Generic adaptation framework for unifying adaptive web-based systems

Generic Adaptation Framework
for unifying adaptive web-based systems

Evgeny Knutov

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Knutov, Evgeny

Generic Adaptation Framework for Unifying Adaptive Web-based Systems
door Evgeny Knutov.
Eindhoven: Technische Universiteit Eindhoven, 2012. Proefschrift.

A catalogue record is available from the Eindhoven University of Technology Library
ISBN 978-90-386-3131-8
NUR 980

Subject headings: hypertext / hypermedia / adaptive hypermedia / framework / recommender systems / web information systems

SIKS Dissertation Series No. 2012-14
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Printed by: BOXPress / proefschriftmaken.nl

Cover Design: Evgeny Knutov
Cover Photo: Evgeny Knutov

Generic Adaptation Framework for Unifying Adaptive Web-based Systems

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven,
op gezag van de rector magnificus, prof.dr.ir. C.J. van Duijn,
voor een commissie aangewezen door het College voor
Promoties
in het openbaar te verdedigen
op woensdag 9 mei 2012 om 16.00 uur

door

Evgeny Knutov

geboren te Sint Petersburg, Rusland

Dit proefschrift is goedgekeurd door de promotor:

prof. dr. P.M.E. De Bra

Copromotor:
dr. M. Pechenizkiy

Acknowledgement

I would like to express my gratitude to those who gave me the possibility to work on the GAF project and supported me during all these years.

First I would like to thank my supervisors Prof. Dr. Paul De Bra and Dr. Mykola Pechenizkiy for their support, guidance, long scientific and inspirational discussions.

I would also like to thank Eindhoven University of Technology (TU/e) for the support in my research and the Netherlands Organization for Scientific Research - Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO) for supporting the GAF project.

Many thanks to the reviewing committee: Prof. Dr. Wolfgang Nejdl, Prof. Dr. Hugh Davis and Prof. Dr. Paul Grefen, who helped to improve this thesis and provided their valuable comments and suggestions. I would also like to thank Prof. Dr. Barry Smyth who let me spend time in his research group doing case-studies and the members of 'Clarity' centre: Maurice Coyle, Peter Briggs, Michael O'Mahony, John Hannon, Kevin McNally, Owen Phelan and Zurina Saaya. I would like to thanks all the inspirational researchers from the 'GRAPPLE' project and AH community. Thanks to all those people I met at AH, UMAP, HT and other conferences I attended for interesting discussions, colleagues from the IS group for help and especially my office mates Jorn, Pieter, Kees and Yongming.

And last but not least thanks to my Mom, Dad and friends for being supportive and encouraging me over the years.

Thank you all!

Contents

Chapter 1

Introduction

The research field of adaptive hypermedia (AH) and adaptive web-based information systems (AHS for short) has been growing rapidly during the past 15 years and this has resulted in new terms, models, methodologies, and a plethora of new systems. Adaptive systems are becoming more popular as tools for user-driven access to information. Adaptation of an information system or service to a user has been proven to be a powerful and useful concept. It is particularly helpful for the reduction of the information overload which is frequently experienced on the Internet or any other information system of a large scale. But it is equally helpful for guiding users towards "interesting" topics, products, artifacts or descriptions thereof in electronic shops, libraries or museums, or for filtering appropriate items from a general or domain-specific news feed.
The Generic Adaptation Framework (GAF) first and foremost creates a common framework for describing current and future AHS and adaptive web-based systems in general. It provides a common taxonomy and a reference model that encompasses the most general architectures of the present and proposed future architectures, including conventional AHS, and different types of personalization-enabling systems and applications such as Recommender Systems (RS) personalized web search, semantic web enabled applications used in personalized information delivery, adaptive e-Learning applications and many more. At the same time GAF is trying to bring together two (seemingly not intersecting) views on the adaptation: a classical pre-authored type, with conventional domain and overlay user models and data-driven adaptation which includes a set of data mining, machine learning and information retrieval tools. To bring these research fields together we conducted a number GAF compliance studies including RS (such as HeyStaks and Twittomender), AHS (such as KBS-Hyperbook, AHA!, APeLS and others), and other applications (e.g. CHIP and RPM system) combining adaptation, recommendation and search. We also performed a number of real systems case-studies and a detailed analysis and evaluation of the framework. Secondly it introduces a number of new ideas in the field of AH, such as the Generic Adaptation Process (GAP) which aligns with a layered (data-oriented) architecture and serves as a reference adaptation process. This also helps to understand the compliance features mentioned earlier. Besides that GAF deals with important and novel aspects of adaptation enabling and leveraging technologies such as provenance and versioning. The existence of such a reference basis should stimulate AHS

research and enable researchers to demonstrate ideas for new adaptation methods much more quickly than if they had to start from scratch. GAF will thus help bootstrap any adaptive web-based system research, design, analysis and evaluation.

1.1 Motivation

The *GAF* reference model is important for a number of reasons. First and foremost it allows people to communicate about the architecture of AH systems using a common framework and vocabulary. Many hypertext systems have been described in terms of the Dexter model [61] for instance, and adaptive hypertext systems are often described using the AHAM [139] terminology. A reference model allows people to specify the functionality of a system and to compare the architecture and functionality of different systems using the same terms and definitions. And this becomes the key point because many research simply don't use the same terminology to describe similar or identical things thus producing systems with the same or slightly different functionality over again. Besides they hardly look out for seemingly unrelated concepts from outer research fields and hardly accept new trends leaving AH research field relatively closed. Essentially we offer a common AH research language for the researchers to communicate, explain the terms and definitions they've been using, elaborate on the functionality which had multiple interpretations.
The reference model also stimulates the development of new systems and application that match more of the functionality captured by the reference model (than the systems existing before the creation of the reference model). The AHA! system [48] was a good example of a system that has evolved to match the AHAM model. But of course AHA! was not a full implementation of the reference model, as AHAM was much more general than any existing system. Thus having an up-to-date common reference baseline new systems would be able to compare and evaluate the functionality (basically speaking the same language). This will help to explain and evaluate the functionality and identify further system extensions (examples are shown in sections 6.1.2 and 6.2.3 accordingly).
Previous reference models described the architecture of mainly closed and small to medium size adaptive hypermedia applications, besides many newer aspects that have entered the AH research field cannot be expressed and are not implemented. Thus there is also need to breach this and bring adjacent research initiatives in the new generic reference AH model. We elaborate on the detailed *GAF* contribution in section 1.4.

1.2 Research Goal and Approach

GAF research focuses on how different adaptive hypermedia aspects can fit in a generic adaptive information system framework. The goal is to describe the framework not as an abstract formal model but rather as an abstract description of a very generic adaptive information system architecture. Thus GAF research consists of the following steps or phases:

- Studying (in a survey form) existing adaptive hypermedia systems, web-based information systems, intelligent tutoring systems, intelligent agent systems and other related fields, in order to create an inventory of methods and techniques used in such systems and applications.

- As as a result of the system analysis the architecture is created. It consists of a number of components, including a part that deals with concepts, concept relationships and relationships in ontologies, a part that deals with individual user-modeling, a part that performs data mining (e.g. to identify groups and group properties), and a part that deals with the low-level adaptation rules. The global architecture describes how the different modules or services work separately and comprise the overall architecture and work together.

- More detailed aspects including semantic-web, reasoning and data-oriented (including data-mining and machine retrieval techniques) parts study how systems can reason over concept structures and are used to define search facilities that use concepts, not page contents, as the basis for searching, to identify user groups, and navigation patterns respectively. Part of this research is fairly standard therefore we don't elaborate on the general application of these methods but rather discuss novelties in AH context. Besides that we touch upon case-based reasoning aspects, define association rules for adaptation and much more.

- The global architecture is used to describe the architecture and models of other research projects (AHAM, AIMS, Hera, LAOS, LAG, MOT, CHIP, Twittomender, HeyStaks, ApeLS, GALE, etc.) and models and systems developed in other research groups.

- A model as general as GAF is able to describe systems that exhibit undesirable properties, like having adaptation specifications that may result in infinite loops when executed. In AHAM termination and confluence problems were already studied and a conservative analysis method for the ECA adaptation rules was developed. In GAF different types of ECA adaptation rules are studied and presented in a single reasoning framework leading to the specification of analysis tools that help application developers determine important properties of their systems.

- A reference model describes the functionality of an adaptation framework and an architecture that could possibly be implemented in a system. In order for GAF to be not just a model for describing and comparing systems, but also a guide for a future development of Adaptive Information Systems (AIS), it is represented from a number of diverse perspectives: data and system modelling, reasoning, process-oriented, compliance, analysis, etc. A complete implementation is not feasible within the boundaries of the GAF and current research observations. But still the main goal is to provide the architecture that needs to be implemented, to separate essential elements from optional and to define criteria distinguishing within these elements, to provide a modular structure that can be used either separately or to-

gether, depending on the needs of the intended application, that can be developed over the generic framework to satisfy different needs.

1.3 Research Questions

This dissertation provides answers to the following research questions:

- *Question 0: What is use of a generic framework and what is the specific use of GAF?*

  Here we answer what is the intended purpose of the adaptation framework, reveal the particular use and potentials of GAF for the AH community and provide extensive definitions of the adaptation framework, adaptation process and related terms we use to define the composition and functionality of GAF. We essentially elaborated on the answer to this question in the motivation in section 1.1 (and that's why this question becomes question "zero" and sets the starting point for the *GAF* research).

- *Question 1: Is it possible to capture all up-to-date AHS functionality and developments? What would be the exemplary set of systems that can serve as a basis to capture the generalities of AH? What approaches, methods and techniques exist to support up-to-date adaptive hypermedia solutions?*

  Most probably we wont be able to capture "All" AH systems in order to create an AH toolkit, but to answer this question we study in details all state-of-the-art systems dating back to the very beginning of AH research up until now, including the most recent developments in AH and adjacent fields. Besides that we try to capture new trends and ideas in AH development during the past 4 years when *GAF* research essentially takes place.

- *Question 2: Is it possible to bring conventional pre-authored Adaptive Hypermedia together with a data-driven approach to personalization?*

  Therefore we investigate data-driven personalization aspects such as Recommender Systems, Search Engines, etc., and conduct a number of compliance studies to bridge these two approaches together (conventional adaptation and data-driven approaches).

- *Question 3: What are the framework requirements ("mandatory" and "optional" building blocks)? How the reference architecture of a generic adaptation framework would look like?*

  To answer this question we present the reference architecture of GAF by analyzing the existing AH systems, extracting the core AH functionality, And at the same time we bring new models, and aspects together defining truly generic adaptation framework which relies on the principle of separation of concerns and leverages AH with data-driven approaches. Thus the main goal of the *GAF* reference model becomes

to provide the reference architecture, to separate essential blocks from optional ones and define the criteria to distinguishing between these elements, also to provide a modular structure that can be used either separately or together, depending on the needs of the application or system designer.

- *Question 4: Is it possible to device a reference adaptation process out of the existing data models in order to clue together data representations the AH research field has been using?*

  To answer this question we consider the process aspects of AHS from the very first classical user modelling-adaptation loop to a generic detailed flowchart of the adaptation in AHS. We introduce a Generic Adaptation Process (GAP) and by aligning it with a layered (data-oriented) GAF architectural building blocks discussed in question 4.

