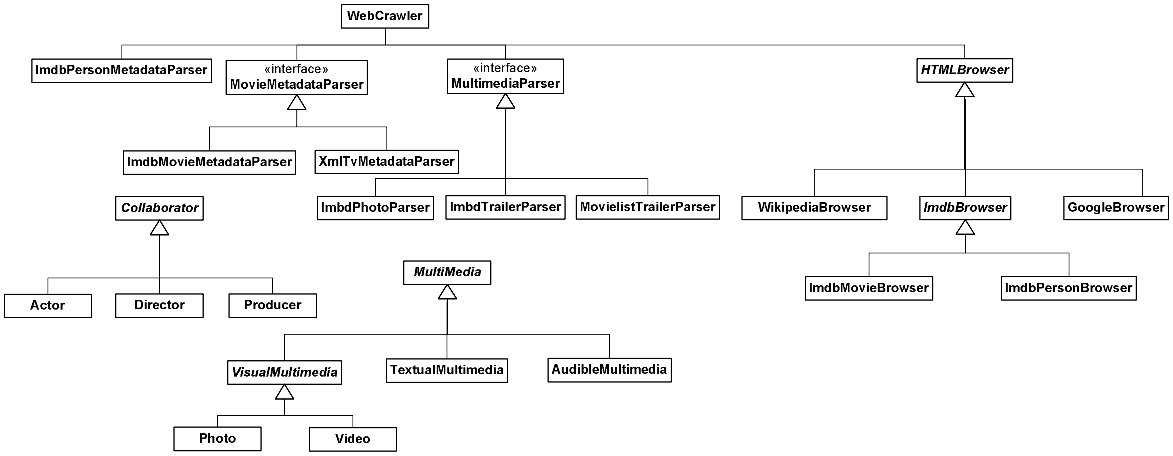


Figure 5.2: IMDb HTML parser extension

The parser was implemented in the Content Retrieval & Serving layer. This grabber was improved and extended, but it turned out that it was not satisfactory for the SenSee application, on basis of the earlier mentioned reasons.

The extension included 5 parts: Data types (Collaborator and Multimedia), Browsers, Multimedia parsers, Metadata parsers and a refinement of the WebCrawler that controls all this.

- The data type Collaborator describes a data structure of an actor, producer or other involved people. In a subclass the role or other particular properties of a collaborator are described. Multimedia is responsible for storing movies, photos and other kind of multimedia objects.
- These browsers are responsible for processing queries and returning result lists, such as an IMDB movie list. The result type is a list of URLs. An URL can be passed to a parser, to parse the actual page.
- The multimedia-parsers are responsible for parsing pages related to multimedia. It can parse IMDB trailer pages or photo galleries. It returns a list of URLs. These URLs point to the location where an image or a movie (trailer) resides.
- In the metadata-parsers, the metadata for a movie or person is collected. Also schedules can be fetched (XML-TV only). It takes a URL as input and returns a parser object. This parser object can be used later, to fetch particular metadata.
- The WebCrawler controls all the before mentioned parts. It joins all the parts to fetch all the metadata, data and URLs.

5.1.3.2 Plain text parser

The plain text parser fetches all the metadata from the text files and puts them into a Sesame store, modelled according to the IMDb scheme. During this phase also Videodetective is consulted, to deliver the trailer URLs and also the IMDb geographical vocabulary is filled with content. The photos are parsed by the HTML parser. Thus a part of the HTML parser (in 5.1.3.1) is reused in the WebCrawler to retrieve the photos.

The parser reads all the separate plain text files [Appendix C] from the IMDb offline database. The attributes of a movie or person are divided over a number of files. Thus, the title is defined in the movies file, the genres in the genres file and the keywords in the keywords file. All the main concepts like movie, person, and companies are defined uniquely (identifier strings) in IMDb. The plain text version of IMDb does not contain identifiers (e.g. 'nm060790', 'tt709090') like in the HTML version. Nevertheless, each movie title and person name in the files is unique. If a movie is produced in the same year with the same title then these movie titles are extended with a Roman Number. From these identifiers a hash string is created, which will be stored as main identifier for a particular instance (Movie, Person).

```
<rdf:Description rdf:about="http://wwwis.win.tue.nl/~ppartout/Blu-
IS/Ontologies/IMDB/Movie#4c3ff8df11bb4f9fd6c2f37253db778d">
<movie:movieTitle>Pirates of the Caribbean: Dead Man's Chest</movie:movieTitle>
<movie:movieKeyword>Piracy</movieKeyword>
</rdf:Description>
```

The first idea was to generate URIs extended with random identifiers; this to guarantee uniqueness and URI formatting (it is not allowed to use certain characters in URI or they must be translated). The random identifier has no relationship with the metadata. Thus, in the next IMDb data file, if the keyword is found, first the Sesame repository must be queried, to fetch the identifier. Due to performance of such a look up of titles or person names (string matching query), it was decided to avoid look ups and replace the random generated numbers by hashes of the title/names. It is known that every title is unique and can be used to generate the hash. The chance to have an ambiguous hash, two different strings with the same hash, is very small. To ensure that such a hash will not be accepted, a data structure was introduced to avoid the phenomenon.

The files are tab separated and almost each property can be easily distinguished by splitting the lines. The majority of the metadata can easily be identified, such as strings, integers and dates. Some metadata fields contain free text; the lack of structure makes it hard to parse. Regular expressions are used to identify dates, integers or strings. ISBNs can also be identified by regular expressions. Also some basic parsing was applied on the free text properties, to grab family relations, like mother of / son of relations. In the free text the problem of order arises; names can be ordered like "first name, last name", "last name, first name" and other combinations. The order in the free text is undefined. This makes the recognition, for example of family relations, much harder. One advantage is that links within free text are defined (by 'qv'); a link is a reference to a movie, person or another concept which exists in IMDb. To identify family relations, a simple algorithm is applied. This algorithm checks the text on strings like "`mother of #name_identifier#`" or "`father of #name_identifier#`" etc.

How regular expressions look like and work is clarified with the next example:

```
$50,000 Challenge, The (1989) (TV)
Pirates of the Caribbean: Dead Man's Chest (2006) 2006
Just an Example (????) ????

"Friends" (1994) 1994-2004
"Friends" (1994) {The One After I Do (#8.1)} 2001
"Rich Man, Poor Man" (1976) (mini) 1976
"Richard & Judy" (2001) {(2005-01-10)} 2005

Regular Expression: "^([^\t]+(\\)|\\}))([\t]*)([0-9?]{4,4}(-[0-9?]{4,4})*)*(.+)*$"
```

What the regular expressions does is to check if a line in the text file is conform the expression.
In natural language the expression checks the complete line (^ $). The attributes of a line are tab separated. In case of the movie text file, there is one tab separator, to distinguish the attributes "title" and "releaseyear(s)". All the characters before a tab are found by the expression `[^\t]+`. A title is always defined, so at least one character must be available and a title is always closed by a parenthesis or brace**.** Then there is an optional part, the releaseyears, separated from the title with a tab character. The optional part is defined in different ways, as can be seen in the excerpt above. Thus, a year is defined by 4 numbers or by question marks (date is unknown). Series have a release period, instead of having an instant. A period is defined by adding `-year`. Furthermore the year part can be followed by a number of characters, which are not important and thereby ending a line. The different parts of the string can be retrieved by a function. For example, parameter index1, returns the unique title string (attribute 1).

This regular expression is just one of the many expressions used in the parser. They form the heart of the parser. There are regular expressions used to identify dates (e.g. 18 September 2007 or 18-09-2007), to discover episode seasons and numbers, running times, titles, names etc.

Due to performance issues with Sesame, the number of requests on the Sesame repository has been reduced to an acceptable value. For example, string matching queries turned out to be very slow, but sometimes they are inevitable. These issues will be discussed later in 5.1.4. The number of accesses to the disk caused by Sesame has been reduced also. In the API of Sesame it is possible to turn off the auto commit, which means that the developer has to implement the commit. It is not recommend committing a newly inserted RDF triple immediately. It is better to create a batch of data to be stored, resulting in fewer hard disk accesses. This solution is implemented by creating a counter in the parser and to perform a commit if an optimal value is reached. This is a trade off between memory and performance, for the test system in use the value lies around the 2000 triples. This value is determined after some tests with a set extracted from the complete IMDb repository. Another measurement was to create an additional index on the repository (in 5.1.4). Finally, some metadata was kept in memory, an example of this is the parsing of countries. The number of countries is relative small and can be kept in memory. If “Netherlands” is processed once, it will be stored in Sesame, but also in memory. The next time the process finds “Netherlands” it only has to check if the country is in memory (skip) or not (store). These optimization steps discussed led to a better performance of the parsing process.

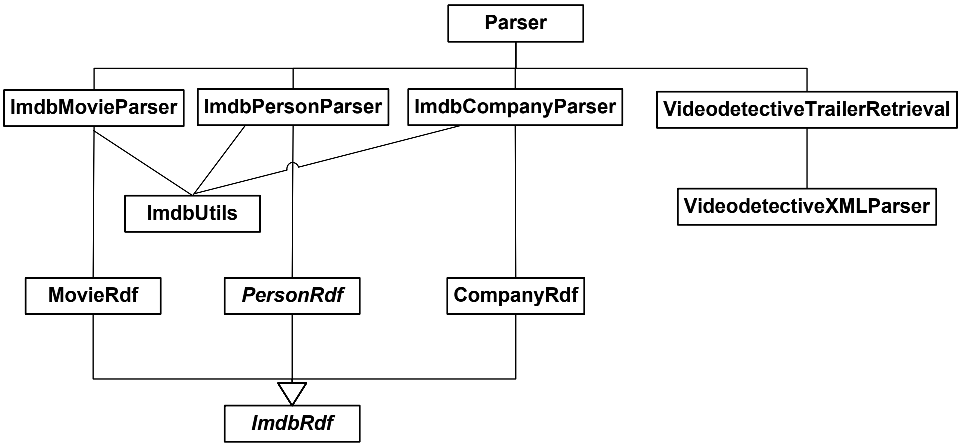


Figure 5.3: Plain text parser

In Figure 5.3 the plain text parser is depicted. The responsibility is divided over a number of classes, a Movie parser, a Person parser and a Company parser. The RDF classes do the actual storage, the parser only parses, unravels the attributes from the text and passes them to the RDF classes. The *VideodetectiveTrailerRetrieval*, connects with the Web service and receives XML documents. The *parser* class controls the whole.

5.1.3.3 IMDb location vocabulary

The location vocabulary is filled with locations defined in IMDb (e.g. filming locations, country and place of birth/dead). The identifier of a location is a concatenation of all the names of the different location levels (e.g. “Netherlands, Noord-Brabant, Eindhoven” in the leaf, “Netherlands” at the top).

```
Movie1 Eindhoven,Noord-Brabant,Netherlands
Movie2 Bosch,Noord-Brabant,Netherlands
```

If the parser checks the first line in the example, it starts to parse the line from the back to the front. First it finds a country name, “Netherlands”, it checks the repository if it already exists or not. If it exists it goes on and checks the next level, otherwise the “Netherlands” is stored in Sesame. This process goes on iteratively and it will result in 4 instances of a location, identified by their URI. *Movie1* has a relation with the first 3 instances. *Movie2* has relations with the first two and the fourth.

```
http://wwwis.win.tue.nl/~ppartout/Blu-IS/Ontologies/Geo/Imdb/Netherlands
http://wwwis.win.tue.nl/~ppartout/Blu-IS/Ontologies/Geo/Imdb/Netherlands/Noord-Brabant
http://wwwis.win.tue.nl/~ppartout/Blu-IS/Ontologies/Geo/Imdb/Netherlands/Noord-Brabant/Eindhoven
http://wwwis.win.tue.nl/~ppartout/Blu-IS/Ontologies/Geo/Imdb/Netherlands/Noord-Brabant/Den Bosch
```

The locations are stored in Sesame in a native store. Sesame also supports inferencing, but unfortunately not on native stores. For this project, the native store is used. Inferencing is the process of deducing (inferring) new information from information you already know. In Sesame an inferencing repository is available; a reasoner interprets all the information that is added to the repository, and does simple reasoning (such as type inheritance) on the RDF scheme level. This could be useful for the location tree. If for example inferencing could be used, it was possible to deduce things like Eindhoven isSubComponent of Noord-Brabant. Now this “inferencing” was mimicked by adding relations to all the locations, Netherlands, Noord-Brabant and Eindhoven instead of only one relation to one location, Eindhoven.

5.1.3.4 IMDb photos

The text dump does not contain links to IMDb photos. Grabbing all the photos at once is not possible, because otherwise the website of IMDb will be overloaded. To check all movies and persons in the database, a few million requests are necessary to get the photo URLs. That’s why the implementation from the photo parsed is integrated in the SenSee framework, at the CRS layer. The URLs are retrieved at runtime; if a user views the details of a movie or person then SenSee starts to retrieve the photo URLs. These URLs are stored in Sesame and are available for the next view. Thus, the second time, the photo URLs are fetched from the repository.

A script has been written to grab the URLs from the website. This works as follows: the user requests the metadata from a particular movie, to see the actual details. The metadata in the Sesame contains an identifier; this is the title from the text file.

```
Pirates of the Caribbean: Dead Man's Chest (2006)
```

This identifier is put in the search field of the IMDb search engine. The search engine has a build-in feature to redirect such a request to the detail page (http://www.imdb.com/title/tt0383574/) where the metadata can be found. Via that page al the other features can be accessed, like the photo galleries. From these photo galleries the photo URLs are taken. Besides that, the parser also retrieves the unique IMDB id (0383574). This is an identifier which is also used by a number of external sites, such as Wikipedia.

5.1.3.5 Videodetective

Videodetective is integrated in the plain text parser. It was also tried to implement the Videodetective, like the photo parser, completely at runtime; if a user enters the movie details, Videodetective will be consulted to fetch the trailer URLs. It turned out that the performance of the application decreased tremendously. Waiting times caused by the Web service are the main issue. Therefore it was decided to transfer the functionality to the plain text parser. In this setting, the parser does not delay the user presentation.

The integration of IMDb and Videodetective works via the principal of matching metadata fields (shown in Table 5.1). This matching is done by comparing strings. The title, director must be equal and at least 3 actors must be correct. This assumption has been made because the strings in the metadata not always match perfectly. Moreover in movie history never two equal movie titles have been produced, with the same three actors and directors. Videodetective delivers maximal 6 actor names. A test with a reduced dataset, 264 movie titles, resulted in 210 trailers found. 50 titles were not available and in 4 cases parsing problems have been detected. Furthermore, no false positives detected.

IMDb	**Videodetective Webservice**
Title: **Gladiator**	<Description>GLADIATOR featuring Richard Harris.</Description>
ReleaseYear: 2000	<Title>**GLADIATOR**</Title>
Director: **Ridley Scott**	<Studio>Dreamworks</Studio>
Actors:	<StudioID>253</StudioID>
Russell Crowe	<Actor1>**Djimon Hounsou**</Actor1>
Joaquin Phoenix	<Actor2>**Joaquin Phoenix**</Actor2>
Connie Nielsen	<ActorId2>9612</ActorId2>
Oliver Reed	<Actor3>**Russell Crowe**</Actor3>
Richard Harris	<ActorId3>7349</ActorId3>
Derek Jacobi	<Actor4>**Tomas Arana**</Actor4>
Djimon Hounsou	<ActorId4>8188</ActorId4>
Tomas Arana	<Actor5>**Richard Harris**</Actor5>
…	<ActorId5>6873</ActorId5>
	<Actor6></Actor6>
	<Director>**Ridley Scott**</Director>
	<DirectorID>6958</DirectorID>
	<Link>http://www.videodetective.com/trailer-preview.asp?publishedid=814369</Link>
	<PublishedId>814369</PublishedId>
	<Duration>226</Duration>

Table 5.1: IMDb-Videodetective matching

5.1.4 Performance IMDb set

The biggest issues during the thesis encountered were the conversion from the IMDb dataset to a RDF repository and the performance of the queries. Sesame was chosen because the scalability and also because the close collaboration with the creators of it (Aduna). Version 2 (Alpha and Bèta versions) allowed to use new features in SenSee, but it turned out that this came in a package with a poor query optimization, discovered during the thesis, especially critical when using large datasets like IMDB. The complete IMDb repository contains 84 million triples, whereof almost 1 million movie titles.

The performance of the repository was tested on a Linux server, Pentium4 3 GHz with 1024 MB internal memory. Version 2 of Sesame was investigated by a number of tests. A first test set was created to give a general impression about the performance. A common query in SenSee is a string matching query, where an end-user can request movies with a particular title, keyword or something similar to that. Another possible query is to return all the cast members in a particular movie. From these tests it was evident that a number of queries performed too slow to be used in a user interface. Sesame provides a method to match strings with a high performance; however this is based on exact matching. Thus, a string "Another" will only find titles "Another", very unpractical if somebody was looking for "Another time" or "Die another day". The results of pattern matching turned out to be quite bad. The query used is described below. The title and limits will be set in the two variables, respectively title, limitStr. Limit is a method to limit the number of results; the query process stops if the limit has been reached.

```
SELECT m FROM {m} movie:movieTitle {z} WHERE z LIKE \""+title+"\" "+limitStr+"
```

The repository contains 976.174 movie titles. * is a character to determine at which side the pattern matching must be done. For example, Another*, finds "Another time" but not "Die another day".

Movie	Limit	Execution time (ms)	#Results
Another*	0	49469.79	134
Another*	10	1433.443	10
Another*	5	1419.04	5
Another*	2	1528.582	2
Another	0	48537.738	289
Another	10	1523.758	10
Another	5	1423.365	5
Another	2	1081.337	2

Table 5.2: Pattern matching results

In Table 5.2 it can be seen that the pattern matching at one side or at both sides does not really affect the query time. The algorithm in Sesame checks every single title (almost one million triples) for this pattern. From two another queries, tested on synopsis and keywords instead of a title, with the knowledge that average(keyword length) < average(title length) and average(synopsis length) > average(title length), it was concluded that the length of the field to be matched, does affect the query time. Therefore it can be concluded that the pattern matching queries depend on the number of triples involved and the length of the stored string.

A second step in the performance tests was to create a test which checks the performance of queries by shrinking the repository and by creating additional indexes on the triples. To check this, 13 basic queries have been defined. Note that query 1 trough 5 can also be defined as path expressions, of length 1.
Each query has been repeated 10 times to ensure that extreme values can be omitted from the end result.

1. Subject query
2. Multi value subject query
3. Predicate query
4. Object query
5. Multi value object query
6. Where query
7. Like query, pattern title*
8. Like query, pattern *title*
9. Like query, pattern *title
10. Equals query
11. Path expression query, length 2
12. Path expression query, length 3
13. Path expression query, length 4

All the queries have been defined to test the general performance. However 1 through 5 and 11 through 13 also to test the performance of the addition of indexes, 6 and 10 just for basic querying and 7,8,9 for the like (pattern matching) operation. Some of them will be described in this thesis, to explain the discovered issues. To understand the definition of the described queries, the SeRQL manual[29] should be consulted.

Shrinking the repository leads to an improvement of performance; there are fewer triples to be checked. The native store uses on-disk indexes to speed up querying. It uses B-Trees for indexing statements, where the index key consists of four fields: subject (S), predicate (P), object (O) and context (C). The order in which the fields appear determines the usability of an index on the query pattern: searching statements with a specific subject in an index that has the subject as the first field is significantly faster than searching these same statements in an index where the subject field is second or third. In the worst case, a query will result in a sequential scan over the entire set of statements. By default, the native repository only uses two indexes, one with a subject-predicate-object-context (SPOC) and one with a predicate-object-subject-context (POSC). However, it is possible to define more or other indexes for the native repository. This can be used to optimize performance for query patterns that occur frequently. Creating more indexes potentially speeds up querying (a lot), but also adds overhead for maintaining the indexes. Also, every added index needs additional disk space.

The first run (in Figure 5.4 and 5.5, on x-axis numbered 1) checked the performance of the original set and a second run (x-axis 2) checked the performance of the set with an additional OPSC index, next to the SPOC and POSC. The additional index needs also some additional disk storage, around 3.5 Gigabytes in this case.

The next test runs ran on a subset of the original repository, by removing triples. The condition to shrink the repository in each run is based on the popularity of a movie; in the third run all movies with a number of votes smaller than 1 (x-axis 3) were removed, in the next run 10 (x-axis 4), 25 (x-axis 5),100 (x-axis 6),200 (x-axis 7),500 (x-axis 8) and finally 1000 (x-axis 9). The number of triples decreased by 31.2 million (from 83.7 to 52.5 million), the number of movie items by 969.000 (from 976.174 to 7.173).

[29] http://www.openrdf.org/doc/sesame/users/ch06.html

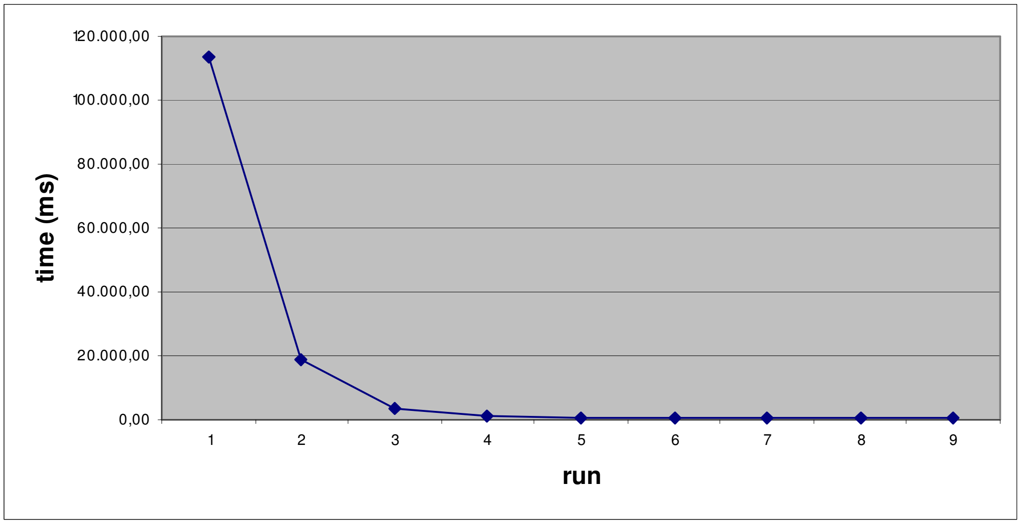


Figure 5.4: Object Query (Query4)

Depending on the queries used in an application, an additional index can have added value. In the test runs an additional index OPSC was added to the repository.