- *Question 5: Does such a framework exist and how do we evaluate the framework?*

  To answer and discuss this question we construct a new way to evaluate AH architecture and models, similar to software architecture analysis approach which allows us to get in depths of reference modelling and system evaluation through the case-studies. The discussion of the answer to this question concerns a number of case-studies which were conducted. The systems studied covered a broad spectrum of personalization systems varying from social recommender system to a semantic web enabled personalized guide. The evaluation helped us to identify both advantages and flaws of the systems. At the same time having an evaluation feedback we could identify some weak points in the framework, which at the time couldnt be envisioned due to a fast developing field of personalization, and especially the variety of applications emerged recently.

- *Question 6: What are the technologies leveraging and extending the potentials of AH? How GAF-derived systems can use these technologies?*

  Here we show that GAF not only describes and helps to evaluate other systems as it was the primary goal of the framework, but also has a great potential in terms of system extensions and bringing a number of seemingly unrelated technologies together which help to leverage and boost adaptation. We touch upon provenance and versioning ideas here in particular.

1.4 Thesis Outline and Contribution

Considering the research questions and approaches undertaken in GAF, the thesis is structured as follows:

- **AH Field Revisited.** The first and very important step of the thesis is studying the existing adaptive hypermedia systems, web-based information systems, intelligent tutoring systems, intelligent agent systems and other related fields. This is done in order to create an inventory of methods and techniques used in such systems and

applications, to establish a common base and background for the new framework. This is the part where we also capture possible directions of any adaptive hypermedia system, so to some extent envision the development in AH research field. This AH study is covered in a survey paper [82] and is discussed in chapter 2.

At the same we are bringing together two (historically disjoint) views on the adaptation: a classical pre-authored type, with conventional domain and overlay user models and data-driven adaptation which includes a set of data mining, machine learning and information retrieval tools.

- **Reference Model, Architecture and Process.** After conducting a number of system studies (representative examples of AHS) and establishing the baseline we could distinguish mandatory and optional elements of the studied systems and essentially of the new framework, identify the principles of functionality and some basic interactions involved in the adaptation as a process.

  So, first and foremost GAF creates a common formal framework for describing current and future adaptive hypermedia and adaptive web-based systems in general. It provides a commonly agreed upon taxonomy and a reference model that encompasses the most general architectures of the present and future, including conventional AHS, and different types of personalization-enabling systems and applications studied.

  Thus we construct the framework from building blocks (or layers) primarily following the separation of concerns and functionalities idea. Then we match the adaptation process (discussed in section 3.3) with the layered structure (section 3.1). This adaptation process model shows inter-layer connections, information and control flows first of all in the AHS, and at the same time it easily matches the workflow representation of the conventional concept-by-concept adaptation. Thus we have a two-fold process presentation which can to some extent be derived from the AHS structure and which can also drive requirements for a modular AHS composition from a process-oriented perspective which was previously published in [80]. The existence of such a reference model and process basis should stimulate AHS research and enable researchers to demonstrate ideas for new adaptation methods much more quickly than if they had to start from scratch. The framework modelling, process and architecture description is presented in chapters 3 and 4.

- **Compliance**[1] **and Case Studies.** In order to bridge conventional AH and data-driven personalization systems we conduct a number of case-studies. This shows that many adaptive web-based system (including conventional AHS, and different types of personalization-enabling systems and applications such as recommender systems (RS) personalized web search, semantic web enabled applications used in personalized information delivery, adaptive e-Learning applications and many more) can be expressed by terms and definitions of our framework bringing all these fields together. Partial coverage of these compliance schemas can be found

[1] See section 1.5 for the extended definition.

in previously published papers [80, 64], while the full discussion is presented in chapters 5 and 6.

- **Analysis and Evaluation Approach.** We also perform a number of real systems' case-studies (continuing the framework compliance approach) and perform a detailed analysis and evaluation of the framework and systems being analyzed as a complimentary step. We analyze building blocks, functionality and architecture of these systems and applications in order to identify design alternatives, system composition, missing elements, improvements and new challenges for system designers and developers. To perform this type of system analysis we primarily base our conclusions of the compliance studies but at the same time develop a set of approaches based on the ATAM (Architecture Tradeoff Analysis Method) [73] software architectures analysis which helps to the in-depth blocks connections and interfaces analysis of the framework and corresponding systems. A brief summary of this approach has been published in [79] and is elaborated in chapter 6.5.

- **AH-enabling Technologies.** GAF deals with important and novel aspects of adaptation enabling and leveraging technologies such as *provenance* and *versioning*. In section 7.1 we re-think provenance aspects (such as data lineage and origin explanation), advantages and issues in terms of AH, by showing how the *GAF* framework and one of the most complete provenance models W7 perfectly co-exist, which means that 'provenance' aspects of *W7* can be re-used in *GAF*. And as a result - how GAF benefits from enabling provenance especially in the areas of adaptation process, UM scrutability and information reliability as well as reasoning and system behaviour analysis. The work on bridging provenance and AH has been presented in [85].

  Another concept rarely considered in the context of adaptation in the hypertext (and hypermedia in general) is versioning, which can be used to capture user dynamics such as re-visitations and re-searches on the web, context changes, etc. Versioning has been considered in papers [83, 84] where particular personalization-enabling use-cases are elaborated. As we show these adaptation enabling and leveraging technologies contribute a lot in adaptation and personalization in general which is going to be discussed in section 7.2.

A summarized outline of the thesis' contribution can be found in Figure 1.1. This figure also reflects the coverage of individual chapters. Overall the thesis covers a variety of AH areas, starting with the comprehensive overview of conventional AH systems, moving towards a process-oriented adaptation perspective, on to building a new reference model of a general purpose adaptive web-based system, enabling a number of new and innovative technologies for the AH field, and concluding with a new AHS evaluation approach with the help of the case-studies and overall system modules analysis. GAF will thus bootstraps any new adaptive web-based system research, design, implementation, analysis and evaluation.

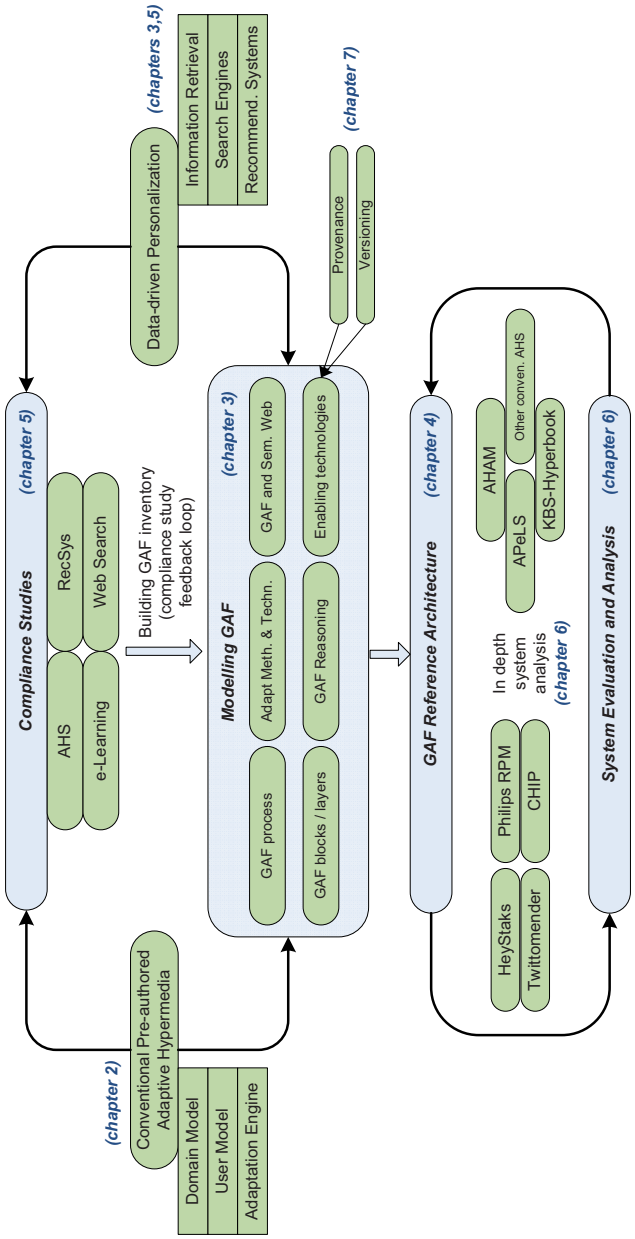


Figure 1.1: Thesis structure outline (including chapters' coverage).

1.5 Important definitions

In this section we would like to introduce a few crucial definitions that we use throughout the dissertation. These definitions give the main idea of an adaptation framework and a model and explain what is meant by an adaptation process and a compliance study.

- **Generic Adaptation Framework (GAF)** — is a generic purpose AH framework which provides the reference architecture, AH systems requirements to separate essential elements from optional and defines system criteria to distinguishing between these elements. It provides a modular structure that can be used either separately or together, depending on the needs of the application and that can be developed over a generic framework to satisfy different needs. GAF enhances adaptation capabilities by including new methodologies and techniques, facilitating more elaborate adaptation. Besides, it concerns the detailed system analysis and evaluation in terms of AHS building blocks, connections and dependencies, approaches that can be used to implement such a system.

- **GAF model** focuses on a layered architecture and discusses the adaptation of the content and navigation to the user properties as well as recommending the links to follow based on the user preferences and knowledge, thus bringing complementary aspects of adaptation, personalization and recommendation in a context of a generic framework which provides properties of information fusion and heterogeneity and serves as a reference model.

- **Generic Adaptation Process (GAP)** — describes interactions in AHS usually starting with the goal statement, exploiting features of the user and domain models in different contexts and adapting various aspects of the information and presentation to the user. GAP tightens the building blocks of the reference model and can be considered as the reference adaptation process.

- **GAF Compliance** — means the conformance between the GAF framework structural layered representation and any other given model, system or application functional breakdown, as well as the correspondence of the adaptation process steps (both flowchart and the 'sequence' chart representation). A system is compliant with GAF if it can be expressed/described in terms and definitions of GAF including functional building blocks, connections, and the adaptation functionality described by these blocks. Moreover we don't see the compliance as a one step procedure, it is an iterative process and each time we encounter a new (and generic) functional block in the evaluated system, it is added to the *GAF* reference 'architecture' executing a *GAF* evaluation-architecture feedback loop.

Chapter 2

Introduction to AHS: history of AH models and systems, AH taxonomy update

2.1 History of AH models and systems

The research field of adaptive hypermedia and adaptive web-based information systems (AHS for short) has been growing rapidly during the past fifteen years (since 1996 when the original AH classification has been introduced [22]) and this has resulted in new terms, models, methodologies and a plethora of new systems. Adaptive systems are becoming more popular as tools for user-driven access to information [23, 30]. Adaptation of an information system or service to a user has been proven to be a powerful and useful concept [29]. It is particularly helpful for the reduction of the information overload which is frequently experienced on the Internet or any other information system of a large scale. In parallel in 1999 a first reference model for adaptive hypermedia applications, called AHAM [47, 139] was defined, and an implementation closely following this model, called AHA! [45, 48], was made available to the research community. This reference model unified the adaptive hypermedia research community and provided a generic architecture that induced research activities in many directions.