This addition leads to a 6 times better performance on object queries (Figure 5.4). The first run takes about 120 s and the second (with OPSC) ends in 20 s. An object query (Query4) is for example, if a user wants all the movies in a certain genre or at a particular time. The genre or time instance is defined by a URI (of the resource), which is the object and fetches all the movies, the subject.

```
SELECT *
FROM {x} y {<http://wwwis.win.tue.nl/~ppartout/Blu-IS/Ontologies/Time/Instant/8f9c1a417155361a>}
```

Also better performances (10% improvement) were measured for Query5, 11 and 12. The reason for this is that they also use the additional index file to enable faster lookups.

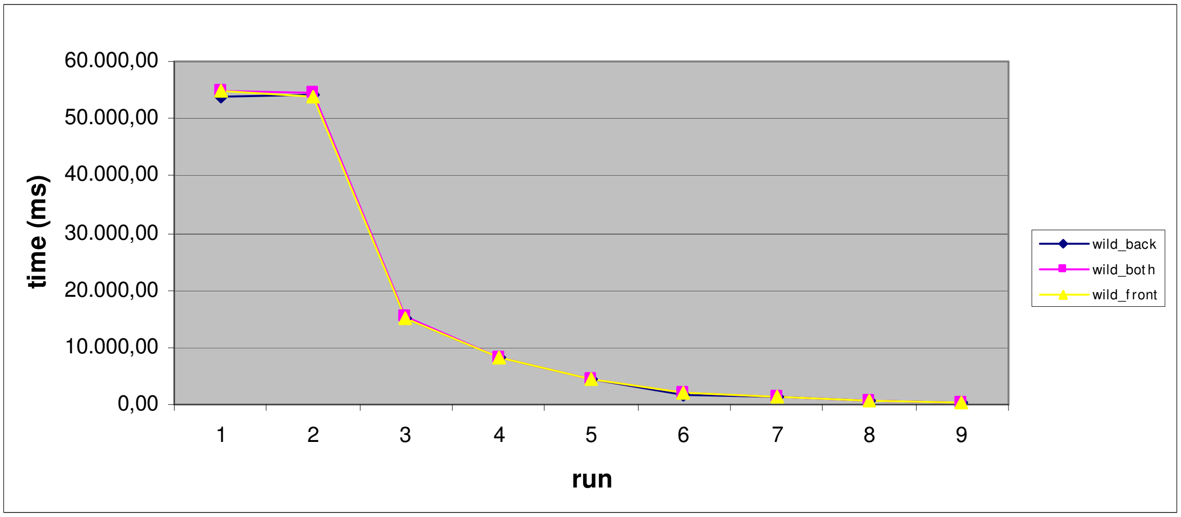


Figure 5.5: String Matching Queries (Query7, 8, 9)

In Figure 5.4 the performance of string matching queries is depicted. Query7 is defined as *wild_back* (from wildcard back), query8 as *wild_both* and Query9 as *wild_front*. Query7 uses a wildcard at the end of the keyword, Query8 uses two wildcards and Query9 uses a wildcard at the beginning. All queries have a quite similar performance. Creating an OPSC index makes no difference (difference between run 1 and 2). At run 3 the movie titles decrease from 976.174 to 261.749 triples. In the end it can be concluded that the behaviour is linear, so by reducing the set 3 times, the query will perform 3 times faster. This assumption is correct, because for each movie title triple, the string property is checked. However the performance of the string matching also depends on the length of the string in each item. For example the length of a string in the property keyword is on average less than the length of a title string. On the other hand, the length of a synopsis is on average greater than a title string. This means that queries on equal sized sets, but with different string lengths, need different processing times.

The last important issue is the absence of a query optimizer. This can be tested by a sample query:

```
SELECT p FROM
{p} person:hasRole {Role} person:inMovie
{<http://wwwis.win.tue.nl/~ppartout/Blu-IS/Ontologies/IMDB/Movie/36b9abf5121f2437bb>}
```

This query retrieves all persons which have a role in a particular movie. The query takes minutes to complete. If the query is split into two parts, first finding the roles per movie and finding with these results the corresponding persons is much faster than executing this query at once. It can be concluded that there is no query optimization in this case.

In general the query optimization of Sesame version (2) is limited. The right choice for indexes affects the performance of the queries. Good results are coming from queries with defined resources on subject or object and equality queries, note that the correct indexes must be used.

These tests resulted in a more detailed investigation by other developers of SenSee and also by Sesame (Aduna) developers. From these results, it was decided to use Sesame 1 instead of 2, because version 1 had a more advanced query optimizer at that time. Secondly the repository was shrunk to the 11.058 (Run 8 in Figure 5.5) most popular movie items in the set. The metadata from the other movie titles was deleted. Company metadata was also deleted, because at that moment the metadata was not in use. Hereby Research Question RQ2 is partial answered. The performance was increased by the solutions described in this paragraph, but still needs to be improved. Also this part of the investigation of the performance focused on the IMDb set and not in the complete (integrated) set.

5.1.5 Client application: IFanzy

The implementation of IFanzy is realised by using the Google Web Toolkit (GWT). It is an open source Java software development framework that makes writing AJAX applications easier. AJAX allows web services to retrieve information from the server by means of asynchronous calls. This means that when the client needs to retrieve a (calculated) value from the server this value can be obtained without having to reload the entire webpage. This enables a fluent web experience without high waiting times.

This asynchronous behaviour is depicted in the Figure 5.6; the client communicates asynchronously with the Web Server. Besides that it can also be seen that the client is implemented in Java 1.4 and the Server in Java 5.0 (1.5). XMLRPC is a specification and a set of implementations that allow software running on different operating systems, running in different environments to make procedure calls over the Internet. The remote procedure call uses HTTP as the transport and XML as the encoding. It allows to transmit, process and return complex data structures.

Thus, the the SenSeeWebClient interacts with the BoxApplicationServer, to request and receive the content. In this BoxApplicationServer functions are implemented to search for content, to login users etc.

If a user searches for content and navigates through the interface then queries will be send to the SenSee framework to retrieve the content. The query is received by the BoxApplicationServer which is part of the Application server layer.

In Figure 3.1 it can be see that Application server layer connects to the personalization layer, which handles search queries, by semantic analysis of search keywords (WordNet), retrieval of user interests and context. Furthermore, it makes use of the ontologies to provide semantic-based filtering of content. Through its service handlers it connects to the User Model, Ontology and Filter services. The extended and updated query is given to the package handling layer where the content integration takes place. At that place the metadata from the different sources is collected and metadata is stored in Sesame.

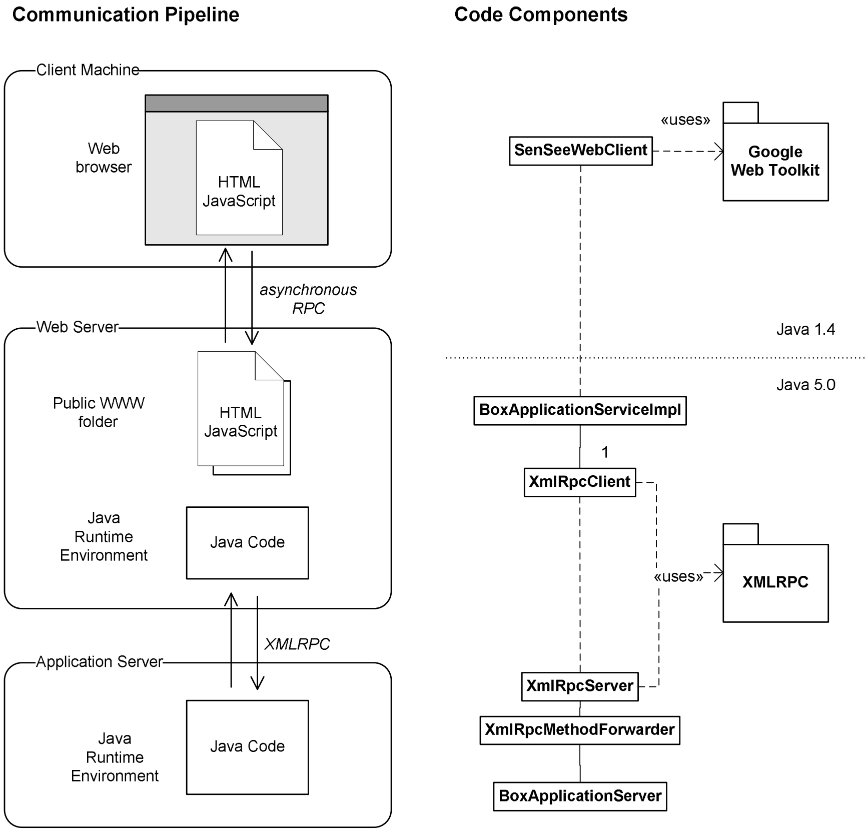


Figure 5.6: Client – Server communication

5.1.5.1 Presentation

During the thesis several custom components have been implemented. GWT provides a lot of basic components, but some of the necessary components had to be designed from scratch, from a basic concept or are a composition of existing components. The implemented components are:

- Movie detail panel, presenting the movie metadata
- Person panel, presenting the person metadata
- Trailer panel, presenting the trailer
- Priority changer (to order and select the facetted presentation)
- Photo gallery integrated in person and movie panel
- Calender-time component, to enable searching for time
- Rating component, to give a rating for a movie or person

The final interface differs a little from the design. At first the calendar-time component differs from the one in the design. This was requested because the calendar component should more resemble the calendar interfaces we know from other applications and thereby feel more natural for the user. The priority changer is also slightly different; instead of arrows to click on, a drag and drop mechanism was introduced. The frame on top of the content frame, to remove filters, disappeared, because it was left out from the interface, for now. All these measurements resulted in an interface which is shown in Figure 5.7 and 5.8.

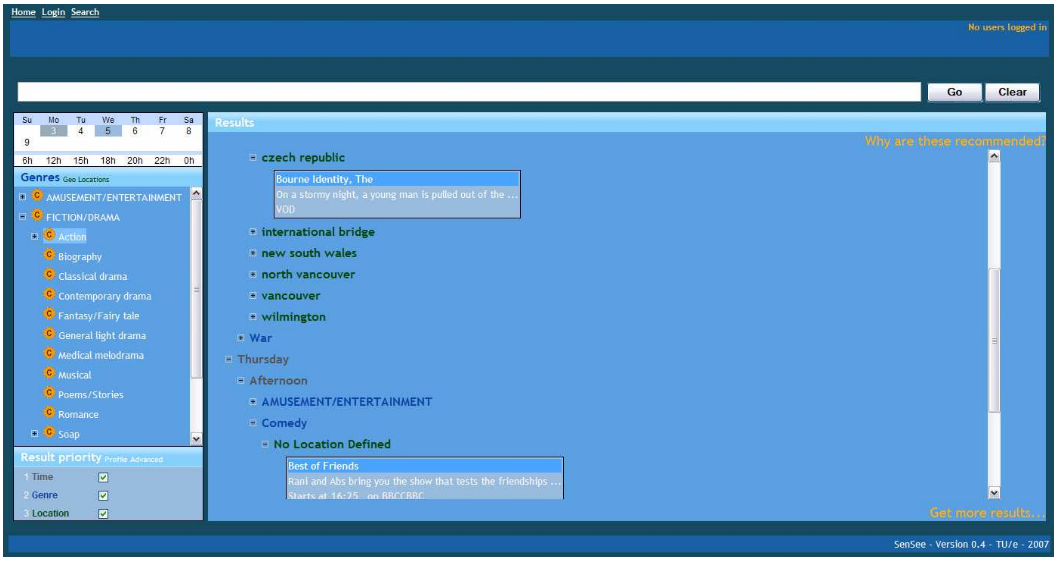


Figure 5.7: Search Screen

What a user basically can do in the screen of Figure 5.7, is search for content, by facets and free text. Genre, time and location have been selected as the three main facets (see 4.4).

- Time: Is visualized as a calendar showing the current day, and the following six days. The users can select a day and further specify the time period. A time selection can only filter the broadcasted results (XMLTV and BBC).
- Genre (Left Tab in interface): Is visualized as a TV-anytime classification tree. Users can select one or more genres for the programme they are searching for.
- Location (Right Tab in interface): Is visualized as Geo vocabulary (IMDb location tree). The interface tree is a direct implementation from the tree in the data set; at the top a root element, "world", one level lower the countries are presented and in the leaves the most specific locations.

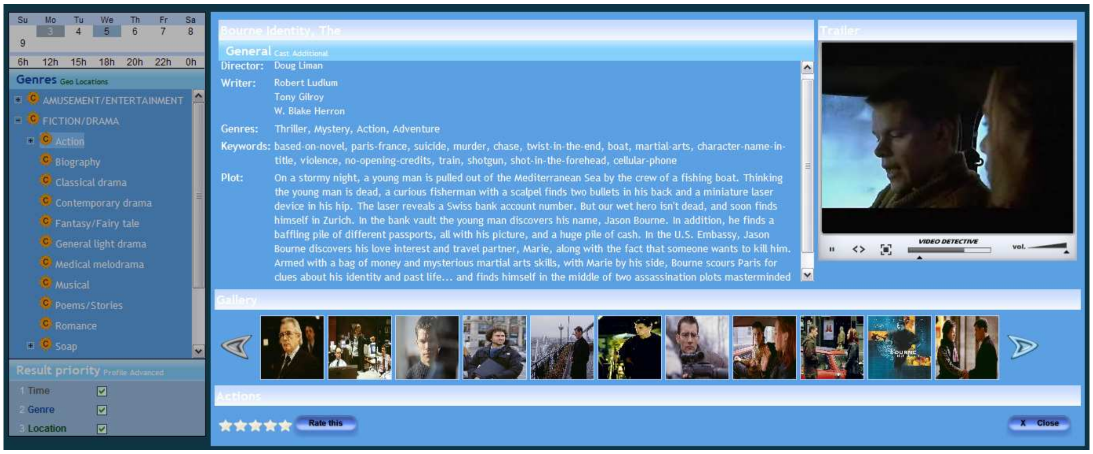
The genre and location trees are part of a tabbed frame. The selection of one of the tabs allows the user access to the trees.

The basis keyword search is similar to the Google search input. By default the input is matched to title, keywords, and actor names in all sources available. There is also an advanced search that triggers some actions before the query is executed. The advanced functionality provides in:

- Keywords broadening: Configured to query for synonyms in WordNet, i.e. terms appearing in the same set as the input term. Currently the functionality is limited, the keyword expansion limits to 5 new terms per keyword. This is also due to the earlier explained performance issues in 5.1.4.
- Keyword conceptualization: All facet-sources (time, genre and location) are parsed for match for every keyword in the broader set. This matching has different modes; to be able to configure the matching. For example, in strict mode only the keyword Sports matches with the genre concept "Sports" and a free match does pattern matching and also finds concepts like "Motorsports".
- Concepts broadening: For a given object property we traverse the corresponding graphs and find related concepts. For the TV-Anytime genre classification it results in all concepts that are narrower than the input concept. For example, *tvacontent:3.2* (Sports) genre will also find the *tvacontent:3.1.1.9* (Sports News), as all programs of the genre Sports News are also about Sports.

The results are dynamically clustered in the 3 facets (time, genre and location). Dynamically clustering enables the user to select certain facets and give an ordering priority to them. A user is free to choose a selection and ordering. Without a selection the result is just a simple list of items. This selection and ordering can be done in the priority changer, in the left corner of the screen. The top priority facet will make out the first level of the tree. If Genre is of highest importance, the first level tree will contain the genres of the first N results in alphabetic order. Grouping occurs in a similar fashion for the extra facets. Note that programs can have several genres and results therefore can occur more than once in the tree.

In the content panel the presentation of the resulting items is implemented. The items are grouped by the selected facets and also ordered according to the ordering in the priority changer. If no location, no time or no genre for specific items is defined then there will be an additional group of items to present them.



Director: Doug Liman

Writer: Robert Ludlum
Tony Gilroy
W. Blake Herron

Genres: Thriller, Mystery, Action, Adventure

Keywords: based-on-novel, paris-france, suicide, murder, chase, twist-in-the-end, boat, martial-arts, character-name-in-title, violence, no-opening-credits, train, shotgun, shot-in-the-forehead, cellular-phone

Plot: On a stormy night, a young man is pulled out of the Mediterranean Sea by the crew of a fishing boat. Thinking the young man is dead, a curious fisherman with a scalpel finds two bullets in his back and a miniature laser device in his hip. The laser reveals a Swiss bank account number. But our wet hero isn't dead, and soon finds himself in Zurich. In the bank vault the young man discovers his name, Jason Bourne. In addition, he finds a baffling pile of different passports, all with his picture, and a huge pile of cash. In the U.S. Embassy, Jason Bourne discovers his love interest and travel partner, Marie, along with the fact that someone wants to kill him. Armed with a bag of money and mysterious martial arts skills, with Marie by his side, Bourne scours Paris for clues about his identity and past life... and finds himself in the middle of two assassination plots masterminded

Figure 5.8: Movie detail Screen

If a user selects a programme item in the presented result list then the client opens a new window. This window contains the metadata describing the programme item. The metadata is divided over a number of tabs, thereby grouping the metadata into specific subjects. The *General* tab contains the metadata describing the movie itself, such as genres, keywords and plot. The *Cast* tab presents all the involved actors. By clicking an actor in that tab, the client opens a window containing information about the actor, like biography, birth date etc. This window is similar to the movie detail window. The *Additional* tab, presents information that is also about the movie, but it can be seen as a nice extension on the basic information.

Furthermore trailers and the photo gallery are added to make the interface more vivid and to give the user features to recognize the movie or a person easily and also to attract the user to watch the movie.

At the bottom, a rate this feature is implemented; this to enable users to rate movies or persons, which will be used in SenSee to personalise. If a user dislikes a specific movie then the system takes this rating into account in giving recommendations for movies.

5.1.6 Final dataset

The final dataset/vocabularies used for the Semantic Web Challenge 2007 demo are shown in table 5.3.

Data source / Vocabulary	**#triples**
User Model (schema)	319
IMDB schema	408
TV Anytime Genre Classification	3.971
Geo Ontology	59.835
Time Ontology	1.547
Country Codes	2.023
WordNet	1.942.887
BBC dataset (random pick)	91.447
XMLTV dataset (random pick)	1.278.718
IMDB dataset	7.969.199

Table 5.3: Size of sources, vocabularies and schemes

In this table it can be seen that a strongly reduced set of IMDb is used, just around 8 million triples, due to the performance issues. Only the 11.058 most popular movies were used. This reduction does not put any restrictions in the generality of the framework. It is enough to show the purpose and working of the demonstrator. In this set, photos and trailers (from Videodetective) are included. The XMLTV set contains around 1.3 million triples, this to describe around 20.000 programme items and their relations. BBC takes 92.000 triples to describe 1575 programme items.

Chapter 6

Conclusion and Future work

Several problems and issues have been tackled during the thesis work. In this section a brief reflection on them are presented and draw lessons learned. The research questions are used to present them in a structure way.

Integration of distributed data collections is one of the features of SenSee. These content sources & vocabularies can differ in semantics, layout and formatting. (Semantic) mappings are necessary to realise integration. XMLTV and BBC Backstage are chosen as metadata source for broadcasts and IMDb is the rich metadata source mimicking the working of the VOD concept. For IMDb a scheme was created on basis of the original IMDb dataset. The scheme uses some vocabularies to enable reasoning and categorisation. Attributes like genre, time and location lend themselves to describe them in a structure/hierarchy. XMLTV metadata is described in XML and BBC in TV-Anytime. The sources and schemas have been integrated to be able to reason and navigate over it in a uniform connected way. The mapping to TV-Anytime enables this uniformity. OWL-Time, WordNet and Geo (IMDb locations) are vocabularies enabling reasoning about time, extending keywords and reasoning about locations. This fulfils research question RQ1.

A slow performing application is not very useful for an end-user. The integrated dataset should be accessible in a way that the user can easily view and search the content without too long waiting times. For a very large and diverse set it is more likely to encounter performance issues. The biggest performance issues have been encountered during the conversion from the IMDb dataset to the RDF repository and the performance of the queries. Sesame was chosen, as the underlying RDF querying and storage framework, because the scalability and also because the close collaboration with the creators of it (Aduna). Version 2 allowed using new features in SenSee, but it turned out that this came in a package with a poor query optimization and bad performance of string matching queries, especially critical when using large datasets like IMDB. The right choice for indexes also affects the performance of the queries. Good results are coming from queries with defined resources on subject or object and equality queries, note that the correct indexes must be used. It was decided to use Sesame 1 instead of 2, because version 1 had a more advanced query optimizer at that time. Secondly the repository was shrunk to the 11.058 most popular movie items in the set. The metadata from the other movie titles was deleted. A smaller set (about 10 times) performs a lot better than the original IMDb set. The investigation by other developers to find a scalable solution with a high performance is still going on; they think to manage an improvement in evaluation time up to a factor 10. It is not recommended to shrink the set. An integrated dataset was necessary to demonstrate the client-application and the reduction does not put any restrictions in the generality of the framework. However the overall result is a slower and less scalable client application (demonstrator). Thereby research question RQ2 is partial answered.