Our first and foremost aim in this chapter is to provide a comprehensive overview of adaptive hypermedia methods and techniques since their introduction 15 years ago and at the same time also to make an overview of the requirements and a modular structure that can be used to update the first generic AH model AHAM that was introduced 13 years ago. And in order to create a full picture we're going to revisit AHAM model in section 2.10.

Quite a few systems were developed in the past 10 to 15 years, mostly providing facilities for eLearning (or Technology-Enhanced Learning as it is sometimes called) which was considered as a primary application area. Examples are KBS Hyperbook [66], APeLS [41], Interbook [27], WINDS [121], MOT [43], RATH [68], etc.

A few attempts have been made to extend the AHAM reference model or provide a new one. The Munich model [87] tried to capture all major parts of the system architecture

using the UML notation. The Goldsmith model (GAHM) [104] was later considered together with AHAM in an attempt to provide a unifying model of all three (AHAM, Munich and GAHM). The comparison from [57] in fact did not provide a unified description in terms of conceptual representation or adaptive techniques, bringing up mostly implementation and meta-data issues of those systems. Most of the new system developments have resulted in new terms, concepts, models, methodologies and prototypes (for example higher-order and open corpus adaptation), which still continue to happen a lot. All previously described ideas have been transferred to new situations, showing new use-cases.
Although AH research has delivered a variety of systems for the same application areas, there is still no consensus as to what is the "ideal" architecture of such adaptive systems. Each development introduced new components, new interfaces, new adaptation techniques, etc. In this chapter, pursuing the unified approach to AHS we will first consider adaptation questions initially raised by Brusilovsky (1996) with respect to the current state-of-the-art systems, giving explanatory examples of most commonly used AH systems and providing their specific details in comparison to each other. Secondly we will try to understand and extract the essence of each adaptation model. The following sections will cover the basics and granularity of a Domain, User and Goal Models, peculiarities of the Adaptation Model in sections 2.5, 2.6, 2.7, consider Context models in section 2.8 as terms of new developments and system decomposition. We will also take a look at Adaptive Presentation and Navigation techniques providing a taxonomy update in section 2.3.
As it is almost impossible to grasp all recently proposed and developed AHS, we will consider only the ones we think are most important and interesting in the field and we will also take a brief look at examples that are very representative and may show some specific characteristic of a particular system. In terms of models we look at the developments starting with the *Tower Model* [46], including *AHAM* [47, 18, 139], the *Munich model* [87, 88], *GAHM* [103, 104] and *LAOS* [43, 44, 65]. We consider systems that have a solid background in AH research and that continue to be subject of research and development, including *AHA!* [45, 48], *KBS Hyperbook* [66], *APeLS* [41], and *Interbook* [27]. We will also touch upon important and solid developments such as *TANGOW* and TANGOW-based systems [35, 34], *GOMAWE* [11], *CoMoLe* [93] and others in order to show clearly expressed differences or provide arguments for the new trends that we consider further.
As a result we will first sketch a modular structure for an AHS reference model that will capture the state of the art and the main new trends which may not yet be part of any AHS or may not yet be considered at all as a part of AHS functionality. We will also describe best practices and new research methodologies in the AHS area that proved their right to exist (being researched and implemented within a number of AH and related projects and investigated by a number of research groups both from AH and related fields). And finally we will describe an overall model (incl. process model) in chapter 3 and architecture in chapter 4 of the new Generic Adaptation Framework (*GAF*).

2.2 Questions of adaptation

The core of adaptation is defined by posing and answering six major questions [1]:

- **(What?)** What can we adapt?
- **(To What?)** What can we adapt to?
- **(Why?)** Why do we need adaptation?
- **(Where?)** Where can we apply adaptation?
- **(When?)** When can we apply adaptation?
- **(How?)** How do we adapt?

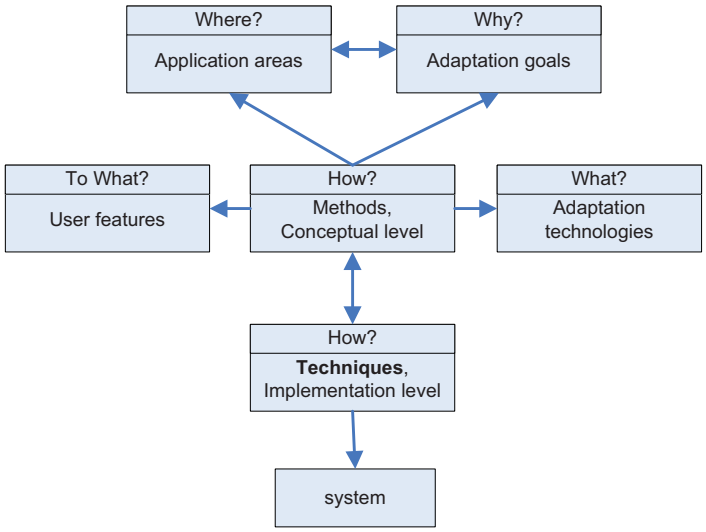


Figure 2.1: AH Methods and Techniques, Original Classification [22]

In the current section we revisit these questions (and corresponding answers), methods and techniques and also address the issue of aligning all the questions (and their answers) in a common, modular structure of a generic purpose AHS architecture. To this end we will also revise the meaning (or definition) of some of these questions in order to capture the most recent ideas.

[1] This type of classification has been initially introduced in [22] where a classification of AH methods and techniques was presented (see Figure 2.1). These questions are elaborated in sections 2.5 - 2.8.

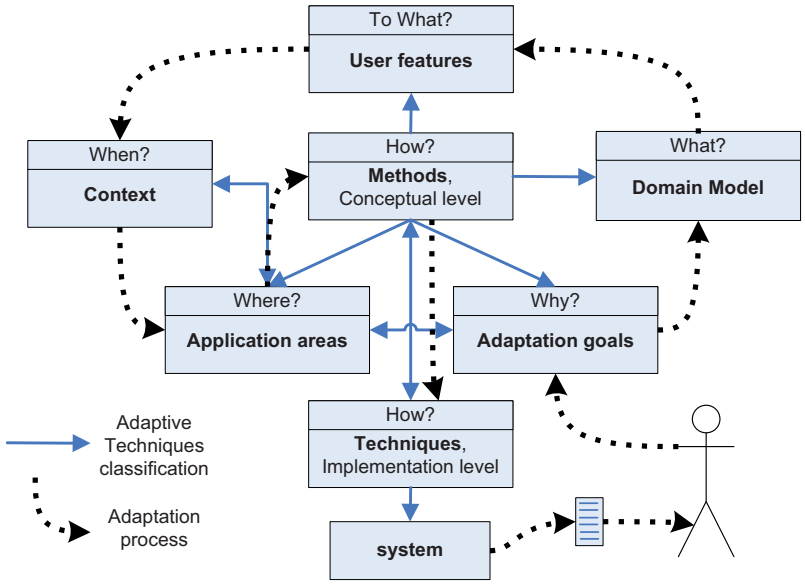


Figure 2.2: AH Methods and Techniques, Adaptation Process Highlights

Figure 2.2 considers the sequence in which the questions should be asked (and answered), thus leading to the definition of the adaptation process. By answering the major adaptation questions we elaborate the adaptation process description outlined in Figure 2.2. This process is usually initiated by the user stating the adaptation goal and thus answering the *Why adaptation is needed?* question. Then in the process we consider the *What?* and *To What?* questions, which emphasize the Domain Model and the User Model description. *When?* and *Where?* in this process go next providing context and application area definitions. And finally we come up with the *How?* question describing methods and techniques on conceptual and implementation level all together resulting in an AH system description.
Previous (reference) models acknowledged that the adaptation in a given application depends on three major factors:

- The application must be based on a Domain Model (DM), describing how the conceptual representation of an application domain is structured. This model indicates relationships between concepts and how they are connected to content presentation in terms of fragments, pages [18], chapters, information units [66], pagelets [41] or any other structure encapsulating information about a concept. DM usually answers a *What?* question, providing a domain structure and information that needs to be adapted, linking concepts to a corresponding content representation. In this case linking of a concept and content structures should be carefully considered as

a separate question as the way this linking is being done may affect the system architecture, from providing authoring tools to make one-to-one (concept to content) correspondence to bringing up dynamic aspects of open corpus and having a topic which resolves query linking, concepts and resources.

- A User Model (UM) has to be created and kept up-to-date to represent user knowledge, interest, preferences, goals and objectives, action history, type, style and other relevant properties that might be useful for adaptation. UM usually answers the *To What?* question, providing user and usage data using the information from DM. Quite often UM may answer the *Why?* question as well, providing information about user objectives using the same conceptual structure.

- The System has to adapt the presentation, the information content and the navigation structure to the user's level of knowledge, interest, navigational style, goals, objectives, etc. Thus the Adaptation Model (AM) has to be provided, indicating how concept relations in DM affect user navigation and properties update (for instance whether the system should guide the user towards or away from information about certain concepts). AM may be presented as a *teaching model* with pedagogical rules [47], a *pedagogical model* [66], a *narrative model* [41] or for instance including a glossary structure [27]. In terms of providing adaptation flexibility, this model may answer the *When?* and *Where?* questions, as well as bringing a *What?* question up again, interpreting constraints on a Domain Model relations structure.

This division into Domain Model (DM), User Model (UM) and Adaptation Model (AM) provides a separation of the major AHS questions. However this division is still mixing up some of the questions (since it only has three parts, for six questions). A further specialization of the *parts* or *layers* is needed in order to achieve a better separation of concerns and offer enough granularity in the architectural structure.

2.3 *How?*: Adaptive Methods and Techniques

Adaptive techniques and methods refer to methods of providing adaptation and their generalization correspondingly. Techniques are usually a part of the implementation layer of an AHS and can be characterized by a specific approach or algorithm. Methods represent generalizations of a technique. Every single method shows a clear idea of an adaptation approach, but at the same time each method can be implemented by a number of different techniques. Likewise some techniques may be used to implement several methods using the same knowledge representation. This set of techniques and methods comprises a toolkit of AH [22]. Both techniques and methods can be applied to content, presentation and navigation adaptation. Previously adaptation to presentation was not considered separately. In this section we distinguish adaptive presentation far beyond content and navigation techniques described in [22]. Some forms of content adaptation really only change the presentation, and some forms of adaptive navigation support do not change the possible navigation but only change "suggestions" by changing the presentation. We

decided to differentiate the three forms of adaptation and to present them in a single diagram in Figure 2.3.
The use of adaptive techniques has changed as AHS have matured. Especially in the field of education AHS have their origin in Intelligent Tutoring Systems (ITS) where all the adaptation decisions (like what to show to the user and which steps the user should take next) were taken solely by the system. Some adaptation techniques still enforce a system decision upon the user, like hiding a fragment of text or removing a link. But in AHS the trend is to offer users more control. This has resulted in the techniques that we show below as "adaptive presentation". They do not change the information or the possible navigation, but only use presentation variations to make suggestions to the user.
The use of particular adaptive techniques is also influenced by the increasing use of online assessment to more accurately measure user knowledge. Compared to early systems modern systems measure user knowledge as well as other properties more precisely [91, 37]. Therefore, having more precise measurements, more observable characteristics - AHS can use a wider range of techniques best suitable for each stored instance of information or user profile properties to provide better adaptation results.