Research question RQ3 deals with the presentation of the integrated set. For an end-user of the system the experience to access and search the content is important. The main issue is how the system can present hierarchical structures involved in the semantic integration to the user.
An end-user wants access to the data, however SenSee itself only allows an (personalised) integrated view on data harvested from the different sources. Therefore an application on top of the framework, iFanzy, has been designed to provide access to the integrated view. This integration needs special attention of the search and browser interface presented to the user. Therefore two ways to access and navigate TV content have been introduced, by search and by facet. Faceted-based presentation of search results is a good demonstration of combining multiple perspectives in one view. Genre, time and location have been selected as the three main facets. The presentation of the results is dynamically clustered in the 3 facets (the dimensions time, genre and location). Dynamically clustering enables the user to select certain facets and give an ordering priority to them.

The implementation of IFanzy is realised by using the Google Web Toolkit (GWT). This enables a fluent web experience without high waiting times. The application depends on GWT's portability and efficiency, it turned out for example that the application is rather slow and inflexible with tree-rendering.

Summarized, some big issues appeared during the thesis.

- The use of live data: Initially it was aimed to work with all live data. For instance, it was not the main idea to store the offline text dump of IMDb, as it is not refreshed that often , but query it live all of the time and parse (and scrape) its HTML-pages. However, after experiencing layout changes, and thus rewrites of the parsers, the text dump parser was implemented. Other reasons also lead to this choice: IMDb does not allow for a great amount of request in a short time and query takes too much time.
- Performance of the entire IMDb set was quite bad. This is already explained in this conclusion of RQ2.

For a future extension, Wikipedia is a good candidate, during the thesis it was not implemented because the scheme was not available and also the time would have restricted the implementation. Another extension could be to enrich the metadata mimicking the broadcast concept with metadata from other sources. A nice addition is for example trailers and photos, like provided in the IMDb set. The last future source extension could be the integration of a source like Youtube, this is also on demand source, but is completely driven by users (e.g. they annotate their videos themselves).

The current faceted-based presentation of the search result is a good demonstration of combining multiple perspectives in one view. However in terms of a conventional TV-guide the typical channel ordering is missing. Thus, in the future a combination of more different facets should be investigated/implemented.

Glossary

API An application programming interface (API) is a source code interface that a computer system or program library provides in order to support requests for services to be made. An API is often a part of a software development kit (SDK).

AJAX Asynchronous JavaScript And XML (AJAX), is a web development technique used for creating interactive web applications. The intent is to make web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire web page does not have to be reloaded each time the user requests a change. This is intended to increase the web page's interactivity, speed, functionality, and usability.

EPG An Electronic Programming Guide. is an on-screen guide to scheduled broadcast television programmes, allowing a viewer to navigate, select, and discover content by time, title, channel, genre, etc, by use of their remote control or keyboard.

GWT Google Web Toolkit (GWT) is an open source Java development framework that lets you escape the matrix of technologies that make writing AJAX applications so difficult and error prone. GWT enables development and debugging of AJAX applications in the Java language using the Java development tools of your choice. When the application is read for deployment, the GWT compiler can translate the Java application to browser-compliant JavaScript and HTML.

HTML Hypertext Markup Language, is the predominant markup language for web pages. It provides a means to describe the structure of text-based information in a document by denoting certain text as headings, paragraphs, lists, etc. Furthermore to supplement that text with interactive forms, embedded images, and other objects.

IMDb The Internet Movie Database (IMDb) is an online database of information about movies, actors, movie stars, TV shows, TV stars, production crew, as well as video games. It is the biggest known collection of movie related material.

Java Java is an object-oriented programming language developed by Sun. It 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 programmes to a new platform all that is needed is a port of the interpreter and a few native code libraries.

Metadata Metadata (meta-data) is generally defined as "data about data," which is of course a very broad definition. In computer science this is generally taken to mean information about a set of data in a particular representation, which typically means schema information, administrative information, and so on. However, in content management and information architecture, metadata generally means "information about objects", that is, information about a document, an image, a video, and so on.

MPEG-2 MPEG-2 is widely used as the format of digital television signals that are broadcast by over-the-air (terrestrial), cable, and direct broadcast satellite TV systems. It also specifies the format of movies and other programmes that are distributed on DVD and similar disks. As such, TV stations, TV receivers, DVD players, and other equipment are often designed to this standard.

MPEG-7 MPEG-7 is a formal system for describing multimedia content. This description will be associated with the content itself, to allow fast and efficient searching for material that is of interest to the user. Thus, it is not a standard which deals with the actual encoding of moving pictures and audio, like MPEG-1, MPEG-2 and MPEG-4.

Ontology An ontology is a data model that represents a set of concepts within a domain and the relationships between those concepts. It is used to reason about the objects within that domain.

OWL The OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans OWL was designed to provide a common way.

Query Query is of set of terms. It can be a single word, phrase or set of words characterizing the information a user seeks from the system.

RDF RDF (Resource Description Framework) is a language that is used in the Semantic web to model meta-data about the resources of the web. RDF statements are nothing more than statements that consist of triples because everything in RDF is a statement with triple i.e. subject, property and object.

RDFS RDFS (RDF Schema) is a language for describing vocabularies in RDF. RDF Schema is a semantic extension of RDF. It provides mechanisms for describing groups of related resources and the relationships between these resources.

Sesame Sesame is an open source framework for storage, inferencing and querying of RDF data.

SeRQL Sesame RDF Query Language, is an RDF query language.

SKOS The Simple Knowledge Organisation System (SKOS) is a family of formal languages designed for representation of thesauri, classification schemes or any other type of structure controlled vocabulary. Concepts can be organized in hierarchies using broader-narrower relationships, or linked by non-hierarchical(associative) relationships.

TGN The Getty Thesaurus of Geographic Names (TGN) is a structured vocabulary providing geographical information.

Trailer A trailer is an advertisement for a movie, series or a programme that will be shown in the future. It is an attractor, with carefully selected shots of strong visual moments, effects and dialogues. The main purpose is to get the people interested and that they decide to watch the movie, series or programme.

URI A Uniform Resource Identifier is a compact string of characters used to identify or name a resource. The main purpose of this identification is to enable interaction with representations of the resource over a network, typically the Web, using specific protocols.

URL Uniform Resource Locator is an URI that identifies a resource via a representation of its primary access mechanism (e.g., its network "location"). In other words a URL also tells how to retrieve the resource.

W3C The World Wide Web Consortium (W3C) develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential. W3C is a forum for information, commerce, communication, and collective understanding.

Web service The W3C defines a Web service as a software system designed to support interoperable Machine to Machine interaction over a network. Web services are frequently just Web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services.

XML XML provides a surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents.

XML Schema XML Schema is a language for restricting the structure of XML documents and also extends XML with data type.

References

[1] ITEA Passepartout Full project proposal
1. Baker, K. & Driessen, H. (Ed.) (2005)

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

[3] SenSee Framework for Personalized Access to TV Content
Lora Aroyo, Pieter Bellekens, Martin Bjorkman, Geert-Jan Houben, Paul Akkermans, and Annelies Kaptein. In: EuroITV 2007. Pages 156-165.

[4] Semantic Annotation of TV-Anytime Data for Personalized TV Content Delivery
Pieter Bellekens, Lora Aroyo and Martin Bjorkman. ESWC 2006.

[5] SenSee white paper – Technical Architecture Description.
TU/e Working Draft, 03 September 2007

[6] On the Construction of a Personalized Home Media System using TV-Anytime packaging
T.J. Dekker & E. Loef. Master Thesis

[7] Personalized Query Refinement in Home Media System
Charanjeev Kaur

[8] TV-Anytime Phase I
ETSI TS 102 822-3-1

[9] TV-Anytime Phase II
ETSI TS 102 822-3-3

[10] Conversion of WordNet to a standard RDF/OWL
Mark van Assem and Aldo Gangemi and Guus Schreiber
May 2006, Genoa Italy, Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)",

Appendix A

Implementation tools

A.1 XML

According to W3C, the Extensible Markup Language[30] (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 Document Type Definition (DTD) or XML Schema[31]. 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. 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 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.

A.2 RDF

The Resource Description Framework[32] (RDF) is a framework for describing and interchanging metadata which is built around the XML 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) 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 (subject, predicate, object), consisting of the resource being described (subject), the property's name (predicate), and the property's value (object). 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[33]. 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.

[30] http://www.w3.org/XML/

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

[32] http://www.w3.org/RDF/

[33] http://www.w3.org/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.

A.3 OWL

OWL[34] 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.

- OWL Lite, being the simplest
- OWL DL
- OWL Full, being the most expressive

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.

A.4 GWT

Google Web Toolkit[35] (GWT) is a suite of programs developed by Google, which it provides free to the open source community to implement Ajax. Program code is written by the programmer in Java. Whereas this may appear to be an obstacle to non-Java programmers, a large number of example programs and it is fairly easy to modify, without too much difficulty. The Google compiler takes the Java source code and converts it to a series of html and JavaScript files that can be displayed on most browsers.

The key advantage is the functionality of the toolkit, such as creating response to mouse movement, uploading and downloading files, series of widgets such as check boxes, selectable input boxes and trees.

A.5 Sesame

Sesame[36] is a Java open source framework for storing, querying and inferencing for RDF. Originally it was developed by Aduna as a research prototype for the EU research project On-To-Knowledge. Now, it is further developed and maintained by Aduna in cooperation with NLnet Foundation, developers from Ontotext, and a number of volunteer developers who contribute ideas, bug reports and fixes.

[34] http://www.w3.org/2004/OWL/

[35] http://code.google.com/webtoolkit/

[36] http://www.openrdf.org

Sesame has been designed with flexibility in mind. It can be deployed on top of a variety of storage systems (relational databases, in-memory, file systems, keyword indexers, etc.), and offers a large variety of tools to developers to leverage the power of RDF and RDF Schema, such as a flexible access API, which supports both local and remote (through HTTP or RMI) access, and several query languages, of which SeRQL is the most powerful one. It can be deployed as a web server or used as a Java library (API).

A.5.1 SeRQL

SeRQL[37] (Sesame RDF Query Language) is a new RDF/RDFS query language that is currently being developed by Aduna as part of Sesame. It combines the best features of other (query) languages like RQL, RDQL and adds some of its own. SeRQL is still in development and is now at revision 1.2. Some of SeRQL's most important features are:

- Graph transformation
- RDF Schema support
- XML Schema data type support
- Expressive path expression syntax
- Optional path matching
- Able to build both select and construct queries.

Some handy features which are still lacking, but are pending to be implemented:

- Aggregation operators (count, max, min, avg)
- SeRQL construct queries cannot construct resources if no explicit URI is provided

A possible query is to ask for every movie title with the substring “Pirates of the” and having a releasedate.

```
SELECT title,reldate
  FROM {m} movie:movieTitle {title}; movie:releaseBeginYear {date} time:inXSDDateTime {reldate}
  WHERE title LIKE "Pirates of the*"
  USING NAMESPACE
    movie = <http://wwwis.win.tue.nl/~ppartout/Blu-IS/Ontologies/IMDB/movie.rdfs#>,
    time = <http://www.w3.org/2006/time#>
```

[37] http://www.openrdf.org/doc/SeRQLmanual.html

Appendix B

Vocabularies

Searching for specific information about a movie in the absence of any form of metadata (information about the movie) is a hopeless task. A question is what kind of information about an object (movie) can help the user? A common case is where the user has seen the movie that is being sought once already, and so the user may remember details about it, such as words from the title, actors, or a genre. These clues can then be used to find the movie by searching using the clues and trying out different searches until the movie shows up. However it can also be the case that the user knows only a little.

Title	Pirates of the Caribbean: At World's End
Genre	Action / Fantasy
Subject	Piracy / Beard / Ship
Date	23 may 2007
Location	Bahamas / Athens / Maui, Hawaii, USA / Utah, USA
Actors	Johnny Depp / Orlando Bloom / Keira Knightley

- *Title*, the title of the movie usually offers good clues as to what the movie is about, but it does not necessarily mention all names of all subjects the user is interested in, and it may also presuppose knowledge the user does not actually possess.
- *Synopsis*, this field is likely to describe what the movie is about, but again may not facilitate search and discovery very effectively; for the same reasons that the title field may fail to do so.
- Subject, this field, which usually contains a set of keywords, is meant to convey precisely what the movie is about. However, much depends on how extensive the set of keywords is; whether all related subjects are mentioned, and whether too many subjects are listed, leading the user to get too many hits.
- Genre, is less precise than subject, however it can classify movie items very well.

The title, synopsis fields are very useful, as they help a user choose between the results. But they do not really facilitate in searching and discovering items effectively. Genre or subject are more effectively.

Assume that the genre or the subject property can have all kind of values.
The problem hereby is that the metadata is not necessarily saying very much about the object. A related problem is that making metadata describing the subject precisely, may also be difficult. It will be often the case that the most of the movies are not primarily about piracy, but about subjects related to piracy. However, if the movie metadata did not contain the keyword piracy, the particular movie can not been found. Another problem is that humans often define their own keywords, which means that the choice of keywords can be quite eclectic. Some people use abbreviations for a particular word, or type the word incorrectly. Different forms of the same keyword, or closely related keywords, are another issue.
Think about United States, United States of America and their acronyms US, USA.
Another problem is that keywords can refer to more objects (car as automobile, but also as railway car).
The conclusion that can be drawn from this is that having just a simple text field with no restrictions is not going to work very well in most cases. That's where vocabularies come into play.
Vocabularies are a set of indexing terms, or subjects used for classification. It is designed for classifying, indexing and searching for information resources. It avoids defining meaningless terms, terms which are too broad, or terms too narrow and to prevent misspelling and different forms of the same term. For example forcing that somebody writes United States. Essentially, if everyone uses the same name for the same concept, things become much easier to find.

A *taxonomy* is a subject-based classification that arranges the terms in the vocabulary into a hierarchy without doing anything further. In other words a structured list or tree formed into a hierarchy with broader terms at the top. Taxonomies were once used by biologists to classify living things into species, genera, families, etc. Ideally, each item in a taxonomy should be mutually exclusive and unambiguous, so if ‘mouse’ appears in one place referring to mammals, it should not turn up in another place referring to computer equipment. For each term the broader and narrower terms are indicated (vertical), and related terms can also be included (polyhierarchical). Note that the taxonomy helps by describing subjects; from the point of view of metadata there is really no difference between a vocabulary and taxonomy. The metadata only relates objects to subjects, whereas the subjects in a hierarchy are arranged. So a taxonomy describes the subjects being used for classification, but is not itself metadata; it can be used in metadata.

The set of terms being described is of course open, but the language used to describe them is closed, since it consists only of a single relationship (broader/narrower relationship used to build the hierarchy).

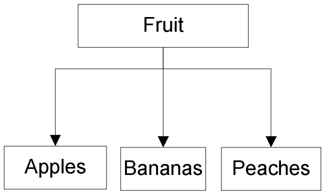


Figure 6.1: Taxonomy

A *thesaurus* is an ‘extended’ taxonomy, it shows connections in between (*related* and *see also* terms) has an underlying index showing words that might spring to mind but which you should not use (*nonpreferred* terms), and tells you what you should use instead (*preferred* terms). It can be polyhierarchical and usually contains *scope notes* to indicate exactly what the term means. A thesaurus lists concepts, the actual words used are not of crucial importance. While a taxonomy is designed to classify things, a thesaurus is designed to help you find the right words or phrases to describe what you are looking for. Thesauri provide a much richer vocabulary for describing the terms than taxonomies do, and so are much more powerful tools.

Thesaurus
Apples

Scope note: *The fruit of any member of the species Malus pumila*
Broader term: Fruits
Related terms: Cooking ingredients, Horticulture
Narrower term: Jonagolds
See also: Apple trees
Use: For Apple computers use Personal computers (Apple)

An *ontology* is a model for describing the world that consists of a set of types, properties, and relationship types. It is the most powerful vocabulary. They are used by people, databases, and applications that need to share information about a certain area. In ICT, ontologies are designed to allow computers to really interact with each other. For a WebCrawler to crawl around the web to find the best holiday, or get you a great deal on a computer, information has to be marked up in such a way that machines can understand it. However it can not distinguish the difference between a computer and a Jonagold if they are both referred as apples.

An ontology is more specific in defining a concept or item and its relationships. With ontologies the creator of the subject description language is allowed to define the language at will. A human will be a noun, and a mammal, and a man and possibly a person. Instead of having ‘babies’ as a narrower term of ‘humans’, it might have babies as offspring of humans. Ontologies often cover all elements of metadata, not just other subject terms. For example, a document could be connected to a person by the relationship ‘written by’, this person may then connect to an organisation by the relationship ‘works for’. This can lead directly from one document to others written by the same person, or by others working for the same organisation that the author works for.

B.1 IMDb

The Internet Movie Database[38] (IMDb) website offers well structured HTML content and they also offer an offline database, to download. This database can be used freely for research purposes. IMDb started as a Usenet newsgroup and they maintained a number of lists. A collection of shell scripts enabled people to search the group. Nowadays, the lists which can be downloaded from IMDb originate from these early days, of course with recent updates.

The database is broken up into lists and a list is not comprehensive. The Actors list contains the names of actors, the titles they were in and the names of characters they played, but not biographical information about them. Biographical information about the actors is in the Biographies list.

B.1.1 HTML version

IMDb offers a well structure HTML website to a user. Also the source of the HTML is structured properly with tags and styles. The search engine handles queries very well and it redirects to detail pages if there is a unique hit.

If somebody searches for the movie title "the gladiator", the system returns a list of movies, related or named "the gladiator". There is a main clustering of "Popular Results" and "Other Results". "Popular Results" contains Popular Titles. "Other Results" contains exact matches, partial matches and approximate matches. The presented result links can be used to parse the detail pages of the movies found. These links look like: `http://imdb.com/title/tt######/`
If somebody enters the movie title "Gladiator (2000)", the search engine recognizes the movie and redirects to the movie page.

A detail page contains the description of a movie, like a synopsis, production year, filming location, cast, duration, recommendations, photos, trailers and much more.
These descriptions are divided over a number of sub pages. These sub pages can easily be deduced from the link of the main detail page. The main link is extended with the sub page. Recommendations can be found at: `http://imdb.com/title/tt######/recommendations`.

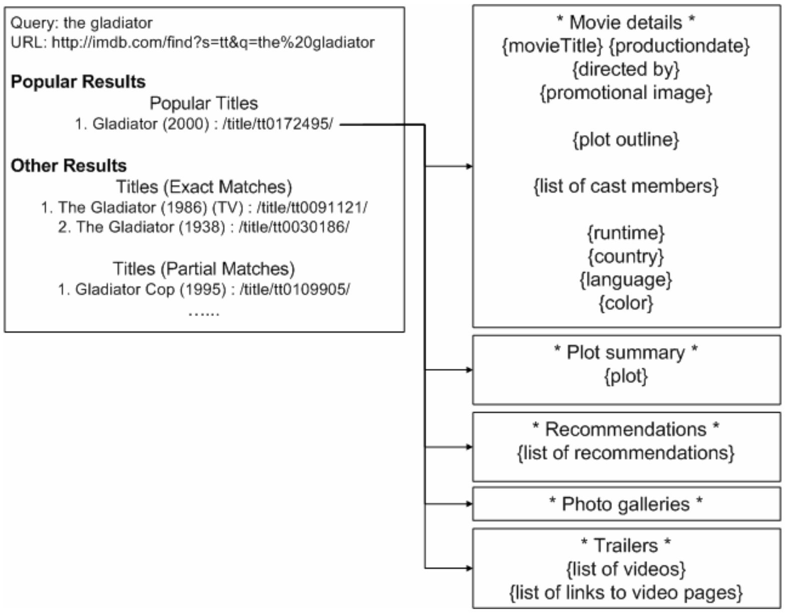


Figure B.1.2: IMDb movie page structure

[38] http://www.imdb.com

The search engine can also be used to return collaborators; actors, producers and directors. The same procedure as in the movie case is applied. The identifier looks like: `nm#######`

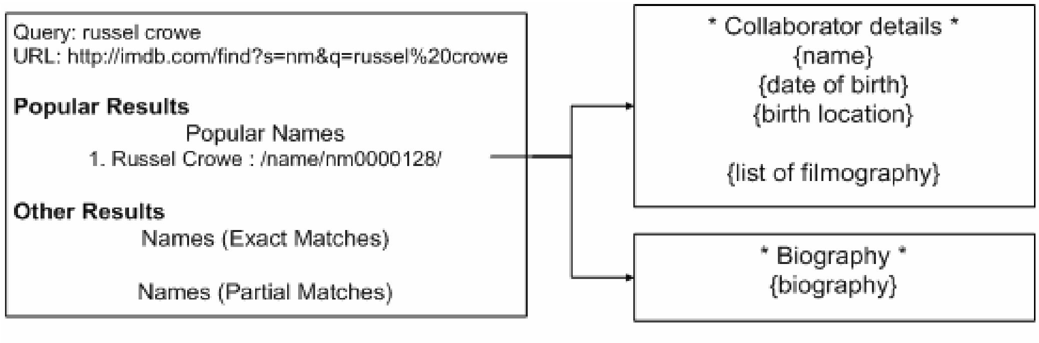


Figure B.1.3: IMDb person page structure

B.1.2 Offline database

The offline database files[39] [Appendix C] are generated by some scripts written by IMDb developers. The files are plain text and start in general with a kind of explanatory text, then fields separated by one or more tab characters. The files are located on 3 mirror ftp-sites, updated once a week.
The files can be found on:

- ftp.fu-berlin.de (Germany)
- ftp.funet.fi (Finland)
- ftp.sunet.se (Sweden)

They also provide files for an incremental approach. Once a developer loaded the complete database files it suffices to read the files with updates the next time. This will safe a lot of computation time.

This offline database does not contain unique identifiers like they are used in the website. Nevertheless, each movie title and person name in the offline database is unique. If there are two movies produced in the same year with the same title, the movie titles are extended with a Roman number (I,II,III,IV,etc). In the person case, ambiguous names are also avoided by Roman number extension.
Example is: `Smugglers, The (1916/I) – Smugglers, The (1916/II)`

A lot of data can be retrieved from these offline databases, but they are not complete. Trailer, photos, recommendations are not stored in the offline databases. A solution is to use the unique title or name and put this identifier in the search field of the IMDb search engine. The search engine will give a redirection in 95% of the cases to a detail page of a movie or person. There are some person names or movie titles which are not recognized as completely unique. Having this detail page it is easy to transform it to a sub page containing trailers or photo galleries.

B.1.3 IMDb Photos

Each movie has a dedicated photo gallery page. The link of a photo page can easily be deduced from the main link. `http://imdb.com/title/tt0172495/` → `http://imdb.com/gallery/ss/0172495`

The photos are divided over a number of photo gallery pages. The link to the next photo page uses an offset to indicate the start of the new set. The structure of the page remains the same.
A recursive implementation can easily process this kind of structures.

IMDb also offers photo galleries showing persons. These galleries are very similar to the movie case.

[39] http://www.imdb.com/interfaces

B.1.4 IMDb scheme

The IMDb scheme was created after studying many web pages describing movies on the IMDb website and studying the (offline) text dump of the database. In the thesis it was attempted to design a detailed ontology for IMDb. The scheme provides the most relevant information from IMDb and enables the syndication with possible other domains, like time, genre, and geography. It describes the fundamental concepts involved in IMDb, including classes such as movies, persons, actors, directors, genres, languages, countries, keywords, etc., and the relations among them.

The movie concepts are derived from the files in [Appendix C].
The scheme is divided over a number of sub schemes for presentation purposes. The movie scheme is model of all the metadata describing a movie, person is a model of the person metadata etc. Note, that there are relations between the different schemes. In the movie scheme a lot of fields are included; the most of them are represented by simple data types. Some of them are picked out, for clarification purposes.

- imdb:movieGenre
  The *movieGenre* property described the genres of the movie in question. Originally IMDb uses just simple strings, like "action". In this scheme *movieGenre* is mapped to the TV-Anytime Genre classification. Thus now the simple string is replaced by a genre in the genre hierarchy of TV-Anytime.
- imdb:filmingLocation
  The filmingLocation is described as a *geo:Location*. It refers to a Location in the Geo vocabulary.
- imdb:productionStartDate
  ProductionStartDate is an example of using OWL-Time. The date from IMDb is mapped to an OWL-time *Instant*. Such an Instant describes datetime concepts very extensively.
- imdb:involvedCompany
  The *imdb:involvedCompany* property is described by a Company, which is also defined in the IMDb scheme. If in a movie a number of companies are involved, these companies are all instances of Company.
- imdb:Episode
  Episode is a subClassOf of Movie. It inherits all the properties a movie has, but extends it with some properties only an Episode can have. Also the relation between a movie and episode is modelled. A main series is modelled like a Movie and the episodes from that series, are related via imdb:hasEpisode.

In the person scheme two relations must be explained. The first relation is *engagedIn*. If two persons are married to each other, then they have both a Marriage instance, with the same resource (URI). Thus, the *imdb:engagedIn* for Person1 and Person2, refers to the same Marriage instance. This means that they are married.

The second is the relation between Person, Role and Movie. A Person can play in a movie, or directs a movie or is just a person which is defined in IMDb but does not collaborate in movies, is just the "husband of". A person who has a role has a relationship with a subClassOf Role. Role itself is just an abstract term. All the subclasses inherit the properties from the abstract Role. If a person has a role in a movie as actor, then the instance of the role is of type Actor (Actor1) and a Person instance (Person1) is related via hasRole. The same person (again Person1) can also direct a movie. Then the instance of the role is of type Director (Director1). The actor and director are part of a movie, this is modelled by inMovie. Thus, Actor1 is inMovie (Movie1) and Director1 is inMovie (Movie2). If the Person plays another actor role in another movie, Person1 hasRole Actor2 inMovie Movie2. Each property of Actor1, Actor2 and Director2 depends on the unique role a Person can play in a specific movie; it is possible to describe a salary which dependents on the role of the Person has played in a Movie.

In the company scheme the same reasoning can be used as in the Person case. Person, Role, Movie are now replaced by Company, Function, Movie. A company has a certain function, Function is again an abstract term, and this function is done in a specific Movie.

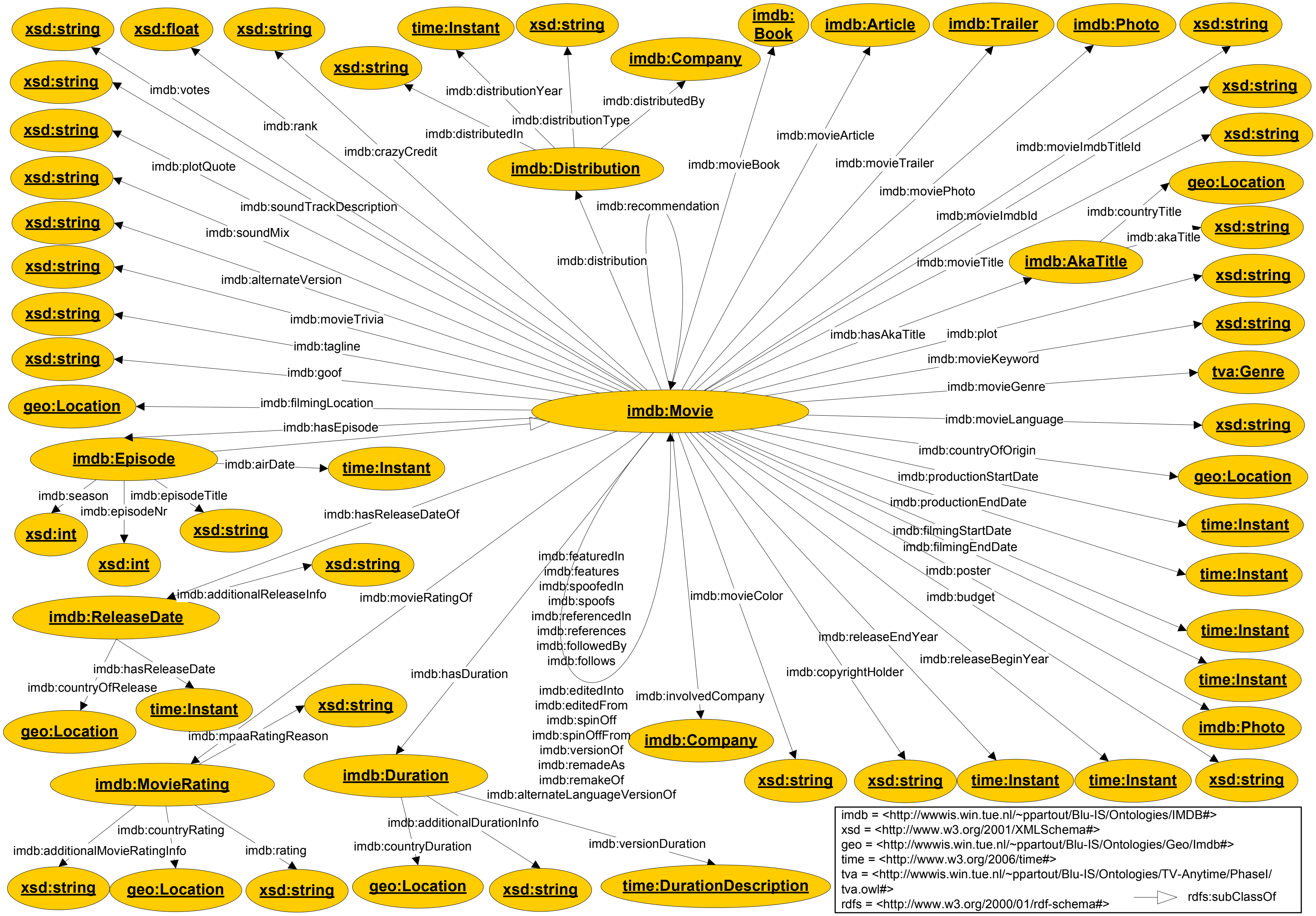


Figure B.1.4: IMDb Scheme – Movie

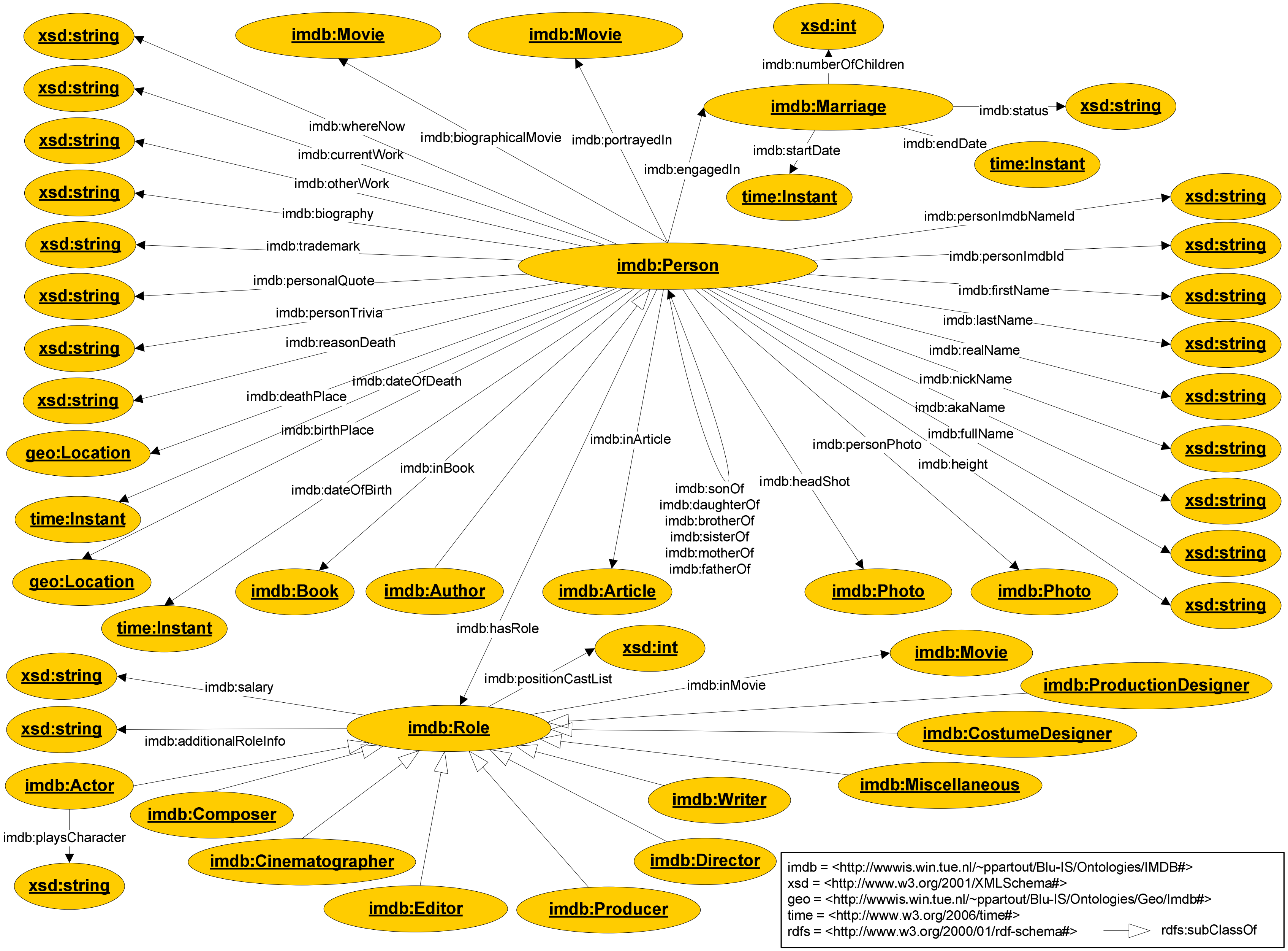


Figure B.1.5: IMDb Scheme – Person

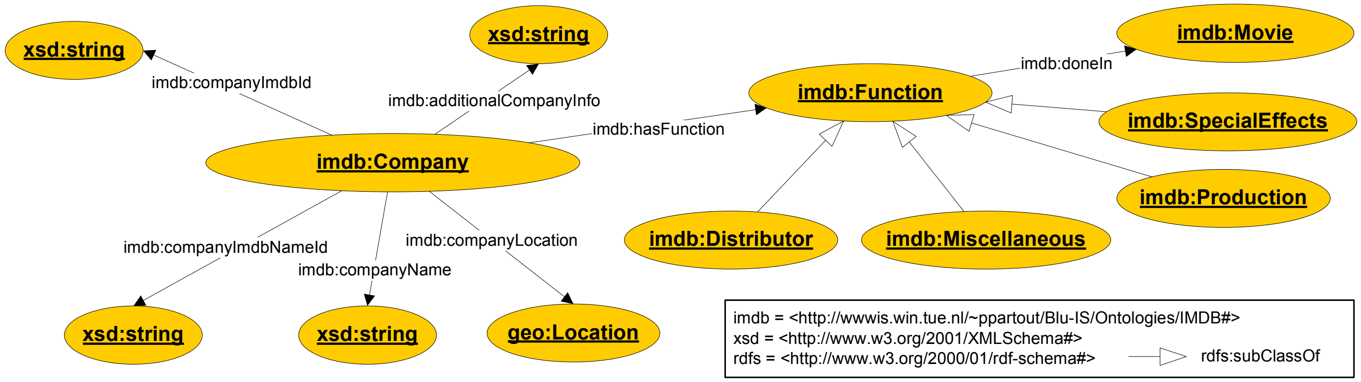


Figure B.1.6: IMDb Scheme – Company

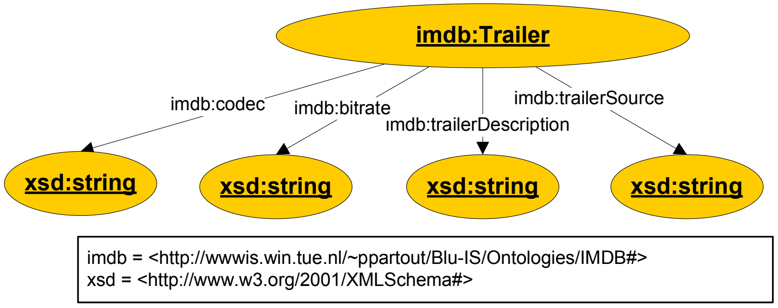


Figure B.1.7: IMDb Scheme - Trailer

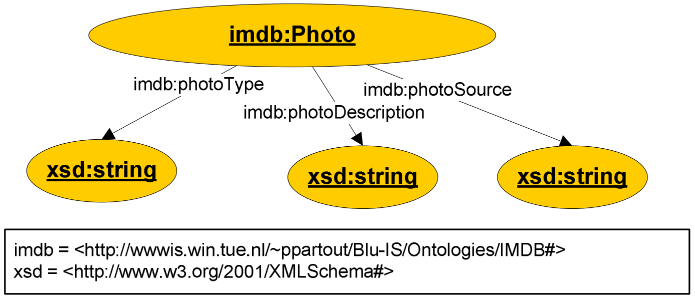

imdb:Photo
imdb:photoType
imdb:photoDescription
imdb:photoSource
xsd:string
xsd:string
xsd:string
imdb = <http://wwwis.win.tue.nl/~ppartout/Blu-IS/Ontologies/IMDB#>
xsd = <http://www.w3.org/2001/XMLSchema#>

Figure B.1.8: IMDb Scheme - Photo

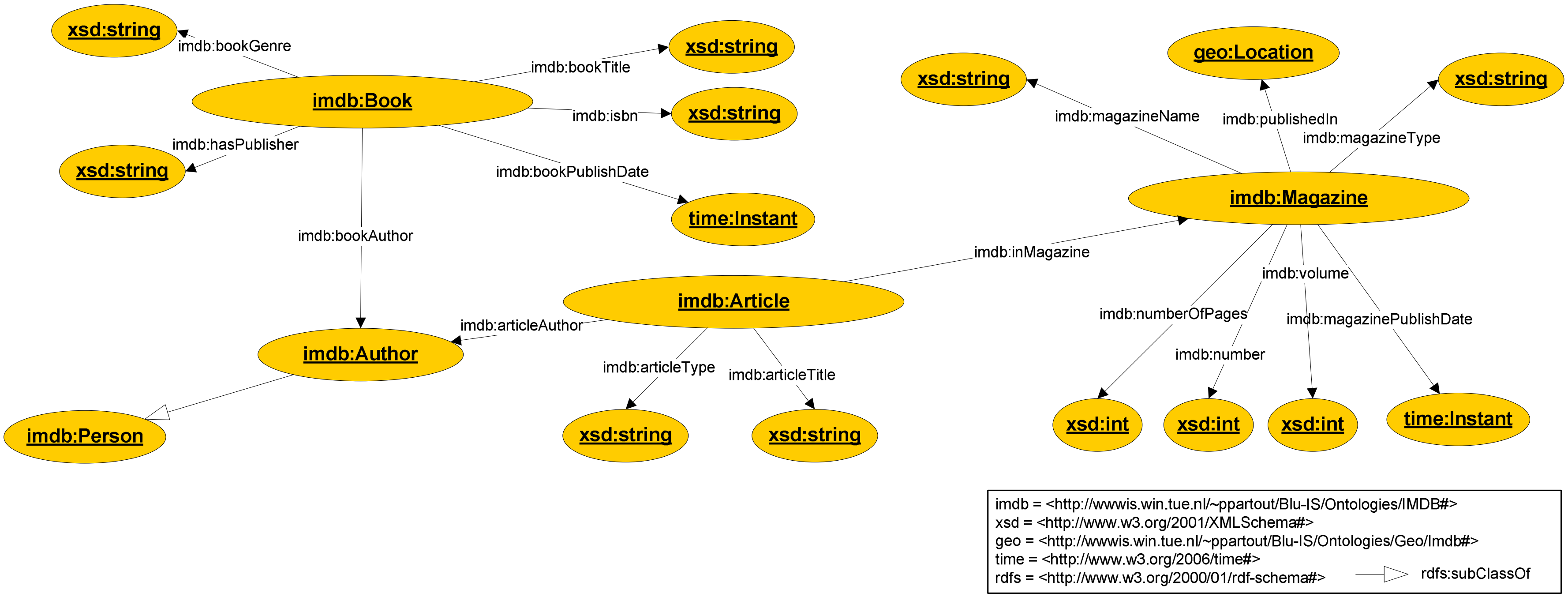


Figure B.1.9: IMDb Scheme - Literature

B.2 Trailers

Trailers can be used to enrich the movie information and to attract a user. Nowadays, each new released movie has also a webpage, including trailers. But these dedicated websites are difficult to grab, because the different structures and URLs of the movie websites. It is not easy to find the exact location of a trailer. There are also big movie databases available, which provide movie related information and trailers. The majority of these providers are commercial organizations. They provide the trailers together with advertisements. Unravelling these kinds of packages is very time consuming and sometimes complex.

B.2.1 Videodetective

Videodetective[40] claims to be the biggest provider of movie trailers. It contains trailers from theatrical releases, TV and DVD releases. The exact size of the database is unknown, but the roughly estimated number is between 100.000 and 200.000 items.

The open interface is a web service[41] returning a XML document. The service is just a URL with some predefined variables to be set (e.g. search term, number of results). The returned XML document contains the found items plus metadata describing the movie. A disadvantage of the service is that it does not allow a great amount of requests in a short time. The interface also does not allow methods to retrieve all the items at once in a simple way. For example it is not possible to retrieve all the trailer links from the movies starting with a letter (e.g. A). Therefore it is quite complicated to copy the entire database and to convert it completely in another format. The request of a movie title is not a problem at all, that is what the service basically provides.

The returned XML document contains a set of items. Each item contains a number of properties describing the trailer. The most important properties are:

- Title: title of the movie
- Actor(1-6): Names of the actors
- Director: Name of director
- PublishedId: An identifier referring to a trailer. This ID must be used to initialize the trailer player.

```
<item>
  <Description>GLADIATOR featuring Richard Harris.</Description>
  <Title>GLADIATOR</Title>
  <Studio>Dreamworks</Studio>
  <StudioID>253</StudioID>
  <DvdReleaseDate></DvdReleaseDate>
  <TheatricalReleaseDate></TheatricalReleaseDate>
  <Actor1>Djimon Hounsou</Actor1>
  <Actor2>Joaquin Phoenix</Actor2>
  <ActorId2>9612</ActorId2>
  <Actor3>Russell Crowe</Actor3>
  <ActorId3>7349</ActorId3>
  <Actor4>Tomas Arana</Actor4>
  <ActorId4>8188</ActorId4>
  <Actor5>Richard Harris</Actor5>
  <ActorId5>6873</ActorId5>
  <Actor6></Actor6>
  <Director>Ridley Scott</Director>
  <DirectorID>6958</DirectorID>
  <Link>http://www.videodetective.com/trailer-preview.asp?publishedid=814369</Link>
  <PublishedId>814369</PublishedId>
  <Duration>226</Duration>
  <DateCreated>3/31/2000 5:21:00 PM</DateCreated>
  <DateModified>3/31/2000 5:21:00 PM</DateModified>
</item>
```

[40] http://www.videodetective.com

[41] http://www.videodetective.com/xmlapi.aspx

B.3 XMLTV

XMLTV[42] is a set of utilities to process TV (guide) listings. These listings are stored in an XML format. A set of grabbers is included in the package to grab the data from the different TV guide sources. For example the Dutch listing is grabbed from tvgids.nl, one of the English listings comes from a machine-readable file produced by the Radio Times website. There are much more grabbers available, for countries like Germany, Finland, Spain, Hungary, Denmark, France, USA and Japan.