2.3.1 Content adaptation support

The presentation of information can be influenced essentially in two ways: by showing/hiding the information or by emphasizing/deemphasizing it. The essential difference here is whether the information is accessible or not. When *inserting, removing or altering* fragments the information content is really changed. Other techniques: *dimming, sorting, zooming and stretchtext* keep the same information available but suggest to the user to only read part of it. This suggestion is made through changes in the presentation, which is why we also place them under "adaptive presentation techniques".
The techniques of *zooming* (which is one of the recent techniques, introduced in [131] and *stretchtext* are useful for additional explanations which need not be read by every user. We would also like to distinguish three different types of scaling/zooming technique. The one mentioned before is a conventional technique providing content (irrespectively to the information type) scaling or zooming, changing the text font, zooming in or out a complete web page or only a pictorial part of it, or scaling down images that appear in the presentation. A *fisheye view* allows us to have a different view on information content or a link structure, that's why we have associated this technique with adaptive navigation support as well. In a fisheye view certain details are kept visible (readable) whereas other details are scaled down a lot, aggregated or deleted entirely. The last one is a *fragment summarization* (for example *text summarization* when text is analyzed statistically and linguistically and a summary text is generated from these important sentences). In stretchtext only the title is shown whereas in zooming/scaling the entire information content is shown, but it may be scaled down (zoomed out) so much that it becomes unreadable using a conventional scaling technique. The user can decide to select/open the information so that it becomes readable (in full size). This is like placing a magnifying glass over the presentation that was scaled down. Accessing the information may also cause user model updates so as to influence the adaptive selection of zoomed information in the future.

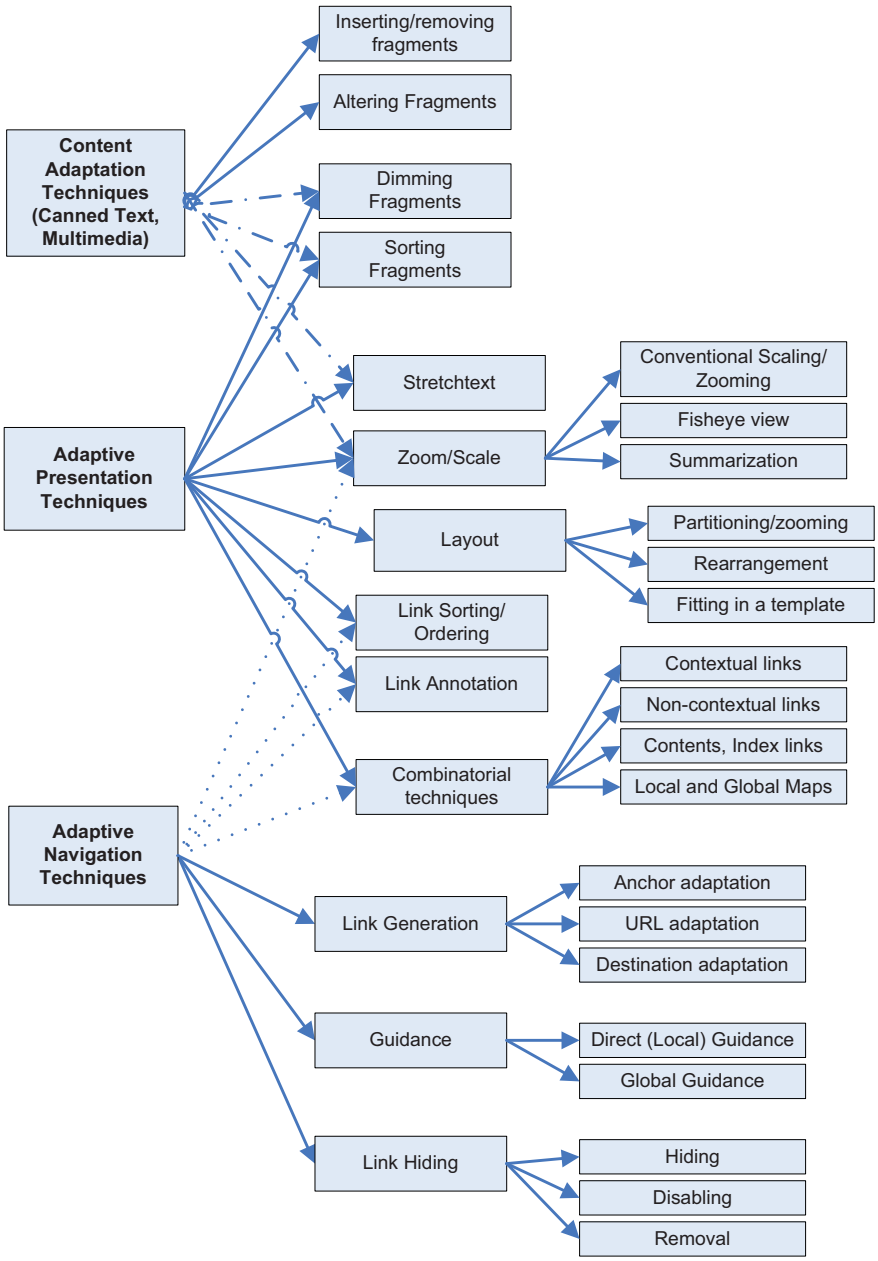


Figure 2.3: Taxonomy of Adaptation Methods and Techniques

2.3.2 Adaptive navigation support

The most recent revision of adaptive navigation instruments was done in [24]. That paper provides an extensive overview of adaptive navigation techniques and methods that are becoming increasingly important in various aspects of adaptive applications from web-based hypermedia to virtual reality. It reviews all major approaches, technologies and mechanisms giving illustrative examples. In this respect we will provide just a taxonomy of adaptive navigation techniques and mechanism used in AHS.

There are two ways in which the user's navigation can be influenced: enforced and/or suggested. The *guidance* techniques present recommended links, which can be obtained either through adaptively selecting links from a larger list (and hiding/removing the non-recommended links) or by generating destinations for predefined link anchors. In all these cases the structure of possible navigation paths (and links) is altered in a way that forces the user to select a link from a "computed" set of links.

Most adaptive navigation research focuses on adaptive navigation support that does not restrict the user but rather provides suggestions as to which links or paths are more appropriate than others. *Sorted lists* of links (placing the strongest recommendations at the top) and *link annotations* using colours and/or icons help the user in deciding which links are appropriate and which are not, but the user is not forced to follow these recommendations. The recommendations can be made by changing the content presentation, which explains why these techniques also fall under the "adaptive presentation" category.

Direct guidance using not just a single step but whole suggested paths were introduced for instance in KBS-Hyperbook, where users were provided with *guiding trails*. Adaptive link sorting is beginning to show up in personalized search engines. Link hiding, with its variants of hiding, disabling and removal, is most commonly used in the AHA! system. Link annotation is used in ELM-ART [31] and its descendents, including Interbook [27]. The link generation technique can be found in [140, 92], but is essentially also the technique used by *amazon.com* to provide its recommendations. A more complete survey of Recommender systems can be found in [2].

We can anticipate the use of three types of *link generation* techniques which may result in *anchor adaptation*, *URL adaptation* and *destination adaptation*. (All three are possible in AHA! for instance, but have mainly been used just to show their existence.) Initially introduced in [22] the *page variants* technique can be explained as a case of destination adaptation. The main difference between *URL adaptation* and *destination adaptation* is that with the former the decision as to which link destination to use is made when the page containing the link is generated, whereas the latter always shows the same link destination (URL), but when the link is accessed the server will decide which actual destination (or page variant) to return.

More powerful techniques can be defined as combinations of previously mentioned approaches to link adaptation. These are *contextual links* embedded into the context of the page, *local non-contextual links* which may include all types of links on a regular page (like links, buttons, lists, pop-ups and etc.). *Links on local and global hyperspace maps* provide graphical representation of local or global hyperspace navigational structure in a network form of nodes. The same approach of a global map structure can be seen in

providing linking from table of contents or index page, which in fact does support a kind of *pre-defined* navigation, but it can be useful in a particular type of applications.

2.3.3 Adaptive presentation support

As we saw above, adaptively changing the presentation can be used to either emphasize/deemphasize part of the content (that is all accessible) or to suggest links to users. However, there is also adaptation to the presentation that is applied for entirely different reasons, like device adaptation or layout preferences.
Layout adaptation can be needed because content (especially in open corpus applications) needs to be presented within a predefined presentation format. Research in the GRAPPLE [2] project focused on integrating adaptive learning environments (ALEs) which is an adaptive system supporting teaching and learning in an educational setting, with learning management systems (LMS), which are used to deliver, track and manage training process [19]. Depending on the LMS an information page to be presented may need to be placed in different frames/windows, and automatically generated view on the navigation structure may be included or may need to be omitted.
Another situation in which layout adaptation is needed is when adaptive presentations need to be adapted to devices with limited capabilities. A large presentation can for instance be scaled down, with the ability to zoom in to parts of it (one at a time), or the presentation may be partitioned into sub-pages that can be selected and viewed one at a time. Parts of information may also be presented within a predefined template layout which is reflected in presentation specification (for example using CSS web site templates with two columns and left navigation or just one column and right navigation).