The grabbers are configurable. It is possible to set the time span for retrieving listings or a selection of broadcast channels. In the end it produces an XML file containing information about the schedule. The format produced differs from most other XML-based TV listings in that it is written from the user's point of view, rather that the broadcaster's. It does not divide listings into channels. The channels are stored into a single unified listing. Each item has details like a name, description, production date and a credit list. There is support for listings in multiple languages and each item can have "language" and "original language" details.

```
<tv source-info-url="http://www.tvgids.nl/" source-data-url="http://www.tvgids.nl/" generator-info-
name="XMLTV" generator-info-url="http://membled.com/work/apps/xmltv/">
  <channel id="7.tvgids.nl"><display-name>RTL 4</display-name></channel>
  <channel id="24.tvgids.nl"><display-name>Film1.1</display-name></channel>
  <channel id="25.tvgids.nl"><display-name>PRO 7</display-name></channel>
  <programme start="20061212203000 +0100" stop="20061212223000 +0100" channel="24.tvgids.nl">
       <title lang="nl">Fantastic Four</title>
       <category lang="nl">Film</category>
       <url>http://www.tvgids.nl/programmadetail/?ID=5969850</url>
  </programme>
  <programme start="20061213224000 +0000" stop="20061214003500 +0000" channel="25.tvgids.nl">
       <title>The Thomas Crown Affair</title>
       <date>1999</date>
  </programme>
</tv>
```

It is possible to generate more complex listings, giving additional information about channels (instead of the listing of identifiers like "bbc.com"), picture and sound, subtitles, age ratings and more. The complete specification is described in the XMLTV DTD[43].

B.3.1 IMDB grabber

The XMLTV package contains also an IMDb grabber. The returned results contain only little information; limited number of actors, director, production year and genre. It is necessary to download the database of IMDb and it must be indexed. The developer of the IMDb grabber script also uses the offline database (B.1.2), but it returns very limited metadata. If the grabber finds a match it adds the following:

1. Exactly title from IMDb
2. If the match is a movie, the date field is set to 4 digit year of production
3. The type of match found (Movie, TV Movie, Video Movie, TV Series, or TV Mini Series) is placed in the categories field
4. The URL to the IMDb page is added
5. The director is added if the match was a movie or if only one director is listed in IMDb (because some TV series have > 10 directors)
6. The top 3 billing actors are added.
7. Genres added to 'categories' field
8. IMDb user ratings added to 'star-ratings' field

[42] http://xmltv.org/wiki/

[43] http://xmltv.cvs.sourceforge.net/xmltv/xmltv/xmltv.dtd?view=markup

An example of XMLTV including IMDb metadata is presented below.

```
<programme   start="20050118010500   +0100"   stop="20050118025500   +0100"   channel="7.tvgids.nl"
clumpidx="0/1">
        <title>Blow Out</title>
        <desc lang="nl">thriller Een geluidsman neemt per ongeluk belangrijk bewijsmateriaal op als
        een senator met zijn auto in de rivier belandt en omkomt.</desc>
        <credits>
                <director>Brian De Palma</director>
                <actor>John Travolta</actor>
                <actor>Nancy Allen</actor>
                <actor>John Lithgow</actor>
        </credits>
        <date>1981</date>
        <category lang="en">Movie</category>
        <category lang="en">Thriller</category>
        <url>http://us.imdb.com/M/title-exact?Blow%20Out%20%281981%29</url>
        <video><aspect>16:9</aspect></video>
        <subtitles type="teletext" />
        <star-rating><value>7.0/10</value></star-rating>
</programme>
```

B.3.2 Scheme

The XMLTV scheme contains a few main concepts, namely *xmltv:TV*, *xmltv:Programme* and *xmltv:Channel*. The first integrates the whole; it contains the channel information and the descriptive metadata about a Programme. Programme and Channel have a relation to enable Programmes to point to a specific channel. This relation makes it possible to identify that a programme at a specific time, on a specific channel is broadcasted.

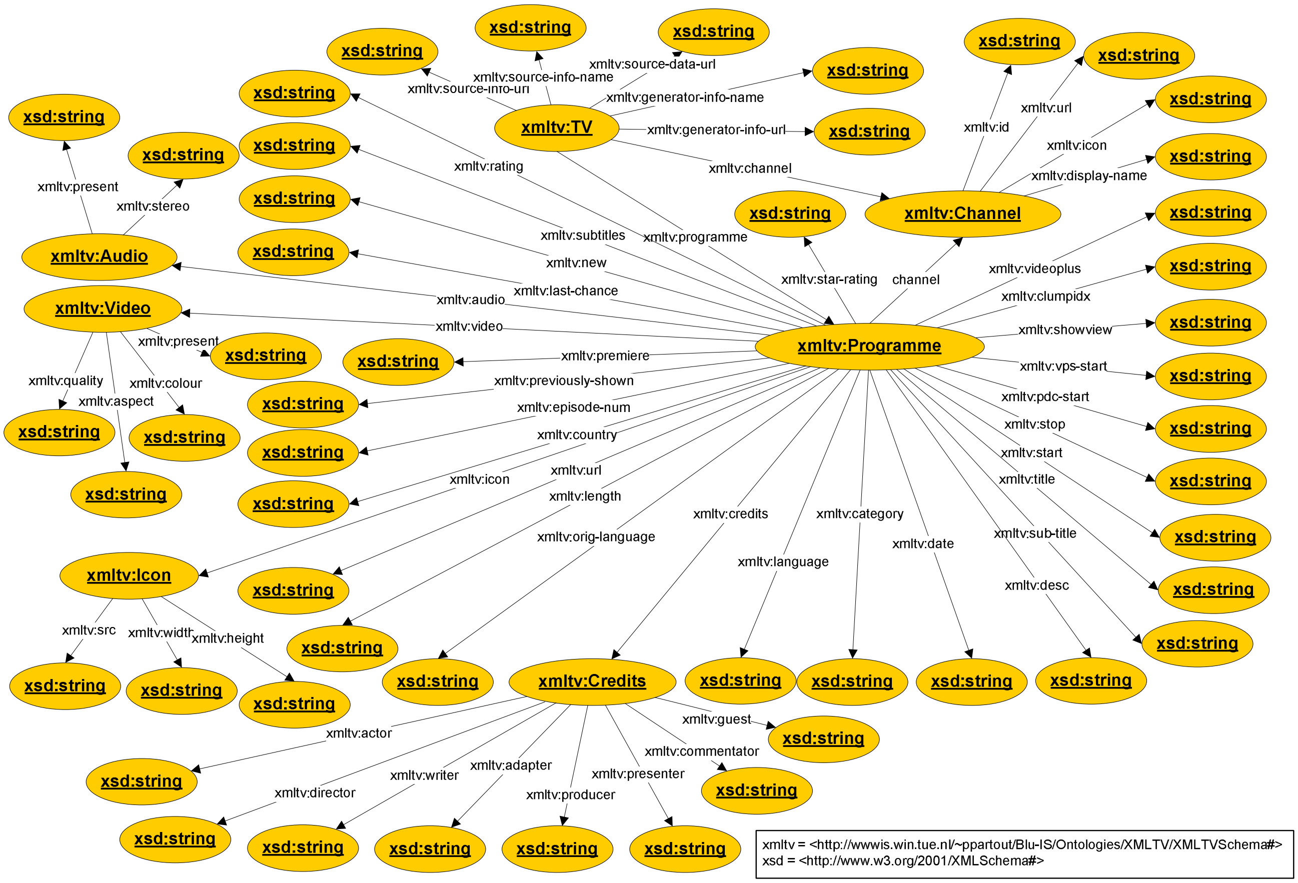


Figure B.3.1: XMLTV Scheme

B.4 BBC Backstage

The BBC backstage[44] is the BBC's developer network to encourage innovation in TV. Content feeds[45] are available for people to build with on a non-commercial basis. The rich programme information from BBC broadcasts can be retrieved directly from BBC Backstage in TV-Anytime XML format. It contains 7 days (today + next 6 days) of 20 channels BBC TV and Radio schedule information. For each channel there are 3 files per day in the data set. One file contains the ProgramLocation, the second the ProgramDescription and the last one the ContentReference.

B.4.1 ProgramLocation

The ProgramLocation table contains the time and channel information. It provides the promotional information to describe where and when the content can be found. The obvious use for these files is in the creation of a traditional channel/time Electronic Programme Guide. Read Appendix B.5 for more information about the TV-Anytime Specification.

```
<TVAMain>
  <ProgramDescription>
    <ProgramLocationTable>
      <Schedule serviceIDRef='BBCOne' start='2007-09-01T00:00:00Z' end='2007-09-01T23:59:59Z'>
        <ScheduleEvent>
          <Program crid='crid://bbc.co.uk/272893884'/>
          <ProgramURL>dvb://233a.1004.1044;7bc@2007-09-01T00:41:31Z/PT00H03M</ProgramURL>
          <PublishedStartTime>2007-09-01T00:40:00Z</PublishedStartTime>
          <PublishedDuration>PT00H05M00S</PublishedDuration>
        </ScheduleEvent>
        <ScheduleEvent>
          <Program crid='crid://bbc.co.uk/272894105'/>
          <ProgramURL>dvb://233a.1004.1044;899@2007-09-01T19:00:00Z/PT02H14M</ProgramURL>
          <PublishedStartTime>2007-09-01T19:00:00Z</PublishedStartTime>
          <PublishedDuration>PT02H15M00S</PublishedDuration>
        </ScheduleEvent>
      </Schedule>
    </ProgramLocationTable>
  </ProgramDescription>
</TVAMain>
```

[44] http://backstage.bbc.co.uk/

[45] http://backstage.bbc.co.uk/feeds/tvradio/

B.4.2 ProgramDescription

The ProgramDescription file contains the metadata of the programme being described. Examples would be title, synopsis, or whether it's subtitled.

```
<TVAMain>
  <ProgramDescription>
    <ProgramInformationTable>
      <ProgramInformation programId='crid://bbc.co.uk/272894105'>
        <BasicDescription>
          <Title><![CDATA[Eurovision Dance Contest]]></Title>
          <Synopsis length='short'><![CDATA[16 countries from across the continent fight it out to be
          crowned the champions of Europe. [S]]]></Synopsis>
          <Genre href='urn:tva:metadata:cs:IntentionCS:2004:1.1'>
            <Name><![CDATA[Entertainment]]></Name></Genre>
          <Genre href='urn:tva:metadata:cs:FormatCS:2004:2.4'>
            <Name><![CDATA[Show]]></Name></Genre>
          <Genre href='urn:tva:metadata:cs:ContentCS:2004:3.5.2'>
            <Name><![CDATA[Quiz/Contest]]></Name></Genre>
          <Genre href='urn:tva:metadata:cs:IntentionCS:2004:1.2'>
            <Name><![CDATA[Information]]></Name></Genre>
          <Genre href='urn:tva:metadata:cs:ContentCS:2004:3.1'>
            <Name><![CDATA[Non-fiction]]></Name></Genre>
          <Genre href='urn:tva:metadata:cs:ContentCS:2004:3.5'>
            <Name><![CDATA[Amusement]]></Name></Genre>
          <CaptionLanguage closed='true'>EN-UK</CaptionLanguage>
        </BasicDescription>
        <AVAttributes>
          <AudioAttributes>
            <NumOfChannels>2</NumOfChannels>
          </AudioAttributes>
          <VideoAttributes>
            <AspectRatio>16:9</AspectRatio>
          </VideoAttributes>
        </AVAttributes>
        <MemberOf xsi:type='MemberOfType' crid='crid://bbc.co.uk/__SEREurovisionDanceContest'/>
      </ProgramInformation>
    </ProgramInformationTable>
  </ProgramDescription>
</TVAMain>
```

B.4.3 ContentReferencingTable

The ContentReferencingTable file contains the precise location(s) where an item of content can be found. In the general case a CRID *resolves* to one or more CRIDs or Locators. In theory other locators such as streams or file locations could be provided in this table, but this is not covered in the current data sets.

```
<ContentReferencingTable>
  <Result CRID='crid://bbc.co.uk/272894105' status='resolved' complete='true' acquire='any'>
    <LocationsResult>
      <Locator>dvb://233a.1004.1044;899@2007-09-01T19:00:00Z/PT02H14M</Locator>
    </LocationsResult>
  </Result>
</ContentReferencingTable>
```

B.5 TV-Anytime

The TV-Anytime[46] forum is an organization established in 1999 to develop a specification of persistent audio and video or another kind of services on a consumer media platform. TV-Anytime defines the meta-data format so that meta-data can be exchanged between various entities such as between the content provider and consumer, among consumers, or between a third-party meta-data provider and the consumer.

For the purpose of interoperability, the TV-Anytime Forum has adopted XML to represent the meta-data. 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 meta-data schema is the formal definition of the structure and type of metadata. TV-Anytime uses the MPEG-7 Description Definition Language (DDL) to describe meta-data 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.

In the context of TV-Anytime the meta-data is descriptive data about content such as a programme title or synopsis. This meta-data is called an attractor because on the basis of this information the end-user will decide whether or not to acquire a particular piece of content. TV-Anytime meta-data is divided into two phases mainly Phase-1 and Phase-2.

TV-Anytime Phase-1 [8] is the set of specifications that enable various operations on the content from both broadcast and online media. Meta-data specification addresses a data model that allows a broadcaster or consumer to describe the content available within the broadcast system. An example ("Gladiator") of the metadata is described below.

TV-Anytime Phase-2 [9] is the extension of Phase-1. Few new concepts like Packages, Targeting (Automatically matching and delivering the content to the profiled consumer), and Re-distribution (sharing the content in and around the system) were added to Phase-2. Phase-2 specification is backward compatible with that of phase 1 so all the elements in phase 1 are still present in the Phase-2 but new elements were added.

In order to integrate the domain specific knowledge into the TV-Anytime meta-data specification and to share the common structure easily with other applications, the XML schemes are translated into an OWL ontology.

[46] http://www.tv-anytime.org/

```
<tva:ProgramDescription>
    <tva:ProgramInformationTable>
      <tva:ProgramInformation programId="crid://tue.nl/imdb/Gladiator/95975749770711">
       <tva:BasicDescription>
          <tva:MediaTitle>
            <mpeg7:TitleImage>
             <mpeg7:MediaUri>http://ia.ec.imdb.com/media/imdb/01/I/59/96/06m.jpg</mpeg7:MediaUri>
            </mpeg7:TitleImage>
          </tva:MediaTitle>
         <tva:Title type="main"><![CDATA[Gladiator]]></tva:Title>
        <tva:Synopsis><![CDATA[ When a Roman general is betrayed and his…]]></tva:Synopsis>
        <tva:Genre href="urn:tva:metadata:cs:ContentCS:2002:3.4.6" type="main"></tva:Genre>
        <tva:Genre href="urn:tva:metadata:cs:ContentCS:2002:3.4.6.1" type="main"></tva:Genre>
        <tva:Language><![CDATA[EN]]></tva:Language>
        <tva:CaptionLanguage closed="true" supplemental="false">nl</tva:CaptionLanguage>
        <tva:CreditsList>
          <tva:CreditsItem role="urn:mpeg:mpeg7:cs:RoleCS:2001:ACTOR">
           <tva:PersonName>
             <mpeg7:GivenName>Russell Crowe</mpeg7:GivenName>
           </tva:PersonName>
          </tva:CreditsItem>
          <tva:CreditsItem role="urn:mpeg:mpeg7:cs:RoleCS:2001:ACTOR">
           <tva:PersonName>
             <mpeg7:GivenName>Joaquin Phoenix</mpeg7:GivenName>
           </tva:PersonName>
          </tva:CreditsItem>
          <tva:CreditsItem role="urn:mpeg:mpeg7:cs:RoleCS:2001:DIRECTOR">
           <tva:PersonName>
             <mpeg7:GivenName>Ridley Scott</mpeg7:GivenName>
           </tva:PersonName>
          </tva:CreditsItem>
        </tva:CreditsList>
        <tva:ProductionDate>
          <tva:TimePoint><![CDATA[2000]]></tva:TimePoint>
        </tva:ProductionDate>
        <tva:Duration>PT2H35M</tva:Duration>
      </tva:BasicDescription>
       <tva:AVAttributes>
       <tva:VideoAttributes>
          <tva:Color type="color"/>
       </tva:VideoAttributes>
     </tva:AVAttributes>
    </tva:ProgramInformation>
   </tva:ProgramInformationTable>
  </tva:ProgramDescription>
```

B.5.1 TV-Anytime Phase-1

TV Anytime phase 1 [8] is a broadcast-oriented metadata specification. In TV-Anytime phase 1each programme (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 CRIDs, 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. <data> is a free format string that is URI 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 scenarios 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 programme resolves into a DVB-locater which indicates the channel and time of the broadcasted programme. To record that specific programme, it is not the DVB-Locator that is used to programme the event into the PVR, but the CRID of the programme. 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:

- 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.
- GroupInformationTable: Contains descriptions of groups of related items of content, i.e. all episodes of “Friends”.
- 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.
- ServiceInformationTable: Contains information about a specific broadcast channel.
- CastMemberInformationTable: A mapping of cast members to unique identifiers. The identifiers can be used in other metadata Instances making searching easier.
- ProgramReviewTable: Contains reviews for content elements.
- 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.

B.5.2 TV-Anytime Phase-2

The Phase 2 TV-Anytime Metadata Schema [9] 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.

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 does not contain the resource itself, but has a reference to the content (a CRID).

```
<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>
```

B.5.3 Classification schemes

The Classification Scheme is an MPEG-7 tool for the provision of controlled terminology for use in classification. An informative set of Classification Schemes has been developed by TVA to provide a universally applicable default set of classification terms.

TV-Anytime programme classification is based on these dimensions:

- Intention
- Format
- Content
- Content Commercial
- Origination
- Intended Audience
- Language
- Content Alert
- Media Type
- Atmosphere
- Audio Purpose
- Purchase Type
- Unit Type

For each of these dimensions, a TVA at least one default hierarchical CS of up to 3 levels has been defined.

- *ActionType CS*: types of user action being monitored to analyse viewing habits.
- *HowRelated CS*: a series of definitions for possible relations between programmes.
- *TVARoleCS*: key cast roles.
- *RoleCS*: is a classification scheme from MPEG-7 import by the TVARoleCS.
- *IntentionCS*: Contains terms that describe the intention of the programme at the conceptual phase, e.g. entertainment, information, education. This information can be useful for the classification of early transmitted "attractors".
- *FormatCS*: Contains terms that describe the format of a programme, e.g. artistic performance, a magazine show, a cartoon.
- *ContentCS*: Contains terms that describe the nature and and/or subject of the programme or commercial, e.g. soccer.

- *ContentCommercialCS*: Contains terms that describe the nature and and/or subject of a commercial.
- *OriginationCS*: Contains terms that describe the origination of the programme (e.g. live from a studio, a cinema release movie etc.).
- *IntendedAudienceCS*: Contains terms that describe the intended audience for the programme (e.g. by gender, age, socio-economic group or educational level).
- *LanguageCS*: Contains terms that describe the language of the intended audience for the programme.
- *Content AlertCS*: Provides a means of alerting a viewer that there are elements within the programme they may not wish to be exposed to, i.e. a "detractor". Strong language, scenes of a sexual nature etc.
- *Media TypeCS*: Contains terms that describe the medium of the content (e.g. Video and Audio, a multimedia application, audio only, audio video enhancements, etc.)
- *AtmosphereCS*: Contains terms that convey the psychological or emotional ("soft") characteristics of a content item.
- *AudioPurposeCS*: Contains terms that describe the main purpose for the audio track to be available.
- *PurchaseTypeCS*: Contains three basic modes of purchase: forever, for limited period of time, for a limited number of plays
- *UnitTypeCS*: Defines quantity units to further qualify a purchase type in "days", "months" or "years" for a period of time and "plays" for a number of plays.

B.5.4 Scheme

Note that the schemes are the adjusted version of TV-Anytime, including OWL-Time and Geo. The blue colored/dark colored nodes are the adjusted parts. The *ProgramDescription* integrates ProgramInformation and ProgramLocation and contains all the metadata that is needed to locate content and show promotional information. The *ProgramDescription* contains the metadata of the programme being described. Examples would be title, synopsis, or whether it's subtitled. The *ProgramLocation* contains the time and channel information. It provides the promotional information to describe where and when the content can be found. A programme location contains information about one instance of a programme. Multiple programme locations from the same broadcast service provider can be grouped to form a schedule. A metadata provider aggregates a set of programme locations (e.g. schedules) into a *ProgramLocationTable*. The typical use is in the creation of a traditional channel/time Electronic Programme Guide.