2.3.4 Adaptive multimedia presentation

Nowadays a lot of photographic and multimedia content is described with extended metadata that can be used for/by adaptation. Moreover constantly extending image repositories, web services, tagging techniques, basic image operations which most of the devices are capable of, starting from computer software to embedded devices and internet applications have appeared. Even if these new technologies or image metadata are not available everywhere (e.g. on a handheld device) it is still possible to make use of image basics - width and height. Having a look at the aforementioned taxonomy of content adaptation we see the part that applies to adaptation in a multimedia context. The techniques that apply to textual content adaptation apply (viewed at an abstract level) to pictorial information as well.
We show a few use cases of adaptation to pictorial information below.
Conditional image inclusion may be quite useful in device adaptation, where only a key part (tagged with some concept; may be a thumbnail) or just a resized image will be shown on a small-screen device or a device with low bandwidth capabilities for example. In case of image resizing generating adaptive presentation becomes very simple since it

[2] http://www.grapple-project.org/

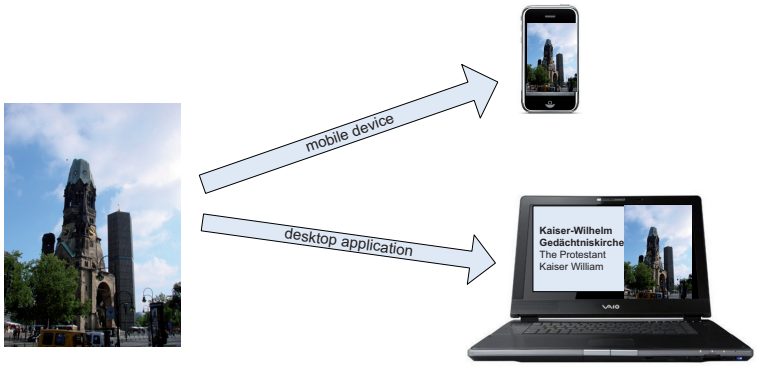


Figure 2.4: Conditional Image Device Resizing

Figure 2.5: Stretch Image to a Picture Set

doesnt require any extensive metadata from an image, but uses only image dimensions (see Figure 2.4). In this case the Zoom/Scaling technique is the best to be used.
As a stretchtext/stretchimage technique example we can think of expanding a single image to a set of pictures or a picture timeline, extending presentation to provide rich multimedia experience and fulfill curious user goals (see Figure 2.5) or just present a thumbnail of an image. In [122] adaptation to the visual/verbal learning style dimension uses the stretchimage technique.
Any tagged part of an image may be conditional (e.g. Flickr photo notes and tags or Facebook photo tags) and associated with a certain concept and thus may be included in presentation (cropping image); one image may contain several noted/tagged areas which may be show within different presentation specification. In Figure 2.6 you can see an illustrated article about Kaiser Wilhelm Memorial church which can be adapted using the conditional inclusion technique both for text and image. While authoring, not only pages are associated with different conditions, but in this case different parts of the whole image are also tagged (like a part with old and new church in a given example), thus fulfilling a certain condition (e.g. interest in architecture styles) results in different presentation both for image and description, while there is a need to store only the original image.
A geo-location condition can be applied as a rule used for presentation generation in the following way: on the one hand at any time point the user is either linked to or located in

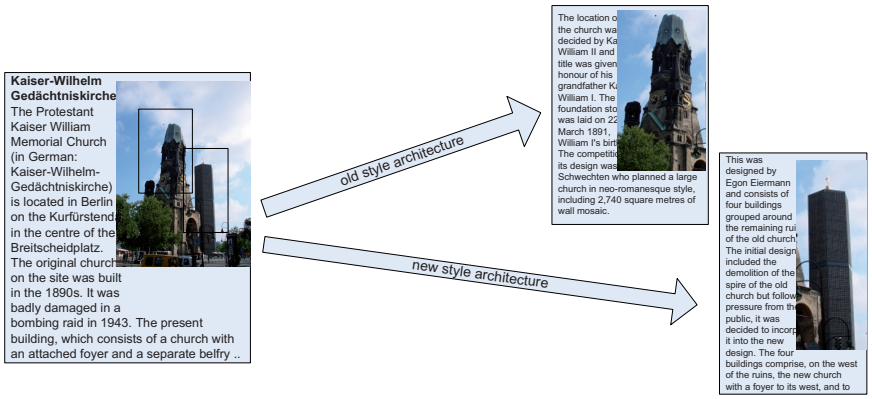


Figure 2.6: Crop Image Inclusion

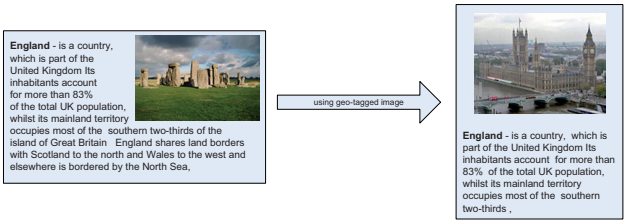


Figure 2.7: Geo-Tagged Image Inclusion

a certain geographical place, on the other hand more and more images especially photos are being tagged with geographical coordinates while taking them using GPS enabled devices or just doing this afterwards associating images with location where they were taken. Thus a new condition can be applied to any geo-tagged image: either to show an image guided by user current location/interest combination of values or adapt presentation taking user position into account. For example, presenting a general information about England and its historical heritage to a person who is currently located in London and just starts exploring England or someone who has already shown more interest in the capital rather than any other city. It is more obvious to present an article with a photo of *Palace of Westminster* rather than *Stonehenge* for example (Figure 2.7). In the first case we may just use a static rule for determining user IP address and associating image coordinates with a corresponding country, region, city; in the second case it is obvious to compare capital coordinates, probably some particular place or even area borders and a geo-tagged images to include them into a presentation using a geo condition to enrich the user experience or broaden their knowledge in a particular area.

The same *conditional* approach may be applied to video and audio information in exactly the same way: conditionally including video and audio content into presentation as a

whole video clip or audio sequence, but on the other hand it becomes possible to make conditional inclusions of video and audio fragments or combining both of them. For example depending on a user style and type and having a video clip as a content associated with a certain band - may result either in just audio playback of a track or embedding the whole video clip into a presentation if the user is interested in videos within this domain rather than just listening to sample songs. Since video and audio information can be presented as a sequence of frames (audio samples) one may be interested in playing back not the whole sequence, but just most interesting moments or listen to an audio fragment. Introducing authoring techniques for tagging/labeling different video sequences or having possibilities to extract meaningful fragments from video or audio content and binding them with different concepts may result in different presentation. For example you are fond of TV quizzes and appear to be not very patient to watch the whole episode and if all the endings of your favourite quiz shows are marked or tagged, then only final minutes with answers will be presented to you and you won't waste your time watching opening credits and probably boring gameplay (e.g. like bookmarking particular chapters of your favourite DVD movie).
In [62] there has been an attempt to extend the taxonomy with multimedia components. It has been presented as a number of multimedia components used by the altering (fragments) technique, such as: models, views, controllers, widgets, graphics items, scripts, strategies. Providing component alteration may result in a system that can change its internal representation (model), its specific view, and/or one of the controls. Altering similar widgets may change the user interface, as well as changing different graphical items.

2.3.5 Media Adaptation Abstraction Layer

By analogy with the text frame-based techniques, all illustrated or pictorial content related to a particular concept is represented in a form of a media frame or a set of frames. Slots of each frame can contain several images and annotation variants of the concept, links to other frames, extended textual description, examples, etc. Presentation rules are used to decide which slots should be presented to a particular user and in which order or at what location.
Having reviewed all known to us presentation techniques and mapped the existing taxonomy on multimedia content showing the use-case and providing examples of techniques application we came up with a text/multimedia cross-techniques taxonomy (Figure 2.8). This cross-classification allows us to create an abstraction layer of presentation techniques used across the variety of AH systems (Figure 2.9).
Having matched adaptive presentation techniques classification on video and image content, one may think of creating an abstraction layer. That becomes an ultimate goal to make particular content presentation methods independent from techniques used by an adaptive engine (AE). In case of a reference architecture and in terms of the GAF project this abstraction can be found very useful. DM authors may describe adaptation rules irrespectively of content type and low-level techniques, and different system designers may use particular presentation techniques to mach their content type. A draft scheme of such an abstraction can be found on Figure 2.9.

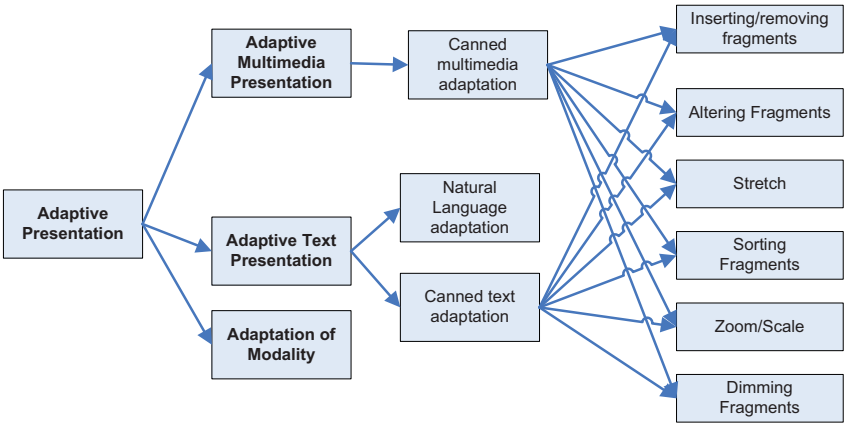


Figure 2.8: Adaptive Presentation Techniques Taxonomy (Updated for a Multimedia Content)

Here are the key points of the abstraction layer:

- The Adaptive Engine uses abstractions of presentation techniques:
  - it sends an event to provide adaptive presentation and a link to a content;
  - depending on a content type, a presentation technique selector parses content properties, methods set by a domain expert (if any) and chooses a method (e.g. resizing for image or font decreasing for text and so on.);
  - the same techniques are provided within authoring tools to isolate authors from describing each content type processing technique/method (only high level operations provided irrespectively of content type; however each content piece should be annotated with own attributes e.g. labels, areas, tags and etc.);
- Adaptation presentation techniques used by AE are independent from particular content handling methods;
- Info properties are used by presentation techniques to perform adaptation (e.g. labels, tags, fonts, size and etc.). For each operation there are certain properties used as parameters;
- Presentation techniques of a particular information type are mapped on a high level techniques repository - each content type may have its default methods;
- Content type recognition allows to identify content type by its URI or metadata and map on a correct implementation method;

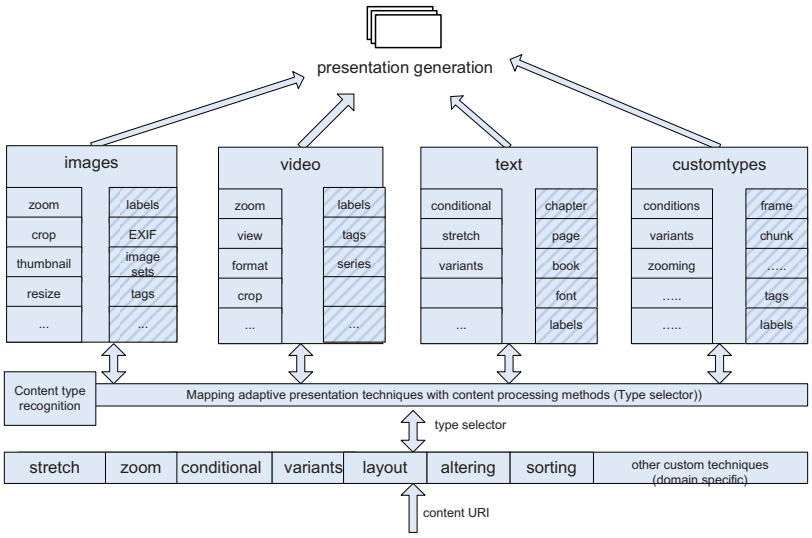


Figure 2.9: Adaptive Presentation Techniques Abstraction Layer

- Domain experts are independent from connecting different content type method with authored content, they only state high-level techniques they want to use (basically it is a techniques taxonomy) irrespectively of content type (image, text, or even a combination of them);
- Content authors use the same list of high-level techniques for all authored contents;
- Abstract techniques are independent from content properties, which allows domain experts and system developers to introduce and describe new information types and corresponding methods;
- Pieces of content and content properties (blocks with hatching) are connected with a particular concept in a domain model (DM).