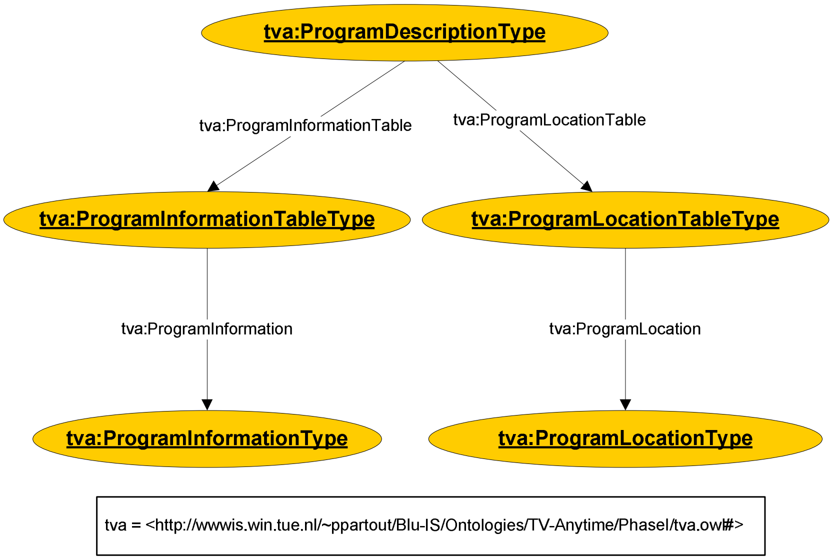

tva:ProgramDescriptionType
tva:ProgramInformationTable
tva:ProgramLocationTable
tva:ProgramInformationTableType
tva:ProgramLocationTableType
tva:ProgramInformation
tva:ProgramLocation
tva:ProgramInformationType
tva:ProgramLocationType
tva = <http://wwwis.win.tue.nl/~ppartout/Blu-IS/Ontologies/TV-Anytime/PhaseI/tva.owl#>

Figure B.5.1: TV-Anytime Scheme - ProgramDescription

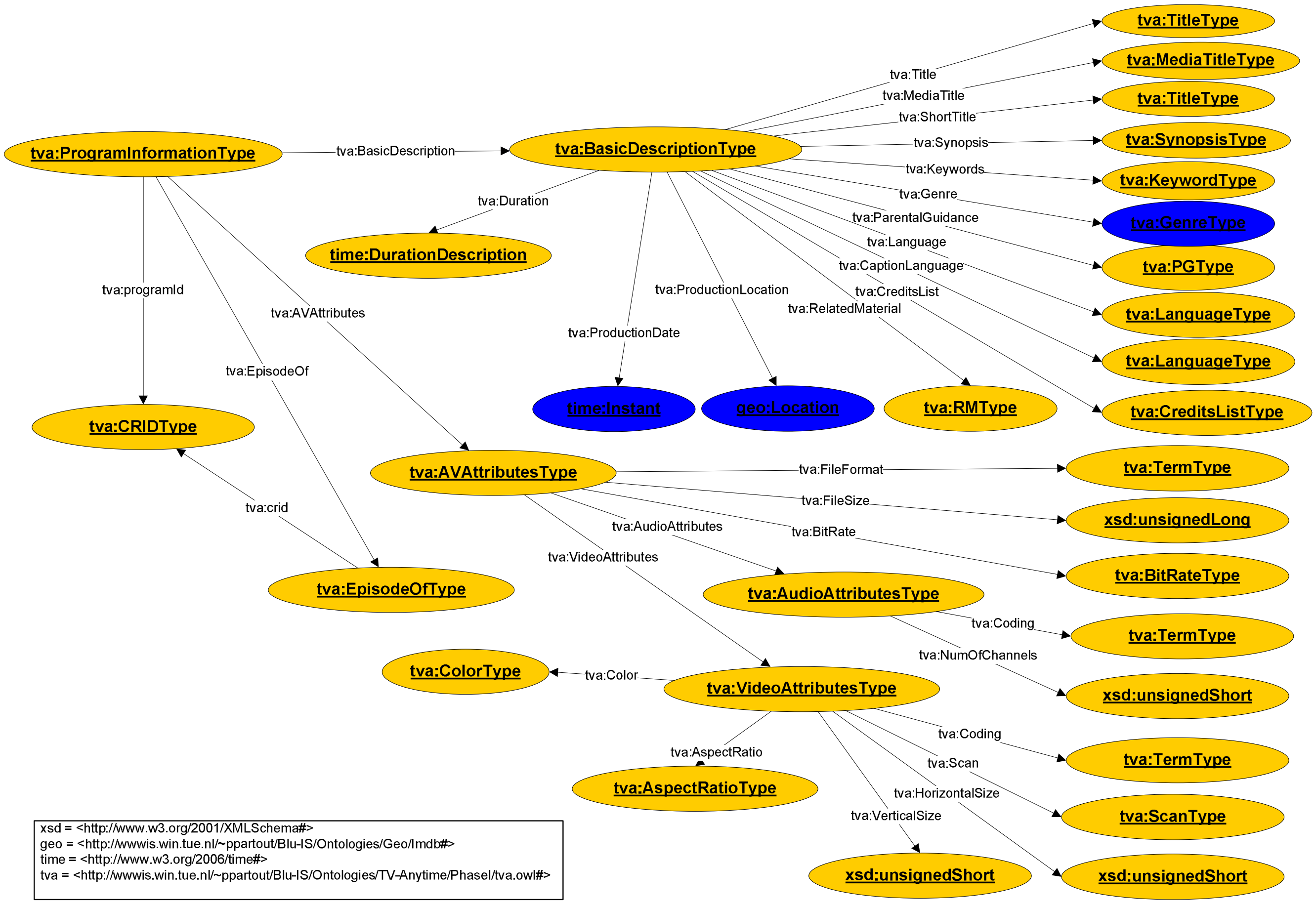


Figure B.5.2: TV-Anytime Scheme - ProgramInformation

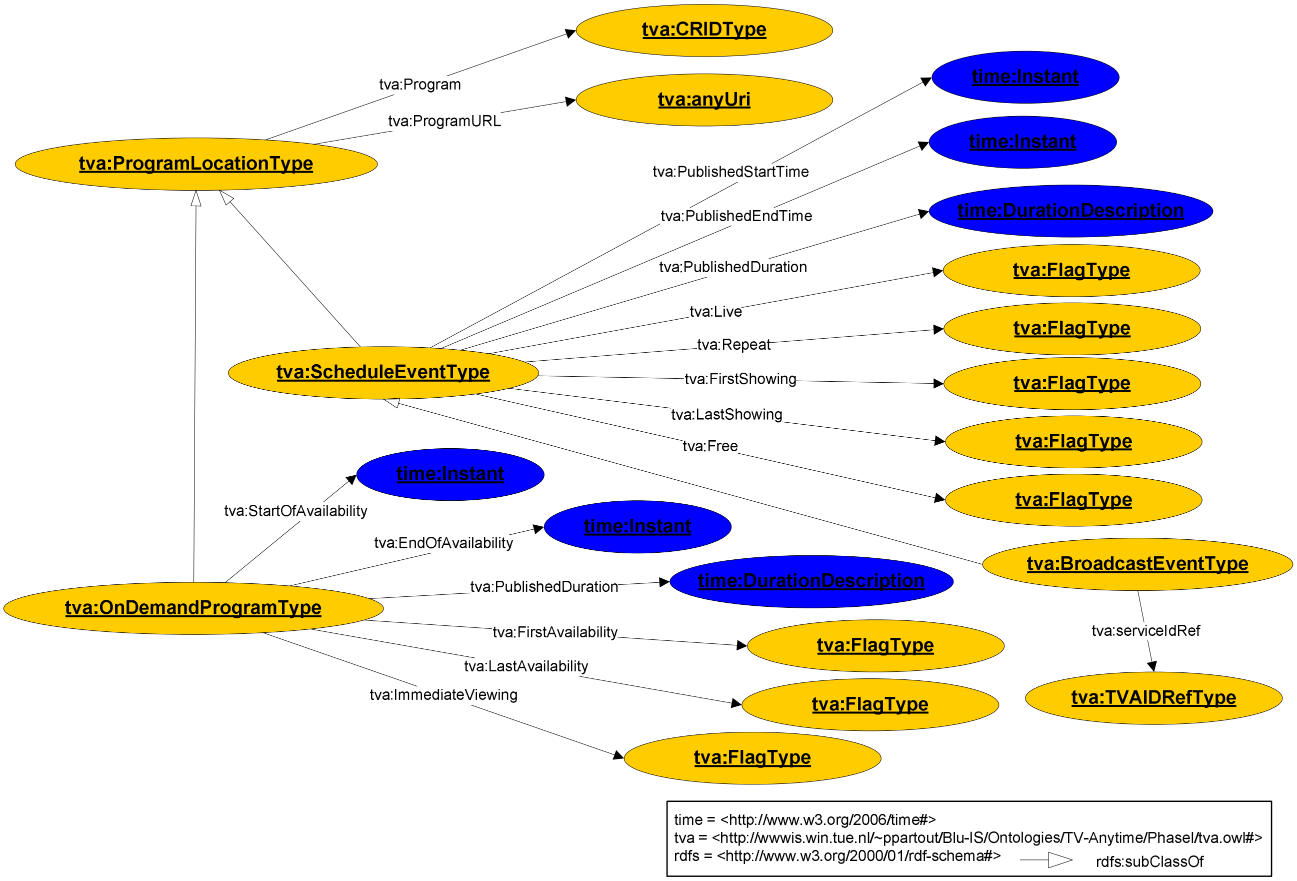


Figure B.5.3: TV-Anytime Scheme - ProgramLocation

ScheduleEvent: is a specific type of a programme location for describing broadcast programme locations. The schedule event associates a given broadcast location (service, time and duration) with a given programme.
OnDemandProgram: is a specific type of a programme location for describing on demand programme locations.

B.6 OWL-Time

In the RDF and OWL communities there are at least two different options to represent time. One way is to use XSD data types (e.g. xsd:datetime) and another one is to use OWL-Time[47]. The OWL-Time ontology is designed for time conceptualizations (e.g. for describing the temporal content of web pages and the temporal properties of web services). It enables applications to express facts about topological relations among instants and intervals, together with information about durations and about date & time information.

XSD data types can be used in simple situations. For example, if somebody wants to model a birthday of a person or the release date of a movie. However, if an application has to support reasoning about time and topological order (before, during, after) between some temporal entities it is better to use OWL-Time. Also if exact information about the date and time is missing it is helpful to use OWL-Time. It can figure out some topological relations via reasoning. If event 1 happened before 2, and 2 before 3, it can be concluded via reasoning and OWL-Time that 1 also happened before 3. To illustrate more clearly the difference between using OWL-Time and using the XSD data types, let's look at a concrete example: Suppose someone wants to watch a TV programme at 14:00 on 18 September 2007. But the little child wants also the watch "Telekids" at 10:00 on the same day, and the programme duration is 120 minutes (2 hours). Will there be an overlap? It can be expressed as follows:

[47] http://www.w3.org/TR/owl-time/

```
<ex:programme1DateTime>2007-09-18T14:00:00+1:00</ex:programme1DateTime>
<ex:programme2DateTime>2007-09-18T10:00:00+1:00</ex:programme2DateTime>
<ex:programme2Duration>PT120M</ex:programme2Duration>
```

XSD data type only

```
<time:Interval rdf:ID="programme1DateTime">
  <time:begins rdf:resource="#p1Start"/>
</time:Interval>
<time:Interval rdf:ID="programme2DateTime">
  <time:begins rdf:resource="#p2Start" />
  <time:durationDescriptionOf rdf:resource="#p2DurationDescription" />
  <time:durationDescriptionDataType rdf:datatype="&xsd;duration">
    PT120M
  </time:durationDescriptionDataType>
</time:Interval>
<time:Instant rdf:ID="p1Start">
  <time:inCalendarClock rdf:resource="#p1StartDescription" />
  <time:inCalendarClockDataType rdf:datatype="&xsd;dateTime">
    2007-09-18T14:00:00+1:00
  </time:inCalendarClockDataType>
</time:Instant>
<time:Instant rdf:ID="p2Start">
  <time:inCalendarClock rdf:resource="#p2StartDescription" />
  <time:inCalendarClockDataType rdf:datatype="&xsd;dateTime">
    2007-09-18T10:00:00+1:00
  </time:inCalendarClockDataType>
</time:Instant>
<time:CalendarClockDescription  rdf:ID="p1StartDescription">
  <time:unitType rdf:resource="&time;unitMinute" />
  <time:year rdf:datatype="&xsd;gYear">2007</time:year>
  <time:month rdf:datatype="&xsd;gMonth">9</time:month>
  <time:day rdf:datatype="&xsd;gDay">18</time:day>
  <time:dayOfWeekField rdf:datatype="&xsd;nonNegativeInteger">3
  </time:dayOfWeekField>
  <time:dayOfYearField rdf:datatype="&xsd;nonNegativeInteger">261
  </time:dayOfYearField>
  <time:week rdf:datatype="&xsd;nonNegativeInteger">38</time:week>
  <time:hour rdf:datatype="&xsd;nonNegativeInteger">14</time:hour>
  <time:minute rdf:datatype="&xsd;nonNegativeInteger">0</time:minute>
  <time:timeZone rdf:resource="&tz-world;ATZ" />
</time:CalendarClockDescription>
<time:CalendarClockDescription  rdf:ID="p2StartDescription">
  <time:unitType rdf:resource="&time;unitMinute" />
  <time:year rdf:datatype="&xsd;gYear">2007</time:year>
  <time:month rdf:datatype="&xsd;gMonth">9</time:month>
  <time:day rdf:datatype="&xsd;gDay">18</time:day>
  <time:dayOfWeekField rdf:datatype="&xsd;nonNegativeInteger">3
  </time:dayOfWeekField>
  <time:dayOfYearField rdf:datatype="&xsd;nonNegativeInteger">261
  </time:dayOfYearField>
  <time:week rdf:datatype="&xsd;nonNegativeInteger">38</time:week>
  <time:hour rdf:datatype="&xsd;nonNegativeInteger">10</time:hour>
  <time:minute rdf:datatype="&xsd;nonNegativeInteger">0</time:minute>
  <time:timeZone rdf:resource="&tz-world;ATZ" />
</time:CalendarClockDescription>
<time:DurationDescription  rdf:ID="p2DurationDescription">
  <time:minutes rdf:datatype="&xsd;decimal">120</time:minutes>
</time:DurationDescription>
```

OWL-Time DateTimeDescription

Programme1 and programme2 are defined as intervals. *begins* is used for specifying the start times of the meetings. The datetimes are specified using *inCalendarClockDataType*. The duration of the meeting is specified using the duration description class.

From this example it can be seen that it is much simpler to describe date and time using the XML Schema data type. However, an advantage of using OWL-Time is that it can express more information than the XML data type only, such as "week", "day of week" and "day of year", so in the above example, it is also know that

18/09/2007 is Tuesday, on day 261 of the year, and in week 38 of the year. Moreover, each field is separate; this means that it is easier to extract the value of some fields for later use and easier to reason about. The unitType is added to define the precision; this date description leaves out the seconds, therefore the minutes are the most specific time unit. OWL-time is preferred in such situation because it facilitates a structure for a reasoner to determine whether if there is a conflict. Such a reasoner, can test and deduce the information. Only defining simple XML date types is not sufficient to reason about.

An example of using OWL-Time in SenSee is to map a search string to a specific time. If a user enters “sport on Tuesday” the system must be able to map the string “Tuesday” to a time concept. The lack of interpretation constrains the search to a simple string matching between the string and the content metadata. Using the time ontology, the string “Tuesday” will be identified as a time concept. OWL-Time allows defining time concepts like “Tuesday”. What Tuesday describes is an interval of two time instances, starting at day 3 at 00:00:00 (begins) and ending at 23:59:59 (ends). With that knowledge and the separation of fields it is a lot easier to find all the items of that particular Tuesday. For example instead of defining “Tuesday”, it is also possible to define “evening”, “morning”, “weekend”, “Sunday Afternoon” etc. Important relations are, *before*, describing that “Monday” comes before “Tuesday” or “noon” before “afternoon” and *contains*, describing that “Weekend” contains “Saturday” and “Sunday”. These are good examples of defining concepts to reason about. Items broadcasted on “Sunday afternoon”, are broadcasted on “Sunday“ and therefore also in the “Weekend”.

The conclusion is that OWL-Time can define complex date and time representations and enables reasoning by deducting information. For more information read the official description of OWL-Time at W3C.

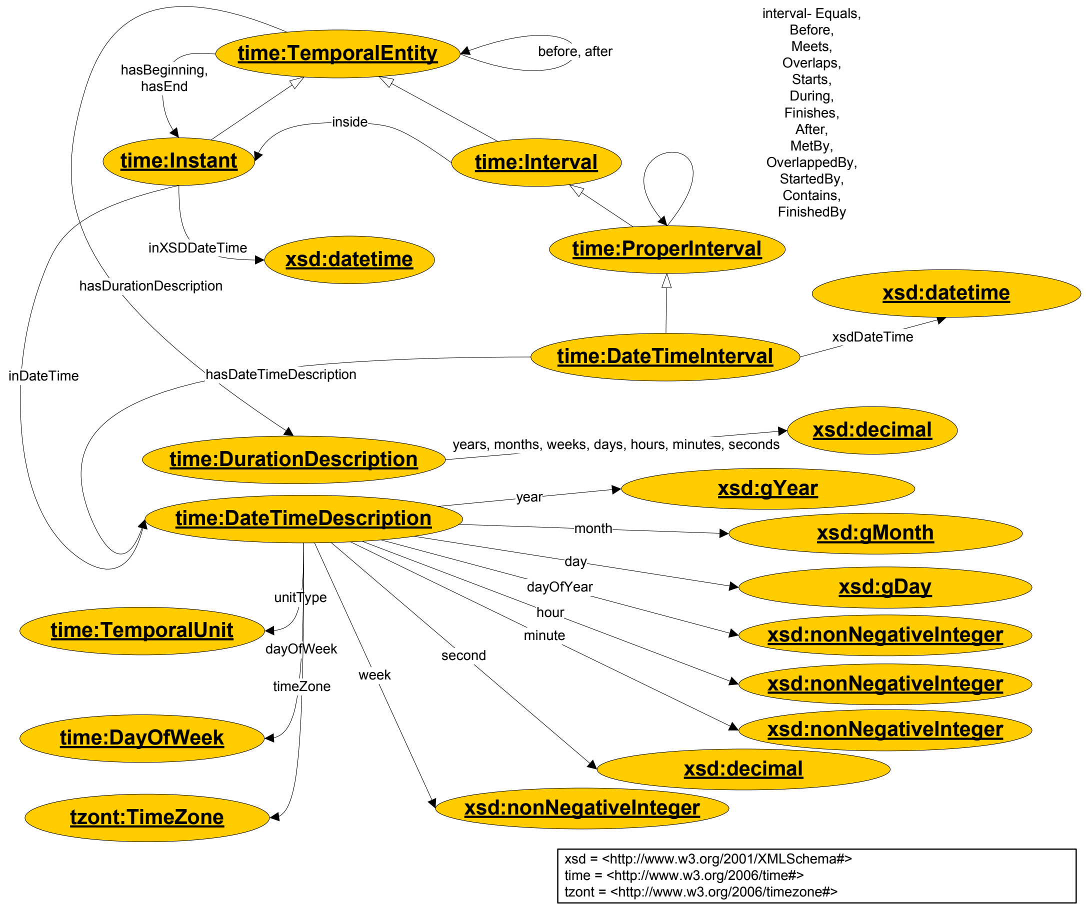


Figure B.6.1: OWL-Time Scheme

B.7 Geographical

Geographical ontologies are useful in providing geographical structures and to enable applications to reason about locations. For example if somebody wants to know about all actors or movies shot in the province of Noord-Brabant. Probably in most data sources it is only known that somebody was born in Eindhoven or Den Bosch. Without any additional geographical information the application is never able to figure out that Eindhoven is part of Noord-Brabant. Using geographical ontologies it is very easy to do such reasoning.

B.7.1 IMDb locations

IMDb[48] implements a hierarchical structure of locations in the data. The children of the root element are countries in the world, seas and oceans and some locations which cannot be defined properly, like a marine ship. Below these elements there is again a structure. The Netherlands for example is divided into provinces. These provinces are split up into cities and villages. Furthermore these cities and villages can have streets or other objects, like houses. The USA has a similar structure; instead of provinces they have states.

The structure is applied in the offline files.

```
Rancho Palos Verdes, California, USA
Grote Kerkplein, Rotterdam, Zuid-Holland, Netherlands
```

A small excerpt from how the location tree looks like:

```
Atlantic Ocean
    •   North Atlantic
            o   Irish Sea
            o   sea bed
                    ▪   Titanic wreck
    •   RMS Queen Elizabeth 2
    •   South Atlantic
Netherlands
•   Noord-Brabant
            o   's-Hertogenbosch
                    ▪   Brabanthallen
            o   Breda
                    ▪   Boschstraat 148
            o   Eindhoven
•   Noord-Holland
            o   Aalsmeer
                    ▪   Studio 1
            o   Amstel River
            o   Amsterdam
                    ▪   Amsterdam Arena
```

The model of this hierarchy is very simple. The form of a location in the IMDb set is built in a hierarchical string. From this string a structured model similar to the RDFS subclass structure has been created. Somebody may wonder why this very general concept is chosen. There is actually a good reason for this, because the hierarchical strings are different. House, street, province/state, region, country or combinations of them can all be possible. Therefore a general concept was chosen.

For example Rancho Palos Verdes, California, USA is modelled like USA `subComponentOf` the world (the root), California `subComponentOf` USA and finally Rancho Palos Verdes `subComponentOf` California. Equally named location names do not cause any problems. The location names have different parents, which means that both names have a different unique node. For instance there is an Athens in both the USA and in Greece, but both locations are still different. In the database there were no deviations like USA in one place and United States of America in another.