2.3.6 Tools Adaptation

Nowadays AH applications include not only content to be read, but also tools to interact with different resources. Though the question of tools adaptation is very specific and usually treated in a context of every single application. Apart from application adaptation it may play an important role in the user modelling and adaptation process. Usually tools adaptation results in providing a different set of features to the different types of users: novice vs. advanced users or group of users. For instance tools used in collaborative workspaces [34] can be either selected or adapted to support collaborative task accomplishment. Also, in the AES-CS system [130] field-dependent and field-independent

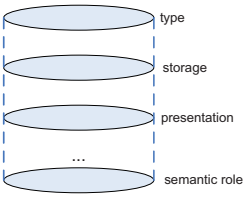


Figure 2.10: Tower Model

users were provided with different orientation support tools (such as a concept map and path indicator). Despite the fact that tools adaptation is still a very specific field, a generic adaptation model should be extensible to accommodate tools adaptation techniques and available instruments in a single, yet versatile and universal adaptation process.

2.4 AHS Reference Models

Having considered the core classification of AH methods and techniques, we now have a closer look at some of state-of-the-art AH systems. We also look back at the beginning of the era of *layered* reference models, models of hypertext systems in particular and systems from adjacent fields, and eventually draw a conclusions about the layered composition of *GAF* which we are going to take into account first and foremost due to *separation of concerns* reasons. It is of a great importance to consider an AH classification of methods and techniques (essentially distinguishing all major adaptation questions) if we want to make the *GAF* architecture flexible and extensible in order to accommodate and be able to describe diverse adaptation functionality. Besides that, all the building blocks (or layers) should be made independent (of each other) in order to describe a greater variety of models and systems and create a truly generic reference framework.
Reference models started having a *layered* architecture with the Dexter Model [60]. In 1992 the Tower Model was introduced [46, 133]. The Tower Model was an extensible data model for Hyperdocuments intended to serve as the basis for integrating hypermedia systems with other information sources, such as DBMS, IR systems, CAD tools, etc. To this end it had functional structures that can express adaptive and dynamic hypertext systems and applications. The Tower Model considered a layered structure, just like Dexter, but considered very explicitly the view (or projection) of each individual object through the individual layers, which led to the definition of the Tower. The model provided definitions of nodes, links and anchors as first-class citizens, and offered modelling constructors to build complex information representations, such as composite objects and cities.
The tower constructor packaged together the multiple levels at which an object was described. These levels would include, among others, a structural description level, and a visual presentation level. For example of a text node tower description in Figure 2.10.
These objects and functionality within the hyperspace were multidimensional, encapsu-

lating different aspects in a tower from an issue or a problem solution to a graphical rendering or a text representation. These different dimensions corresponded to different levels of a hyperdocument description and belonged to a different conceptual spaces, resembling one of the goals we stated (to achieve a clear layers separation in a generic system). As a more advanced structure the City comprised by a number of towers gave an opportunity of viewing a hyperdocument from different perspectives. The tower and city constructs can be considered the basis for later models like LAOS. Having presented a set of modelling constructs that made it possible to integrate a wide variety of information sources into a hypermedia systems, the Tower Model predicted and provisioned the structure and dynamics of AH systems, indicated that a layered structure of hypertext systems that can be used to provide flexibility and interoperability of the system within different concept spaces.
In the following models and systems like AHAM, LAOS, KBS-Hyperbook, APeLS and others we can clearly identify where the major adaptation factors (and therefore 'layers' or system building-blocks) belong. The AHAM model layout matching the basic adaptation questions is presented in Figure 2.11. We can do the same for the UML based Munich model (presented later in Figure 3.11), the LAOS authoring model in Figure 2.13 and APeLS system architecture in Figure 2.14. All figures underline the presence of major adaptation factors in each model and to some extent represent a layered or block-based (which essentially can be presented as layers, each responsible for a particular functionality or representing a model or a system component) structure according to these factors.

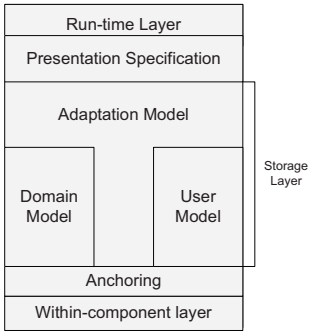


Figure 2.11: AHAM - Adaptive Hypermedia Application Model

In order to perform adaptation based on Domain and User knowledge an *author* is required to specify how the system interaction results in different information presentation units based on DM. In AHAM, this is done by means of an adaptation model (AM) consisting of adaptation rules. An adaptation engine (AE) interprets these rules to handle link anchoring and to generate the presentation specifications. AHAM uses *ECA* (Event-Condition-Action) rules to describe the UM update and the adaptation processes, without requiring represented systems to actually do the same. In [139] the associated problems of termination and confluence problems have been considered, proposing static analysis of rules and a simple strategy for dynamic enforcement. The adaptation engine in the AHA!

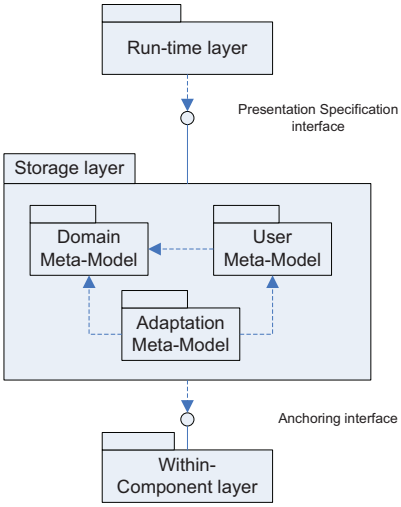


Figure 2.12: Munich Model

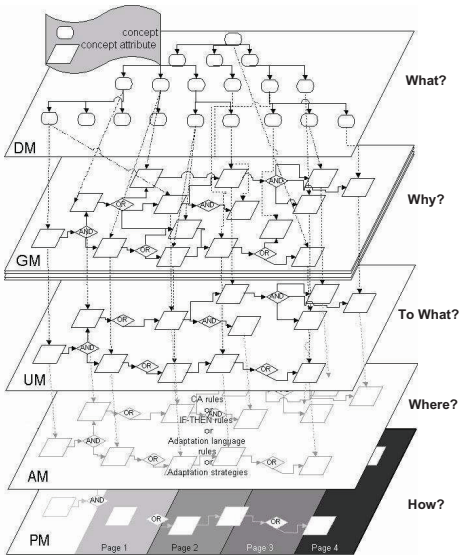


Figure 2.13: LAOS Model

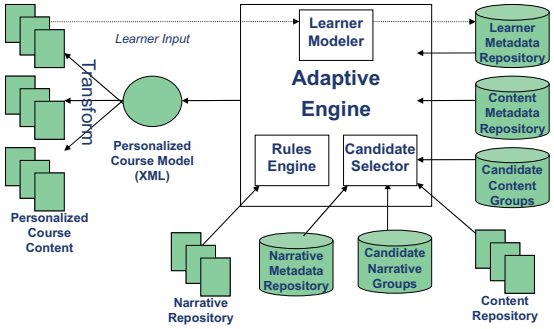


Figure 2.14: APeLS System

systems uses ECA rules. The same reasoning approach is applied in KBS Hyperbook, introducing also deduction rules, which are based upon the object oriented conceptual modelling *Telos* language. The APeLS system uses *JESS* (Java Expert System Shell) which in fact represents facts that make certain rules applicable and then asserting them, which is of an ECA reasoning type.
Since ECA rules are low level they are difficult for authors to understand. Therefore some AHS provide authoring tools that hide the actual ECA rules and offer higher-level constructs, which correspond to "concept relationships" and "concept relationship type" in AHAM.
AHAM Rule Example:

This (generic) rule specifies that all relevant fragments (F) of the page (P) (where Fragment(P,F) is a predicate) will be shown to the user on access:

Condition: **select** P.access
Action: **update** F.pres == 'show'
where Fragment(P,F) **and** F.relevant = **true**

There can be also a work-flow based constructs, concept type based rules or programming based constructs such as LAG or LAG-XLS languages [42, 122].
Now let's choose the most representative AH examples of the past almost two decades and consider their modelling approaches by decomposing each system and studying their elements and functionality, and each model (Domain, User, Goal, Adaptation, Context models) in detail.

2.5 What?: the Domain Model

The *Domain Model* of an adaptive hypermedia application usually consists of concepts and concept relationships. A concept represents an abstract information item from the

application domain. In most of the systems the concepts form a hierarchy. As a result each concept can be either an atomic (primitive) concept or a composite concept that has child concepts (sub-concepts) and a description of how they fit together. Some systems and corresponding authoring tools allow graph based approaches. More complex ways of connecting concepts are also possible, as is often done in defining subject domain ontologies.

In this section we investigate a number of conventional AH systems in detail in order to identify trends and build a set of tools to construct a skeleton of DM for a new reference model. By studying basic and common DM elements of these systems we identify the core DM elements which are later extended by introducing the process notion and augmented with data-driven, semantic web and other features which allow us to cover a wide range of web-based personalization systems.

This new DM being built is discussed later on in chapter 4 in the context of the *GAF* architecture. We also come back to a question of DM modelling and representation in section 3.3 where process-oriented aspects are discussed. In section 7.1 we describe provenance properties of DM and in section 3.10 (table 3.2) elaborate semantic web features and approaches of the domain modelling. Later we revise our views on DM modelling after conducting a number of compliance studies in chapter 5, and even further update it after a number of real case-studies in chapter 6.

In many AHS concept hierarchies and their representation may vary from system to system, providing indexing facilities like in Interbook [27], mapping domain concepts onto a document space which contains documents and test items (and the concepts themselves). Each textbook is structured as a hierarchy of chapters and sections with atomic presentations, tests or examples. Interbook applies adaptive navigation support (but no content adaptation). The same hierarchical presentation can be traced in KBS Hyperbook [66, 67], where the system uses a knowledge base which consists of so-called *Knowledge Items* or essentially concepts. In this respect each document from the document space is indexed by some concepts from the knowledge base which describe the content representation and hierarchical structure. In APeLS the concepts are encapsulated into a Narrative structure where each narrative can be hierarchically split into sub-narratives.

Providing this type of Domain Model structure (where all concepts are fine grained and hierarchically structured down to low level representation primitives) makes it possible to apply adaptive techniques, working with fragments of fine-grained information units representing each concept and making adaptive presentation and navigation come into play. In general structures follow the same scheme of concept hierarchy (presented as a directed acyclic graph), providing an arbitrary number of object enclosures. This allows to apply adaptive techniques directly to a low level structure of fragments and pages performing user adaptive navigation and presentation support. Each system proposes its own way to encapsulate content information: in a form of a *Pagelet* (in APeLS), which contains content and a content model, representing general, pedagogical and technical information and which may be assigned to a certain content group. Or it may be an Information Unit just encapsulating content information as in KBS-Hyperbook. And these Information Units are indexed to map the Knowledge Items structure. In the AHAM model and in the AHA! system content representation is based on pages consisting of *fragments* (see Figure 2.15).