[48] http://www.imdb.com/LocationTree

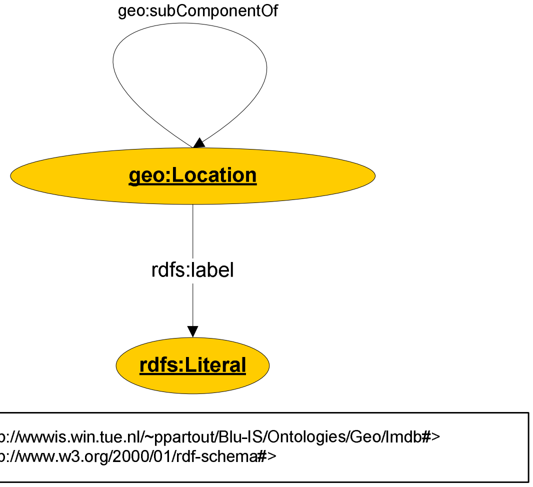

geo:subComponentOf
geo:Location
rdfs:label
rdfs:Literal
geo = <http://wwwis.win.tue.nl/~ppartout/Blu-IS/Ontologies/Geo/Imdb#>
rdfs = <http://www.w3.org/2000/01/rdf-schema#>

Figure B.7.1: Geo Scheme

B.7.2 TGN

The Getty Thesaurus of Geographic Name[49] (TGN) is a structured vocabulary providing geographical information, via narrower - broader relations and parent - child relations. It contains around 1.1 million names, place types, coordinates, and descriptive notes, focusing on places important for the study of art and architecture. The RDF scheme[50] is also used in another project done at the TU, E-Culture. There is also an interface to query the repository of this project[51].

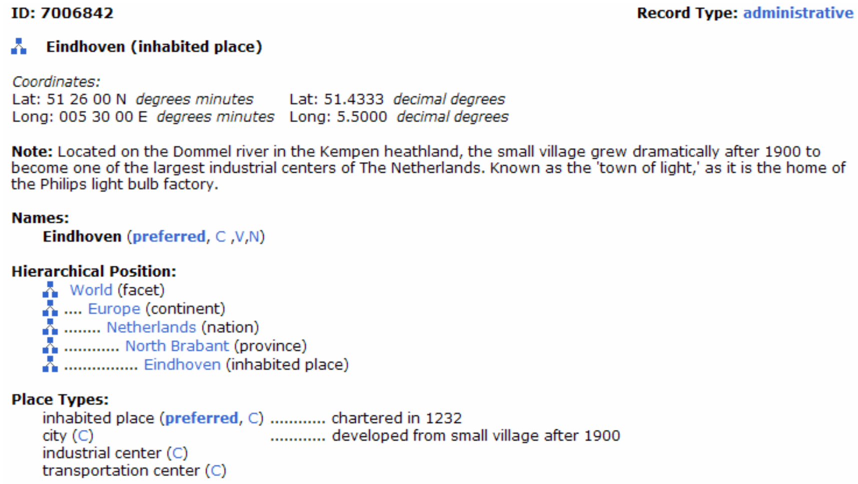
The following query is submitted to receive information about Eindhoven.

```
SELECT * FROM {<http://e-culture.multimedian.nl/ns/getty/tgn#7006842>} y {z}
```

The result of the query:

```
rdf:type                              tgn:AdministrativePlace
tgn:standardLatitude                  "51.4333"
tgn:standardLongitude                 "5.5"
vp:id                                 "7006842"
vp:labelPreferred                     "Eindhoven"
rdf:placeTypeNonPreferred             tgn:83040
rdf:placeTypeNonPreferred             tgn:83131
rdf:placeTypeNonPreferred             tgn:83371
tgn:placeTypePreferred                tgn:83002
 vp:descriptiveNote                   "Located on the Dommel river in the Kempen
                                      heathland, the small village grew dramatically after 1900 to
                                      become one of the largest industrial centers of The
                                      Netherlands. Known as the 'town of light,' as it is the home of
                                      the Philips light bulb factory."
vp:parentPreferred                    tgn:7003624
rdfs:subClassOf                       tgn:7006842
iface:latitude                        "51.4333"
iface:longitude                       "5.5"
iface:narrower                        tgn:7003624
tgn:placeType                         tgn:83002
tgn:coordinateLatitudeLongitude       "51.4333"
tgn:coordinateLatitudeLongitude       "5.5"
vp:parent                             tgn:7003624
ec:spatial                            tgn:83002
rdfs:label                            "Eindhoven"
```

[49] http://www.getty.edu/research/tools/vocabulary/tgn/index.html
[50] http://e-culture.multimedian.nl/resources/tgn.rdfs
[51] http://e-culture.multimedian.nl/demo/user/query



ID: 7006842 **Record Type:** administrative

Eindhoven (inhabited place)

Coordinates:
Lat: 51 26 00 N *degrees minutes* Lat: 51.4333 *decimal degrees*
Long: 005 30 00 E *degrees minutes* Long: 5.5000 *decimal degrees*

Note: Located on the Dommel river in the Kempen heathland, the small village grew dramatically after 1900 to become one of the largest industrial centers of The Netherlands. Known as the 'town of light,' as it is the home of the Philips light bulb factory.

Names:
Eindhoven (**preferred**, C ,V,N)

Hierarchical Position:
World (facet)
.... Europe (continent)
.... Netherlands (nation)
.... North Brabant (province)
.... Eindhoven (inhabited place)

Place Types:
inhabited place (**preferred**, C) .... chartered in 1232
city (C) .... developed from small village after 1900
industrial center (C)
transportation center (C)

Figure B.7.2: TGN website interface

A disadvantage of this repository is the lack of an API which can be used to access the repository easily and another disadvantage is a too strong focus on important places in art and architecture. This means that less interesting locations for the area of art and architecture are not included.

B.7.3 Geonames

Geonames[52] is another repository providing geographical information. This large and public geographical database contains a world-wide list of geo-administrative and geo-physical names. It provides data such as geographical names and postal codes. The database contains over 6.5 million entries for geographical names whereof 2.2 million cities and villages. The Ontology for Geonames is available in OWL/Full[53] and OWL/Lite[54] with the former importing the latter and adding some declarations. Geonames is using 303 redirection ("see other") to distinguish the concept from the document about it.

1) http://sws.geonames.org/2756253/
2) http://sws.geonames.org/2756253/about.rdf

In the case of Eindhoven two URIs (1,2) are important. The first URI stands for the town Eindhoven. It can be used to refer to the town. The second URI is the document with the information Geonames has about Eindhoven. The latter can be used to do some reasoning about it. For example if somebody wants to know more about the country or province Eindhoven resides in. An excerpt of the Eindhoven document, as obtained through the RDF Webservice at URI `http://sws.geonames.org/2756253/about.rdf`:

```
<Feature rdf:about="http://sws.geonames.org/2756253/">
  <name>Eindhoven</name>
  <alternateName xml:lang="en">Eindhoven</alternateName>
  <alternateName xml:lang="nl">Eindhoven</alternateName>
  <featureClass rdf:resource="http://www.geonames.org/ontology#P"/>
  <featureCode rdf:resource="http://www.geonames.org/ontology#P.PPL"/>
  <inCountry rdf:resource="http://www.geonames.org/countries/#NL"/>
  <population>209620</population>
  <wgs84_pos:lat>51.45</wgs84_pos:lat>
  <wgs84_pos:long>5.4666667</wgs84_pos:long>
  <parentFeature rdf:resource="http://sws.geonames.org/2749990/"/>
  <nearbyFeatures rdf:resource="http://sws.geonames.org/2756253/nearby.rdf"/>
  <locationMap>http://www.geonames.org/2756253/eindhoven.html</locationMap>
  <wikipediaArticle rdf:resource="http://en.wikipedia.org/wiki/Eindhoven"/>
</Feature>
```

[52] http://www.geonames.org
[53] http://www.geonames.org/ontology/ontology_v2.0_Full.rdf
[54] http://www.geonames.org/ontology/ontology_v2.0_Lite.rdf

Features in Geonames are related to each other. Think about children (countries for a continent, administrative subdivisions for a country (province) etc.). In the case of Eindhoven the *parentFeature* returns Noord-Brabant. Also neighboring countries (Germany, Belgium) or places can be retrieved.

For example: `http://sws.geonames.org/2756253/nearby.rdf`

```
<Feature rdf:about="http://sws.geonames.org/2746214/">
  <name>Tongelreep</name>
  <nearby rdf:resource="http://sws.geonames.org/2756253/"/>
</Feature>
<Feature rdf:about="http://sws.geonames.org/2756251/">
  <name>Eindhovensch Kanaal</name>
  <nearby rdf:resource="http://sws.geonames.org/2756253/"/>
</Feature>
<Feature rdf:about="http://sws.geonames.org/2746599/">
  <name>Strijp</name>
  <nearby rdf:resource="http://sws.geonames.org/2756253/"/>
</Feature>
```

Geonames offers a good API which can be used in a Java project to query the Geonames repository and to process the results. The database is very large, containing hierarchical structures of data. It also extends the data with some useful additional information, like coordinates, neighboring places and Wikipedia pages.

B.8 Wordnet

WordNet[55] is a lexical database of the English language. Nouns, verbs, adjectives and adverbs are grouped into sets of synonyms called synsets. WordNet was created at the Princeton University in 1985. In 2006 the database contained about 150,000 words organized in over 115,000 synsets.

The goal of WordNet is to present a combination of a dictionary and thesaurus and to support automatic analysis of text and artificial intelligence. The database and software tools can be downloaded and used freely.

W3C also created WordNet [56] in an RDF/OWL representation of the entire structure of WordNet. By doing so, WordNet data can be accessed via RDF APIs and query languages, and to be mixed with non-WordNet data, as well as with other lexically-oriented material, such as extensions to, and derivatives of, WordNet.

There are three core concepts in WordNet, namely: the synset, the word sense and the word. Words are the basic units, e.g. 'car', while a sense is a specific sense in which a specific word is used (car as a motorcar or car as a railcar). Synsets group word senses with a synonymous meaning, such as {car, auto, automobile, machine, motorcar} or {car, railcar, railway car, railroad car}. Railway car is an example of collocation; a sequence of words that belong together and form a specific meaning. The various meanings of a word are in different synsets. The meaning of the synsets is further described with a short description; definitions and/or example sentences.
Most synsets are connected to other synsets via semantic relations.

Example for the word "height":

```
Synset 1
Height, tallness (the vertical dimension of extension; distance from the base of
something to the top)

Synset 2
Altitude, height (elevation especially above sea level or above the earth's surface) "the
altitude gave her a headache"
```

[55] http://wordnet.princeton.edu/

[56] http://www.w3.org/TR/wordnet-rdf/

B.8.1 Wordnet differences

WordNet Basic has a separate schema and instances files. The schema is a stripped-down version of the Full version. It does not provide the classes WordSense and Word, and the properties to connect them to Synset. The basic version has an additional property, namely senseLabel. The instance RDF file that belongs to WordNet Basic does not have instances of WordSense and Word, and each Synset has a set of senseLabels. The sense-Labels are filled with the content of the lexical forms that are connected to a Synset through WordSense and Word in the Full version. The instance files for the WordNet relations can be loaded individually as for the Full version, but only properties (relations) between Synsets are sensible to load. The relations antonymOf, seeAlso, participleOf and derivationallyRelated are between Word-Senses, and can therefore not be used in WordNet Basic.

B.9 Literature

Literature data can be very useful in providing additional information about movies and persons. Movies and literature are closely related, many written stories are adapted to the screen or vice versa. Maybe users could also be interested to read the book or to watch the movie. Literature repositories can help in gaining more information about the movie literature.

B.9.1 ISBNDb

ISBNdb[57] project is a database of books in different languages. They also provide on-line and remote research tools which can be used by individuals and scientists. The database contains almost 3 million books, 900.000 authors and 188.000 publishers. They provide a free and easy accessible Web service. It is based on standard HTTP queries and XML based responses. A maximum of 500 requests can be submitted per day.

For Example an ISBN query can be formulated as follows:

```
http://isbndb.com/api/books.xml?access_key=ZC68KAAG&index1=isbn&value1=0060988967&results=texts,authors
,subjects,details
```

The result of the query:

```
<ISBNdb server_time="2007-04-20T12:44:29Z">
 <BookList total_results="1" page_size="10" page_number="1" shown_results="1">
  <BookData book_id="ego_trips_big_book_of_racism" isbn="0060988967">
   <Title>ego trip's Big Book of Racism!</Title>
   <AuthorsText>Sacha Jenkins, Elliott Wilson</AuthorsText>
   <PublisherText publisher_id="regan_books">Regan Books</PublisherText>
   <Details dewey_decimal="" physical_description_text="8.1&quot;x10.0&quot;x0.9&quot;;
   304 pages; 1.9 lb" language="" edition_info="Paperback; 2002-10-01" dewey_decimal_normalized=""
   lcc_number="" change_time="2005-01-02T16:33:02Z" price_time="2007-04-20T12:18:17Z" />
   <Summary>Ferociously intelligent one moment, willfully smart-ass the next, ego
   trip's Big Book of Racism is a glorious, hilarious conflation of the racial
   </Summary>
   <Subjects>
    <Subject subject_id="amazon_com_entertainment_humor_political">Amazon.com -
    Entertainment -- Humor - Political
    </Subject>
    <Subject subject_id="amazon_com_nonfiction_social_sciences_discrimination">Amazon.com --
    Nonfiction -- Social Sciences -- Discrimination &amp; Racism
    </Subject>
   </Subjects>
   <Authors>
    <Person person_id="jenkins_sacha">Jenkins, Sacha</Person>
    <Person person_id="elliott_wilson">Elliott Wilson</Person>
   </Authors>
  </BookData>
 </BookList>
</ISBNdb>
```

[57] http://isbndb.com

B.9.2 WorldCat Library

WorldCat[58] is the world's largest network of library content (1 billion items in more than 10.000 libraries) and services. WorldCat libraries are dedicated to providing access to their resources on the Web, where most people start their search for information.

They also offer a service to download and process metadata that enables users of a Web application to link to WorldCat library lookup for published content. This set of metadata is in XML. It contains over four million records representing many popular titles and a selection of unique items. The set can only by used with permission from the copyright holder. An excerpt of the XML file can be found below.

```
<record>
 <url>http://worldcat.org/wcpa/oclc/70043491</url>
 <title>For one more day</title>
 <author>Mitch Albom</author>
 <edition>1st ed.</edition>
 <publisher_data>
   <location>New York</location>
  <publisher>Hyperion</publisher>
  <year>c2006</year>
 </publisher_data>
 <language>eng</language>
 <oclc-number>70043491</oclc-number>
 <isbn>1401303277</isbn>
 <lccn>2006-18824 2006018824</lccn>
 <ddc>813/.6</ddc>
 <literary-form>Fiction</literary-form>
 <genre>Fiction</genre>
 <document-type>Book</document-type>
 <subject>Mothers and sons</subject>
 <subject>Mothers (Death)</subject>
  <library-count>839</library-count>
 <holding-countries>DE, HK, CA, SG, GB, US</holding-countries>
</record>
```

[58] http://www.worldcat.org

Appendix C

IMDb offline database

The offline database [59] is a text dump of the IMDb metadata. The metadata is divided over a number of files.

All names and titles in the database are unique. These "identifiers" are used in the text files.
To guarantee this uniqueness, Roman numbers are introduced. There are two types of Roman numerals appearing in the names. Here is an example of the first type: `Ahern II, Lloyd`
Lloyd Ahern II is a cinematographer that has the same name as his father Lloyd Ahern except for the "II" used to distinguish between them. The Roman numeral is directly and officially part of the name and not in any parentheses. It is used in actual credits of movies and in other information sources that have data on this man. The reason is simply to make clear that you are talking about the son and not the father.

The second type of Roman numeral appearing in names is very similar, although there is a crucial difference. Again the Roman numeral is added to distinguish between two or more different people that share a name but are not identical. But this time the number is not part of the official name of a person as used in credits or other information sources. It is a number added by IMDb and it is always within parentheses, for example:
`John Smith (I)`

The need for these numbers arose naturally as the database grew over time and more and more name clashes occurred with the consequence that credits belonging to one person were also showing up for all others with the same name. Generally, the more common a name is the more likely there will be mix-ups and the more Roman numerals have to be added. As an example look up John Taylor for which already 15 variants are stored.

There are no strict rules as to which person gets which number although more important or famous people have lower numbers and also people that came chronologically before other people. The need for Roman numerals follows from the basic database design. The names and titles themselves are the keys into the data and must be unique, hence the need for Roman numerals.

The (qv) tag used within textual description means that the name also is an identifier within IMDb. Sometimes these identifiers are enclosed by underscores.

C.1 Files description

C.1.1 Cast Lists

- Actors
- Actresses

Each entry consists of up to five pieces of information: the name, the title, an attributes field (to store, for example, academy award nominations), the character name and a number to specify their position in the credits. Only movies and recurring TV roles are defined, no TV guest appearances.

Overview of keys:

- "xxxxx" a television series
- "xxxxx" (mini) a television mini-series
- [xxxxx] character name

[59] http://www.imdb.com/interfaces

- <xx> number to indicate billing position in credits

Note that the overall series is not included in the list, but that the actor is part of an episode.

```
Bloom, Orlando 'Kingdom of Heaven': Interactive Production Grid (2005) (V) [Himself]
               Lord of the Rings: The Return of the King, The (2003) [Legolas] <8> Pirates of the
               Caribbean: At Worlds End (2007) [Will Turner] <2>
               Pirates of the Caribbean: Dead Man's Chest (2006) [Will Turner] <2>

Perry, Matthew (I) "Friends" (1994) {The Last One: Part 1 (#10.19)} [Chandler Bing] <5>
                   "Friends" (1994) {The Last One: Part 2 (#10.20)} [Chandler Bing] <5>
```

C.1.2 Crew Lists

The miscellaneous list is a "catch all" for people in careers not covered by the other filmography lists. Anyone who worked on a movie can be listed in that list.

The keyword "as" is used in each file, to indicate that the person in question used a different name or abbreviated name. Uncredited (uncredited) can also be found in each file.

- Cinematographers

```
't Hart, Martijn Brooklyn Connection, The (2005)

Wolski, Dariusz (I) Pirates of the Caribbean: At Worlds End (2007)
                    Pirates of the Caribbean: Dead Man's Chest (2006) (director of photography)
                    Pirates of the Caribbean: The Curse of the Black Pearl (2003)

Baca-Asay, Joaquín     Architecture of Reassurance (1999)
                       Roger Dodger (2002) (as Joaquin Baca-Asay)

Bachelet, Jean Bas-fonds, Les (1936) (uncredited)
```

- Composers
  Overview of keys:
  (songs) : The composer composed songs for the movie
  (also songs) : The composer composed also songs for the movie

```
Arne, Thomas Augustine Atlantic Ferry (1941) (from "Rule Britannia") (uncredited)
                       Picture to Post (1969) (r)
                       Pirates of the Caribbean: The Curse of the Black Pearl (2003) (from "Rule
                       Britannia") (uncredited)
Arnheim, Gus   Angel Heart (1987) (r) (song "I Cried For You")
Arnold, Cecil  Cohens and the Kellys in Africa, The (1930) (non-dialogue version) (uncredited)
               Outside the Law (1930) (foreign version) (uncredited)
               Tarzan the Tiger (1929) (uncredited)
               "Bugs" (1995) {Hollow Man (#3.4)}  (title music)
               "Stargate SG-1" (1997) {CR (#4.15)}(original themes)
```

- Costume Designers

```
Adrian (I)             -But the Flesh Is Weak (1932) (gowns)
                       After Office Hours (1935) (gowns)
```

- Directors
  Some additional information (combinations possible):
  - (co-director)
  - (second director)
  - (reality segments)
  - (episodes)
  - (segment director)
  - (archive footage)
  - (segment) (DVD only)
  - (documentary director)
  - (location)
  - (minor segments) (uncredited)

    - (co-director) (as Damián Acosta)

- Editors
  Some additional information (combinations):
    - (co-editor)
    - (head editor)
    - (documentary segment)
    - (original version)
    - (English version)

- Miscellaneous
  A big variety of possible additional information: (playback singer), (stagehand), (programmer) etc.

- Producers
  Some additional information (combinations):
    - (producer)
    - (assistant producer)
    - (executive producer)
    - (co-executive producer)
    - (co-producer)
    - (line producer)
    - (associate producer)
    - (coordinating producer)
    - (supervising producer)
    - (talent producer)
    - (senior talent producer)
    - (cinematics producer)

- Production Designers
- Writers
    - (song) The writer is the author of a song on which the screenplay was based.
    - (story) The writer contributed on the story for the screenplay only or the writer is the author of a story on which the movie was based.
    - (play) The writer is the author of the play on which the screenplay was based.
    - (novel) The writer is the author of the novel on which the screenplay was based.
    - (novel XXX) The writer is the author of the novel on which the screenplay was based, but the novel is not the same title as the movie.
    - (book) The writer is the author of the book on which the screenplay was based.
    - (article) The writer is the author of the article on which the screenplay was based.
    - (operetta) The writer is the author of the operetta on which the screenplay was based.
    - (also XXX) The writer was involved in some other aspect than just writing the screenplay. They may have authored the novel the screenplay was based on, or developed the story, etc.

  Can also be (novel) – (novels), (story) - (stories). Some more examples of additional terms: (writer), (co-creator), (character development), (screenplay), (script), (adaptation), (also adaption), (dialogue)

C.1.3 Person Lists

- Also known as Names
  The aka list contains details of different names people have been billed under during their careers.

```
Depp, Johnny
(aka Noodlemantra, Oprah)
```

- Biography
  The biographies list contains personal trivia on people in the database including dates of birth, dates of death, mini-biographies and books about them.
  - NM: Person name
  - RN: Real name
  - NK: Nick name
  - DB: Date of Birth, Location
  - DD: Date of Death, Location (reason)
  - BG: Biography line
  - SP: Spouse
  - HT: Height
  - TR: Trivia
  - OW: Other Works
  - QU: Personal Quotes
  - SA: Salary
  - BT: Biographical movies
  - PI: Portrayed in (movie)
  - AT: Article
  - BO: Biography (print)
  - CV: Cover photo Magazine
  - IT: Interview (Publicity)
  - PT: Pictorial
  - WN: Where are they now

```
NM: Bloom, Orlando
RN: Orlando Jonathan Blanchard Bloom
NK: Orli
NK: OB
DB: 13 January 1977, Canterbury, Kent, England, UK
HT: 5' 11"
BG: Orlando Bloom was born in Canterbury, Kent, England on January 13,
BG: 1977.
TR: * During the filming for _Lord of the Rings: The Fellowship of the Ring, The (2001)_ (qv), he
fell off his horse and broke one of
TR:   his ribs.
TR: * (2002) Chosen as one of Teen People Magazine's "25 Hottest Stars Under
TR:   25."
OW: * (2002) Appeared in commercial for Gap clothing "Denim Invasion" with
OW:   Kate Beckinsale.
BO: * Robert Steele. _Orlando Bloom._ London: Plexus Publishing, 2004.
BO:   ISBN 0859653544
QU: * "Elf Envy...they all had it."
WN: * (January 2006) Filming Pirates of the Caribbean 3 in the Bahamas.
SA: * _Elizabethtown (2005)_ (qv) -> $3,000,000
SA: * _Kingdom of Heaven (2005)_ (qv) -> $2,000,000
IT: * "Radio Times" (UK), 5 November 2005, Vol. 327, Iss. 4258, pg. 18-19, by: Andrew Duncan, "Bloom
at the top"
AT: * "Hunks" (UK), 14 December 2004, by: Wakas Mir, "Orlando Bloom"
PT: * "Playboy" (USA), December 2005, Vol. 52, Iss. 12, pg. 93, by: Stephen Rebello, "Sex in Cinema
2005"
CV: * "Entertainment Weekly" (USA), 14 July 2006, Vol. 1, Iss. 886
CV: * "Entertainment Weekly" (USA), 7 October 2005, Vol. 1, Iss. 843

DB: 1 November 1944, Washington, District of Columbia, USA
DB: 16 December 1972, Auckland, New Zealand
```

C.1.4 Movie Lists

The following lists contain information that relates to individual titles.

- Also known as Titles
  The Also known as titles list contains original foreign language titles for the primary titles in the database.

```
Pirates of the Caribbean: Dead Man's Chest (2006)
   (aka P.O.T.C. 2 (2006)) (USA) (promotional abbreviation)
   (aka Pirates 2 (2006)) (USA) (informal short title)
   (aka Pirates of the Caribbean - Fluch der Karibik 2 (2006)) (Germany)
   (aka Pirates of the Caribbean 2 (2005))(USA) (working title)
   (aka Pirati dei Caraibi - La maledizione del forziere fantasma (2006)) (Italy)
```

- Alternative Versions
  There are many kinds of alternate versions, the most common are:
  Director’s cuts/Special Editions, Restored Versions, Censorship changes and Television versions.

```
# Pirates of the Caribbean: The Curse of the Black Pearl (2003)
- Disk 2 of the DVD features 19 deleted/extended scenes:
- The full version of the scene where Will accidentally "takes" a candle off the wall.
- Extended scene of Port Royal when the wind makes a sudden change when the Aztec
  Gold "calls".
```

- Business
  A list of business matters. The list includes information such as production dates, studio locations, film budgets, box office revenues, distributor rentals revenues, copyright information and other financial/statistical data.
  - MV: movie title
  - BT: budget
  - AD: Admissions
  - CP: copyright holder
  - GR: gross
  - OW: Opening weekend
  - WG: Weekend gross
  - PD: Production dates
  - SD: Filming dates

```
MV: Pirates of the Caribbean: Dead Man's Chest (2006)

AD: 3,084,149 (Brazil) (10 September 2006)
AD: 3,053,848 (Brazil) (3 September 2006)
BT: USD 225,000,000
CP: © 2006 Disney Enterprises Inc.
OW: EUR 1,747,551 (Netherlands) (16 July 2006) (132 screens)
PD: 22 October 2004 - ?
SD: March 2005 - 10 September 2005
WG: EUR 510,545 (Netherlands) (20 August 2006) (133 screens)
WG: EUR 1,747,551 (Netherlands) (16 July 2006) (132 screens)
```

- Certificates
  The certificates list contains the certificates / ratings classifications (MPAA, BBFC etc.)

```
Pirates of the Caribbean: Dead Man's Chest (2006) Netherlands:12
Pirates of the Caribbean: Dead Man's Chest (2006) USA:PG-13 (certificate #42793)
Pirates of the Caribbean: Dead Man's Chest (2006) Switzerland:12 (canton of Vaud)
Pirates of the Caribbean: The Curse of the Black Pearl (2003) UK:12 (video rating)
Pirates of the Caribbean: The Curse of the Black Pearl (2003) UK:12A (original rating)
Pirates of the Caribbean: The Curse of the Black Pearl (2003) Canada:PG    (Alberta/British
Columbia/Manitoba/Ontario)
```

- Color Information
  The color information list records whether the movie was shot in color or black&white.

```
Casino Royale (1967)                                    Color   (Technicolor)
Casino Royale (2006)                                    Black and White
Casino Royale (2006)                                    Color
```

- Countries
  The countries list records the country of origin.

```
Pirates of the Caribbean: Dead Man's Chest (2006)    USA
Vet hard (2005)                                      Netherlands
Vet hard (2005)                                      Belgium
```

- Crazy Credits
  The crazy credits list contains trivia and jokes that appear in movie credits.

```
# Pirates of the Caribbean: Dead Man's Chest (2006)
- Stay till the very end of the credits to find out the fate of one of the
  characters.
- 'Johnny Depp' (qv)'s dentist has his own credit at the
   end of the movie.
```

- Genres
  The genres list records the genres which describe the movie. The name and number of Genres are subject to change over time. Current genres in use:
  Action, Adventure, Animation, Biography, Comedy, Crime, Documentary, Drama, Family, Fantasy, Film-Noir, Game-Show, History, Horror, Music, Musical, Mystery, News, Reality-TV, Romance, Sci-Fi, Short, Sport, Talk-Show, Thriller, War and Western

```
Pirates of the Caribbean: Dead Man's Chest (2006)    Action
Pirates of the Caribbean: Dead Man's Chest (2006)    Adventure
Pirates of the Caribbean: Dead Man's Chest (2006)    Fantasy
```

- Goofs
  The goofs list contains a list of mistakes/goofs in movies.
  Goof types: BOOM, CHAR, CONT, CREW, DATE, FACT, FAIR, GEOG, PLOT, MISC, SYNC

```
# Pirates of the Caribbean: Dead Man's Chest (2006)
- FAKE: When Jack Sparrow falls into the grave, the print of a modern
  shoe can be seen clearly in the dirt at the top of the grave.

- DATE: When Jack Sparrow, Elizabeth and Norrington are arguing on the
  pier in Tortuga, the hull of a ship can be clearly seen behind them
  bearing a Plimsoll line, which wasn't used until the late 19th
  century.
```

- Keywords
  List of keywords defined per movie. These keywords are a kind of summary of the plot. The list of possible keywords is too extensive to summarize in this document.

```
Pirates of the Caribbean: Dead Man's Chest (2006)    dice-game
Pirates of the Caribbean: Dead Man's Chest (2006)    axe
Pirates of the Caribbean: Dead Man's Chest (2006)    part-of-trilogy
```

- Language
  The list of languages used in a movie.

```
Pirates of the Caribbean: Dead Man's Chest (2006)    Turkish
Pirates of the Caribbean: Dead Man's Chest (2006)    English
```

- Literature
  The literature list records references to essays, critical analyses and books related to specific movies.

Literature types:

- MOVI: indicates the title of the movie and the year of first release
- SCRP: Bibliographical information of the original screenplay/teleplay (if published) also original film text
- NOVL: Bibliographical information of the original literary source (only if published) → original novel, theatre play, short story
- ADPT: Bibliographical information of the adaption of the literary source (only if published) → adaption of novel, play, short story
- BOOK: Monographic book related to this film production
- PROT: Protocol of production process of this film -> "on location"-literature
- IVIW: Interviews of cast&crew of this filmproduction and related to this specific film production
- CRIT: Reviews of this specific film production in printed media (in newspapers, magazines, non-monographic books)
- ESSY: Essays related specially to this particular film production (in newspapers, magazines, non-monographic books)
- OTHR: Bibliographical information of other literature of this film production

Categorical information types:

- (NP) Newspaper
- (WNP) Weekly Newspaper
- (MG) Magazine
- (BK) Book

```
MOVI: Pirates of the Caribbean: Dead Man's Chest (2006)

ADPT: Disney, Walt. Pirates of the Caribbean. (Disneyland)

BOOK: McCafferty, Catherine. Pirates of the Caribbean: Dead Man's Chest - The Movie Storybook.
    Disney Press, 2006, ISBN 1423100255

CRIT: Horwitz, Jane. Pirates of the Caribbean: Dead Man's Chest (PG-13, 140 minutes). In: The
    Washington Post (USA), Vol. 129, Iss. 214, 7 July 2006, Pg. 33 in weekend section/family
    filmgoer, (NP)
CRIT: Schwarzbaum, Lisa. "Less Than C-Worthy (D+)". In: Entertainment Weekly (USA), Vol. 886,
    14 July 2006, Pg. 59-60, (MG)
CRIT: Thomson, Desson. Yo-Ho-Hum: Johnny Depp Shakes His Booty But 'Pirates of the Caribbean'
    Is Just a Rum Joke. In: The Washington Post (USA), Vol. 129, Iss. 213, 6 July 2006, c1+c5,
    (NP)

ESSY: Lytal, Cristy. "Pirates of The Caribbean: Dead Man's Chest - The Flight on The Mill
    Wheel". In: Premiere (USA), Vol. 19, Iss. 9, June 2006, Pg. 82-83, (MG)
ESSY: Rottenberg, Josh. "The Piracy Debate". In: Entertainment Weekly (USA), Vol. 886, 14 July
    2006, Pg. 36-41, (MG)
```

- Locations
  The locations list holds information on where movies were filmed.

```
Pirates of the Caribbean: Dead Man's Chest (2006) Bahamas
Pirates of the Caribbean: Dead Man's Chest (2006) Los Angeles, California, USA
Pirates of the Caribbean: Dead Man's Chest (2006) St Vincent, St Vincent and the Grenadines
Pirates of the Caribbean: Dead Man's Chest (2006) Walt Disney Studios, Burbank, California,
USA(studio)
Pirates of the Caribbean: At Worlds End (2007) Guadalupe, California, USA (dunes)
Pirates of the Caribbean: At Worlds End (2007) Rancho Palos Verdes, California, USA (sea
sequences)
Pirates of the Caribbean: At Worlds End (2007) Universal Studios, Universal City, California,
USA (stage 28)
```

- Movies
  The basic list of all movie titles and production dates. Unknown production dates are indicated by (????). For series, dates are defined as a interval (1994-2004).

(1) Movies, unless otherwise indicated the title is a movie
(2) TV-movies, indicated by a (TV) tag
(3) Made for video movies, indicated by a (V) tag
(4) Made for videogames, indicated by a (VG) tag
(5) TV-series, indicated by enclosing the title in " "
(6) Mini-series, indicated by enclosing the title in " " and a (mini) tag

Example Movie:

```
    Pirates of the Caribbean: Dead Man's Chest (2006) 2006
```

Example Series/Episode:

```
    "Friends" (1994) 1994-2004
    "Friends" (1994) {The One After I Do (#8.1)} 2001
    "Rich Man, Poor Man" (1976) (mini) 1976
    "Richard & Judy" (2001) {(2005-01-10)} 2005
```

- Movie links
  The movie links list notes links between movies in the database such as sequels, remakes and also trivial connections such as homages and spoofs.
  Connection types:
  - Featured in / Features
  - Spoofed in / Spoofs
  - Referenced in / References
  - Followed by / Follows
  - Edited into / Edited From
  - Spin off / Spin off from
  - Version of
  - Remade as / Remake of
  - Alternate language version of

```
Pirates of the Caribbean: Dead Man's Chest (2006)
(featured in "Richard & Judy" (2001) {(2006-07-19)})
(followed by Pirates of the Caribbean: At Worlds End (2007))
(follows Pirates of the Caribbean: The Curse of the Black Pearl (2003))
(referenced in "Richard & Judy" (2001) {(2006-07-19)})
(references Black Pirate, The (1926))
(spoofed in Epic Movie (2007))
(spoofs Phantom of the Opera, The (1925))
```

- MPAA-ratings-reasons
  This list contains movies that the MPAA has rated for USA certification.

```
MV: Pirates of the Caribbean: Dead Man's Chest (2006)
RE: Rated PG-13 for intense sequences of adventure violence,
RE: including frightening images.
```

- Plot Summaries
  The plot summaries list has brief outlines of some of the movies in the database.

```
MV: Pirates of the Caribbean: Dead Man's Chest (2006)

PL: Captain Jack Sparrow's life is in debt to the legendary Davey Jones,
PL: Captain of the Flying Dutchmen.
```

- Quotes
  The quotes list contains memorable quotes from movies.

```
# Pirates of the Caribbean: Dead Man's Chest (2006)
'Bootstrap' Bill Turner: Wondering how it's played?
Will Turner: I understand. It's a game of deception. But your bet includes all the dice, not just
your own. What are they wagering?
'Bootstrap' Bill Turner: Oh, the only thing we have. Years of service.
```

- Release Dates
  The release dates list records the release dates of movies on a country by country basis.

```
Pirates of the Caribbean: Dead Man's Chest (2006) USA:24 June 2006 (Disneyland) (premiere)
Pirates of the Caribbean: Dead Man's Chest (2006) UK:3 July 2006 (London) (premiere)
Pirates of the Caribbean: Dead Man's Chest (2006) UK:6 July 2006
Pirates of the Caribbean: Dead Man's Chest (2006) Netherlands:13 July 2006
Pirates of the Caribbean: Dead Man's Chest (2006) Pakistan:6 August 2006 (Karachi) (premiere)
Pirates of the Caribbean: Dead Man's Chest (2006) Switzerland:27 July 2006 (German speaking region)
```

- Running Times
  The running times list contains the length in minutes of movies in the database.

```
Pirates of the Caribbean: Dead Man's Chest (2006)   150
Gladiator (2000)                                           171    (extended version)
Gladiator (2000)                                           155
Frighteners, The (1996)                                    110
Frighteners, The (1996)                                    USA:122 (director's cut)
"Friends" (1994)                                           22
"Friends" (1994) {The One After I Do (#8.1)}        22
"Friends" (1994) {The One with Ross and Monica's Cousin (#7.19)} USA:22
"Friends" (1994) {The One with Ross and Monica's Cousin (#7.19)} 22
```

- Sound Mix
  The sound mix list records whether movies are recorded in mono, stereo, DTS, etc.
  Example Sound-mix types are:
  Silent, Mono, Fantasound, Sensurround, 70mm 6-Track, Dolby Stereo, Dolby Stereo SR, Dolby Stereo Digital, CDS, DTS, DTS-Stereo and SDDS.

```
Pirates of the Caribbean: Dead Man's Chest (2006) SDDS
Pirates of the Caribbean: Dead Man's Chest (2006) Dolby Digital
Pirates of the Caribbean: Dead Man's Chest (2006) DTS
```

- Soundtracks
  The soundtracks list contains details of songs used on the soundtracks of movies.
  It is not defined what kinds of actions are used and if the keywords are limited. A small list of used keywords:
  By, Performed by, Used by, Published by, Composed by, Written by, Sung by, Music by, Under License, Vocal by, Produced by, By Arrangement, Lyrics by, Played In, Played as, Composed & Conducted by etc.

```
# Pirates of the Caribbean: Dead Man's Chest (2006)
- "Two hornpipes"
  Written by Skip Henderson

# Pitch People (1999)
- "Danza di piccoli schiavi mori"
  Written by 'Giuseppe Verdi' (qv)
  Performed by the 'National Symphony Orchestra of Ireland' (qv)
  Conducted by Rico Saccani
  Courtesy of Naxos of America, Inc.
```

- Tag Lines
  The tag lines list contains advertising tag lines used to publicise movies.

```
# Pirates of the Caribbean: The Curse of the Black Pearl (2003)
Prepare to be blown out of the water.
Over 3000 Islands of Paradise -- For some it's A Blessing -- For others...
```

- Technical Info
  The technical list records technical information such as aspect ratio, negative formats, print formats, cinematographic process etc.
  - MET: Length of a film in meter.
  - OFM: Film negative format (mm) or "Video" with an additional attribute for the TVstandard.
  - PFM: Film negative format (mm) or "Video" with an additional attribute for the TVstandard.
  - RAT: Aspect Ratio, width to height (_.__ : 1).
  - PCS: Cinematographic process or video system.
  - CAM: Camera model and lenses.
  - LAB: Laboratory, "name, city/region, country" – "name,country"

```
Pirates of the Caribbean: Dead Man's Chest (2006) CAM:Arri 235 (hand-held)
Pirates of the Caribbean: Dead Man's Chest (2006) CAM:Panaflex Platinum Cameras, Primo Lenses
Pirates of the Caribbean: Dead Man's Chest (2006) LAB:Technicolor, USA
Pirates of the Caribbean: Dead Man's Chest (2006) OFM:35 mm
Pirates of the Caribbean: Dead Man's Chest (2006) PCS:Digital Intermediate (master format)
Pirates of the Caribbean: Dead Man's Chest (2006) PCS:Super 35 (source format)
Pirates of the Caribbean: Dead Man's Chest (2006) PFM:35 mm (anamorphic)
Pirates of the Caribbean: Dead Man's Chest (2006) RAT:2.35 : 1
```

- Trivia
  The trivia list contains all kinds of trivia for movies.

```
# Pirates of the Caribbean: Dead Man's Chest (2006)
- The Black Pearl is the only ship in the Pirates of the Caribbean:
  Dead Man's Chest and its sequel which operate under power. All others, except for the
      "Interceptor" from the first movie, are built atop other ships such as barges with only
      the portions seen by the cameras completed.
- The ship, The Black Pearl, was built in Bayou La Batre, Alabama at
  Steiner Shipyard, just a couple of blocks from Bayou La Batre City Hall.
- One of the film's last lines - "Bring me that horizon" - was conceived by
  'Johnny Depp' (qv) on the morning the scene was filmed.
- 'Naomie Harris' (qv) was filming this film together with
  _Miami Vice (2006)_ (qv). All of her scenes as Tia Dalma were filmed at Bahamas during the
      weekends.
```

C.1.5 Company Lists

File list containing involved companies.

- Distributors

```
Pirates of the Caribbean: Dead Man's Chest (2006) Buena Vista Home Entertainment(BVHE) [de]
    (2006) (Germany) (DVD)
Pirates of the Caribbean: Dead Man's Chest (2006) Buena Vista International Finland [fi]
    (2006) (Finland) (theatrical)
Pirates of the Caribbean: Dead Man's Chest (2006) Buena Vista International UK [gb] (2006)
    (UK) (theatrical)
Pirates of the Caribbean: Dead Man's Chest (2006) Buena Vista International [fr] (2006)
    (France) (theatrical)
Pirates of the Caribbean: Dead Man's Chest (2006) Buena Vista International [nl] (2006)
    (Netherlands) (theatrical)
Pirates of the Caribbean: Dead Man's Chest (2006) Columbia Pictures [ph] (2006) (Philippines)
    (theatrical)
Pirates of the Caribbean: Dead Man's Chest (2006) Falcon [cz] (2006) (Czech Republic)
    (theatrical)
Pirates of the Caribbean: Dead Man's Chest (2006) Walt Disney Studios [us] (USA)
Pirates of the Caribbean: Dead Man's Chest (2006) Buena Vista Pictures [us]
```

- Miscellaneous

```
Pirates of the Caribbean: Dead Man's Chest (2006) Bahamas Film Studios, The [bs] (movie
    studio)
Pirates of the Caribbean: Dead Man's Chest (2006) Bender ET [us] (GFCI shock protection
    provided by)
Pirates of the Caribbean: Dead Man's Chest (2006) Chapman/Leonard Studio Equipment [us]
    (cranes)
Pirates of the Caribbean: Dead Man's Chest (2006) Choir of the King's Consort [gb] (choir)
Pirates of the Caribbean: Dead Man's Chest (2006) Hat Trick Catering [us] (catering)
Pirates of the Caribbean: Dead Man's Chest (2006) Panavision [us](cameras and lenses)
Pirates of the Caribbean: Dead Man's Chest (2006) Scarlet Letters [us] (end titles)
Pirates of the Caribbean: Dead Man's Chest (2006) Sony Pictures Studios [us] (score recorded
    at)
Pirates of the Caribbean: Dead Man's Chest (2006) Technicolor [us] (prints)
Pirates of the Caribbean: Dead Man's Chest (2006) Walt Disney Records [us] (soundtrack)
```

- Production
  The production companies list records the production company / studio which produced the movie.

```
Pirates of the Caribbean: Dead Man's Chest (2006) Second Mate Productions [us]
Pirates of the Caribbean: Dead Man's Chest (2006) Jerry Bruckheimer Films [us]
Pirates of the Caribbean: Dead Man's Chest (2006) Walt Disney Pictures [us]
Pizza Runners (1996) Tangents Productions
Piñatas: The Movie (2005)     Excalibur Animacion [es] (in co-production with)
Piñatas: The Movie (2005)     UKFS 3 [gb] (in co-production with)
```

- Special-effects

```
"Australian Survivor" (2002) Ua3 Special Effects [au] (pyrotechnics)
Pirates of the Caribbean: Dead Man's Chest (2006) Animal Makers [us] (animatronic animal
    design)
Pirates of the Caribbean: Dead Man's Chest (2006) Asylum VFX [us] (additional visual effects)
    (as Asylum)
Pirates of the Caribbean: Dead Man's Chest (2006) Evil Eye Pictures [us] (visual effects)
Pirates of the Caribbean: Dead Man's Chest (2006) Gentle Giant Studios Inc. [us]
Pirates of the Caribbean: Dead Man's Chest (2006) Proof [us] (pre-visualization)
```