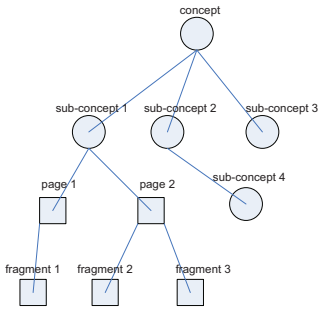


Figure 2.15: Domain Model Concept Hierarchical Structure

Whereas most systems have a fixed concept structure in TANGOW concept structures are reconfigured in different ways according to the rules and depending on the user for which this website/course is intended.
A concept relationship is a meaningful relationship between concepts. In AHAM it is represented as an object (with a unique identifier and attribute-value pairs) that relates a sequence of two or more concepts. Each concept relationship has a type (e.g. direct link, inhibitor, 'part of' or prerequisite) which may play a role in the adaptation. Such a Domain Model structure representation applies to most AHS systems. In KBS-Hyperbook we may see the dependency graph of all the KI's (knowledge items), in AHA! we have binary relationships of arbitrary types [45], and in APeLS we have a form of relationships map in a Narrative Model. In some of the systems or models (for instance LAOS) the 'prerequisite' type is withdrawn from DM as it is more related to a certain variant of content interpretation.
In the Munich reference model [87, 88] a more formal UML notation Domain Model view can be found, presenting all relations and entities in terms of UML associations, compositions, interfaces, links and packages, providing a formalized overall intuitive visual representation and a formal unambiguous specification of an Adaptive Hypermedia System Model. Two basic classes of a Domain Model are Component and Domain. The Component structure is represented by an abstract Component class, that can be either a Concept (class Concept), which is in turn can represent Atom or Composite class or concept relationship (class 'ConceptRelationship').
Considering yet another generic purpose AHS model GAHM we can see that here personalization is essentially carried out by handling hyperpages, which are defined to be a sequence of certain chunks, each of which is comprised of a content specification or so-called *C-Spec*, which may be presented in a form of data values or requests to a database and may be associated with a set of template variables, which can be marked as a placeholder for the content. Carefully considering this combination of *C-Spec* and *R-Spec* or rendering specification (which in turn describes how content has to be rendered) we may conclude that the aforementioned specifications to some extent can be mapped (by providing a description of functionality overlap of each system sub-components/models) onto

the AHAM and Munich models. With respect to a Domain Model template variables and content specifications represent the conceptual structure of a Domain.
The (above) typical approach at defining a domain model as a set of concepts and concept relationships does not take into account dynamic aspects such as the construction of goals or tasks as structures over domain model concepts. Since the adaptation is moving towards more intelligent processes taking into account user interaction towards certain objectives, perhaps following a certain workflow in a highly dynamic context we see an emerging need for a separate model or layer to handle the *Why?* question, which we deal with in the *Goals and Tasks* section 2.7.
Another trend is to attempt to utilize the Domain Model as an ontology or vice versa. For instance providing an Integration Model (IM) and Integration Model Ontology (IMO), which allows specifying a Domain Model and ontology mapping as mentioned in [8, 136]. Alternatively the GOMAWE system [11] proposes a model based on a semantic data representation which can be easily utilized in process automation and knowledge reuse across applications. AHAM can almost handle the single ontology case (because it considers concepts and arbitrary concept relationships), however it has no provisioning dealing with multiple ontologies. In this respect considering the semantic web reasoning approaches is becoming more challenging than was initially thought. Research into reasoning over different ontologies is becoming one of the core AH research areas, because one cannot assume that different applications of which the adaptation must be combined are using the same ontology [8, 136, 33, 11, 50].
In general the Domain Model is considered to be a static structure being defined and authored by a domain expert, which implies that adaptation can be provided only within the bounds of a Domain modelled knowledge space. However the move towards open corpus adaptive systems defined in [28] is becoming one of the challenges in the AHS research field. It aims to extend AHS with the possibility to operate on an open corpus of documents, which is not known at design time and in addition to this can be constantly changing and expanding [25].
If an AHS has to deal with the *open corpus* document space the problem of mapping concept(s) to content arises. Having a great variety of content structures we may have:

1. one-to-one concept to content matching;

2. selection of content resources which results in one-to-many relations;

3. a link which is represented in a query (concept query) (e.g. topic resolving query) to provide content resources to be mapped with a certain concept;

4. or one resource may be a (partial) match to different concepts

Even though when combining just two Domain Models the task of reasoning is becoming challenging, in case of *open corpus* it becomes even more difficult in terms of concept and content alignment. As the *consensus* as to how concepts and content (resources) match may change over time the concept to content mapping problem is related to the research topic of concept drift [97, 132].

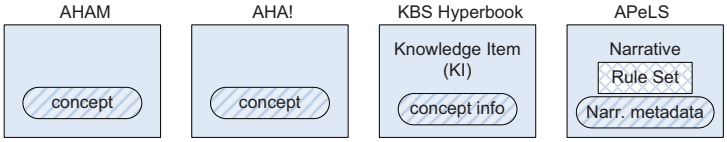


Figure 2.16: Domain Models' Information Unit (IU) Structure

Having done a brief overview of core adaptive functionality in terms of rules (basically represented by ECA type rules) which are interpreted by an adaptive engine to deliver user navigation and presentation support, we didn't mention a challenging idea of higher order adaptation. Though most of the systems adapt to one parameter (recommender systems adapt to what they think the user's interests are, learning systems adapt to what they think the user's knowledge is, some systems perform device adaptation), more advanced systems can do adaptation to more than one parameter at once or can *monitor* the user in order to decide to change they way in which the adaptation works. For instance in [122] the user's learning style is monitored, and as the observed learning style is detected (or changed) the way in which the system adapts also changes. More in general a second order adaptive system would use machine learning techniques to discover usage patterns and adapt the way in which it adapts to the user or provides the following information to a domain expert for more accurate refinement. In general there may be no limit to adaptation orders: a system may learn how to adapt the way in which it learns how to adapt its adaptation strategy, etc.

In Table 2.5 we present a summary of the Domain Model functionality and specification approaches used in AHAM (and the AHA! implementation), KBS Hyperbook, APeLS and Interbook, as the most representative comparison examples, even though we realize that other systems with some additional properties exist. Each row in the table presents a description of a particular system, its properties and aspects which we consider describe more or less the same system functionality in comparison to other systems. On the other hand the table shows all the differences both in approach, implementation and composition of each system, as well as a difference or similarity in terms used to describe system functionality. E.g. *Content Grouping* (present only in APeLS and KBS Hyperbook) is implemented either in a way of grouping similar content pagelets in APeLS content groups or grouping a sequence of concept and associated content in Project Units fulfilling similar user objectives. In other systems grouping is not possible which we denote by *n/a*.

To augment the table we present similarities and differences in the structure and representation of the *Information Units* in Figure 2.16 and *Pagelets* in Figure 2.17 in the aforementioned systems.

As we can see both concept and content information are encapsulated in the same way. They represent the concept information irrespectively of the concept attributes, which may vary in some systems. And they also contain the content information which in most cases has a hierarchical structure and one parent node may consist of a several fragments of the same content or the sequential structure (e.g. certain type of narratives). A sequence here

Table 2.1: DM properties: concept related

..	..	AHAM	AHA!	KBSHyperbook	ApeLS	Interbook
Concept	An abstract representation of an information item from the application domain	Concept information (attribute–value) pairs, sequence of anchors, presentation specification, atomic concepts represent single fragment of information, composite concepts use child attribute to specify sequence of composite concepts	Concepts like in AHAM with restrictions also have types; and associated with a template (which can have only fixed number of attributes)	Knowledge item (KI) abstract representation of domain knowledge (e.g. if, class, run_method) (may also be a compound structure)	Encapsulated in narrative model metadata (each narrative may add a new concept and corresponding narrative rules)	Glossary entries are domain concepts
Concept Relationship	Represents semantic relationship between concepts	Authored semantic linking between concepts in a form of: (C1, C2, T, A) where (T–type): link, prerequisite, inhibit, part (compositional) and (A–attribute value)	Arbitrary types of relationships: fragment, link, contain	Dependency graph of the *KIs* denotes semantic links between information units (IU). Each IU is connected to one or more KI presenting which concept represents corresponding content in IU	Is presented in a form of a relationships map in the narrative model	Concept relationships (navigational paths between glossary items) types (1) first-page (2) sub-section (3) domain-concept (4) bookset (5) login-page (6) requirement (7) outcome and (8) fragment.

Table 2.2: DM properties: content related

		AHAM	AHA!	KBS Hyperbook	ApeLS	Interbook
Indexing	Explicit indexing options: mapping concepts, projects, etc.	n/a	n/a	Knowledge items (*KI*) index Project units and information units	n/a	Glossary entries index domain concepts. Concepts are indexed on textbooks (bookshelves)
Content data presentation	Content unit structures	Pages and fragments (page may consist of several fragments)	Pages and fragments	Information units (IU)	Content information is presented in a form of a *pagelet* which may belong to a certain content group (see Figure 2.17)	Content info is presented in a Textbook (shelf of textbooks). Glossary (glossary entries provide link to a certain textbook and connection to a certain domain concept)
Content grouping	Content grouping according to similarity of presentation, objectives, etc.	n/a	n/a	Project units are mapped on information units. Project unit defines a number of *KI* that has to be learnt to fulfill project goal	Content group (content *pagelets* are organized in a group fulfilling the same learning objective (LO))	n/a
Storage	Content storage part (layer) of the AHS	Within-component layer	Within-component layer	Domain model	Content domain	Textbooks/textbook shelves

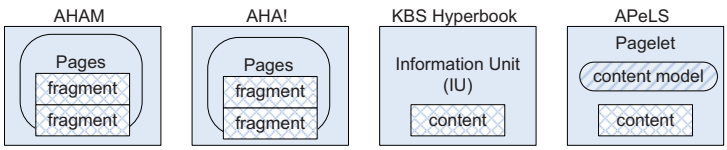


Figure 2.17: Domain Models' Pagelet Structure

means one-by-one concepts adaptation and corresponding content delivery depending on the narrative (project to follow, goal, set of goals, etc.).
Having investigated a number of state-of-the-art AHS we now have a picture where to start building a DM for a new reference model, we have created an toolset and cleared out a number of promising and prospective directions which we are going to exploit further when constructing the *GAF* architecture and describing all the nitty gritty details of the framework.

2.6 To What?: the User Model

Like we did in the DM investigation in the previous section, we move on and consider the User Model (UM) in the same way. We now study basic UM composition and properties used in conventional systems applied mainly in the e-Learning domain. In the following chapters after considering a number of related aspects such as a process-oriented perspective in section 3.3, provenance oriented aspects in section 7.1 in the user modelling as well as the semantic web aspects in section 3.10, we will present the overall UM architecture in the context of the *GAF* framework in chapter 4. Hereafter in the section we perform a preliminary analysis of the UM structures and composition based on the classic systems and models from the AH field as well as give an insight on the prospective UM solutions.
As an initial approach the adaptation process in adaptive systems was made based on user characteristics represented in UM. Since that time many systems used their own approaches and/or adapted to something else rather than user characteristics. In [86] Kobsa suggested to distinguish adaptation to user data, usage data, and environment data. User data points the way towards the adaptation goal. Usage data is comprised of the user interaction that still could be used to influence the adaptation process. Environment data comprises all aspects of the user's environment that are not related to UM or usage process or behaviour.
UM usually consists of entities for which we store a number of attribute-value pairs. For each entity there may be different attributes, but in practice most entities will have the same attributes. Therefore, it can be a table structure, in which for each entity the attribute values for that concept are stored. Most entities in UM represent concepts from DM. Some entity instances may represent a user's background, preferences, interest, learning style, or even a platform or environment specific properties.
Usually the analogy between the structure of UM and DM is that UM contains an overlay

structure over DM, mapping the user's domain-specific characteristics like knowledge over the domain knowledge space. This is typically done by associating attribute values with each identifiable piece of user knowledge, interest or other characteristic for each concept of a given domain. When considering different aspects about different concepts the table representation (using a universal instance) would result in a sparsely filled table, but alternative implementation structures do not suffer from this problem.
In general we have domain dependent and independent properties. Domain dependent properties usually are: user knowledge, test results, learning objectives, problem solving tasks or short term objectives. Domain independent properties are: user credentials, preferences, cognitive and learning styles, user environment (time, place, equipment, etc.) and group affiliation if any. In this respect user experience or background can be considered as domain independent properties, however in case of overlap between domains background knowledge may be fitted again within DM structure.
Dealing with an Overlay Model, the LAOS model tried to eliminate the classical UM overlay structure by avoiding hidden adaptation rules, representing UM as a concept map in such a way that relations between the variables in UM can be expressed explicitly (without the need to express UM concept relationships as DM relationships). LAOS also uses GM (Goals and Constrains Model) [44] (uniforming ontological representation [90]) to express goals and constraints separately from DM.
From another angle user properties are considered as static (covering user personal characteristics, such as age, gender, grade, or capabilities) and dynamic (which is the information about user interaction with the system like knowledge, skills, motivation, plan, activity or goal). An AHS must handle static and dynamic UM properties in a different way: it can just use static properties but it must monitor (changes to) dynamic properties as well as use them for the adaptation.
Considering this generic and quite popular overlay approach we may easily identify that for instance in KBS Hyperbook the user is modelled as a current state of his/her knowledge snapshot at each time (overlapping DM KI vector structure). The Learner Model in APeLS is authored to meet a DM structure and is represented in a set of concepts and user knowledge of each concept. It may also contain user prior-knowledge, learning style and user objectives. However we will consider the user goal question separately in section 2.7 to have a clear separation of AHS system layers (similar to the LAOS system). There are no strict rules saying that we can't use task representation within UM and treat it together with user characteristics. However we should first consider the question of system usability and therefore treat task representations and application models separately, in order to pursue system models independence and interoperability.
We may also identify the property reflecting the way knowledge is deduced and stored in UM. Most of the systems (like AHA!, Interbook) use a conventional scheme of updating knowledge level basing on the DM concept competence and keeping it in a UM overlay. Others (KBS-Hyperbook) use a probabilistic approach by means of Bayesian network calculating the conditional probability that knowledge x is known to the student under the condition which is denoted by previously detected information about this student.
As defined in the AHAM reference model the user model may also consists of a persistent part and a volatile part. For each concept attributes of which the value is maintained were

considered (for instance page was read or what is the level of knowledge). In this respect an AHS could recalculate some other attribute values on the fly. Some AHS may verify prerequisites satisfaction for a concept each time it is accesses or when a link to it is shown (backward reasoning), while another AHS may calculate and store prerequisite satisfaction each time it changes, for instance as a ready-to-access attribute in UM. For the future we foresee a new scenario where the AHS may already pre-compute (and store) the UM states that would result from future possible interactions like following a link. This thus allows the system to serve adapted information more quickly than when the (forward or backward) reasoning only starts when the user actually performs that interaction (follows that link). A smart system may predict the most likely future interactions and pre-compute several steps into the future (almost like what a chess program does). From a model point of view this can be considered as performance optimizations. In future however the predictions may also be shown to the users at which time they will influence the interaction and thus also need to be incorporated in the model of the system architecture.
In the tables 2.3 and 2.4 we present a summary of domain dependent and independent UM properties as they are presented in state-of-the-art conventional AHS models/systems: AHAM (and the AHA! implementation), APeLS, KBS-Hyperbook and Interbook. We also consider goals and objectives here as a part of UM, though in the following section we discuss a question of Goal model separately. Note that we concentrate on the representation of the user model, not on the process of obtaining or even deducing that information, which is covered in chapters describing the adaptation framework modelling in chapter 3 and architecture in chapter 4.
Having done the same we did in a previous section with DM gives us a good reference point to start constructing a new UM for *GAF*. In the context of a new reference model it would cover not only the early (described in this section) and current developments in AH field, but also extend it to a whole new level and capture relevant features from adjacent fields in order to contribute to a truly generic *GAF* reference model.

2.7 Why?: Goals and Tasks

In the previous section we already mentioned that the user goal can be considered a user property that can be stored in the User Model. However, we also saw that this is not the most natural approach. When considering goal-driven adaptation in existing AHS we cannot achieve good adaptation by just considering goals as user model properties. A goal is becoming not just an objective that has to be fulfilled, but evolves into a hierarchical structure of goals, objectives, tasks, requirements, workflows, depicting more task oriented and procedural approach. A *Goal Model* thus deserves to be a separate part of any up to date AH model.
An attempt to catch goal-driven adaptation has been made in KBS Hyperbook, where user defined or proposed system tasks have been mapped onto *projects'* units, each representing an index of *knowledge items* (essentially concepts presentation in the system), providing an elaborated task approach, where *projects* meant to be real application issues that can be faced by performing a certain sequence of tasks (learning in terms of the

Table 2.3: UM properties: domain independent

		AHAM	AHA!	KBS Hyperbook	ApeLS	Interbook
User goal and objectives	Overall learning goal stated by interaction with user	User follows a link to a (different) page	User follows a link to a (different) page	(1) for direct guidance (2) for goal based learning: knowledge items (KI) to be learned are selected by user goal (with triggering event for AE) consists of KI array (3) for project-based learning: goal and project repository	Learning objective – state the goal of learning procedure	User stated / assigned learning goal
User goal statement	Goal statement by the user	n/a	n/a	(1) user defined (2) proposed	(1) user defined	(1) user defined
System internal objective	Goal interpreted in terms of adaptive engine (AE) and domain model (DM)	Concept to learn (one step at a time) (stated with triggering event for AE)	Concept to learn (one step at a time)	Project (consists of project units mapped on KI) or KI to learn for guidance tour to reach a certain goal	LO is mapped to a certain content group that has to be learned (decision on LO can be done runtime (based on learner and environment information)	Represented as a set of concepts to be learned
Properties Domain independent	User common static parameters	Yes	Yes plus authored attributes	Yes	Yes	Yes
Properties Domain independent	Experience and/or Background	n/a (not stated explicitly, but can be considered and expressed in UM)	n/a (not stated explicitly)	n/a	n/a	n/a
Properties Domain independent	Preferences (font types, pictures, examples, size, etc.)	n/a	Link coloring (default or defined)	n/a	n/a	n/a

Table 2.4: UM properties: domain dependent

		AHAM	AHA!	KBS Hyperbook	ApeLS	Interbook
Domain Dependent properties	Cognitive / learning style	n/a	Can be authored (not offered as a default option)	n/a	Supported via narratives (each narrative supports different pedagogical approach dealing with the same course meeting the same LO)	n/a
	Explicit user environment settings (time, place, etc.)	n/a	n/a	n/a	(for example, device dependent narratives mentioned)	n/a
	Knowledge	Represented by a set of concepts and a number of attributes for each content entity (attribute–value pairs) representing user knowledge of each concept (knowledge, interest, etc.)	Represented by an array of concept and a number of attributes for each entity	Knowledge vector KV is an array of knowledge items [K1, K2, . . . Kn], each is weighted according to user confidence in this knowledge	Competencies learned describes users prior-knowledge described with the same vocabulary (concepts) as narrative (DM)	Knowledge attribute value estimating users' knowledge on each concept
	Learning objectives	n/a (tracked by AE)	n/a (tracked by AE)	n/a	Competencies required describe user learning goals (minimum knowledge learner should acquire to complete a course)	
	Problem-solving task (short-term user goal)	Yes (next page guidance; local guidance)	Yes (next page guidance; local guidance)	Direct guidance	Yes (course authoring dependent)	

eLearning approach and orientation of KBS Hyperbook application), each consisting of dealing with a new concept. Thus having a diverse structure of *projects* one may fulfill different application goals having basically the same Domain and User model structures, being used all over in AHS.
Quite a similar approach has been followed in APeLS, where a *learning object* (LO) instance was able to fulfill a learning requirement, which was mapped to a certain content group, providing a choice of content depending on the users objective. At the same time LO are coupled with *narratives* to provide a domain dependent structure.
In the LAOS framework [65] a goal separation approach has been considered more elaborately, proposing the *Goals and Constrains Model* (GM). This model essentially filters useful domain concepts and groups them together, according to the goal. Because GM is a separate layer it allows the formation of goals that deal with more than one Domain model. The GM defines concept relationships that do not belong to the domain model but only define structures needed to satisfy a goal.
In the *TANGOW/COL-TANGOW* [35, 34] or *CoMoLe* [93] systems there is a set of tasks to be accomplished by users. These tasks are proposed at different times to different users according to the state of their UM and context, which makes a task dependent not only on the User Model but context of usage as well, which can not always be expressed through UM properties. We will consider context questions in the section 2.8.
As a generalization of a goal centric approach one may think of creating a hierarchy of goals and corresponding tasks comprising this goal, workflows that need to be followed to complete a requirement. Such a hierarchical structure should be aligned with a Domain Model to describe mappings between Models in order to have better adaptation results. In this case we may think of a Goal Model which might have the same structure as an overlay with the Domain and User Models correspondingly to provide concept sequences for a higher level goals representation.
As another aspect of the goal driven paradigm of adaptive systems we can consider deducing a goal from what other users have been doing within a given hyperspace. This may provide goal inference and recommendations to follow or just leading the user by previously discovered navigation patterns. In this case we may say that this goal has been learnt from other users' interaction with the system, which is opposite to the classical example where goals and corresponding task are usually assigned to or chosen by users from a known set of objectives.
So we conclude that *GAF* should be capable of versatile goal assignment, either when it was created and given by a domain expert or proposed/recommended by the system itself (e.g. modelled in [96]). We elaborate these ideas in chapter 4 where the sub-architecture of the Goal model is described and also show how a number of seemingly unrelated systems and applications can share the same goal representation via *GAF* framework in conducted case-studies (chapter 6).

2.8 Where? and when?: application and context models

When talking about AH Systems in general, there is a wide range of application areas, however the major one still remains e-Learning or Educational hypermedia with a great diversity of systems. There are also on-line information systems, which cover the fields from cultural heritage, for example [117, 16, 123] to TV guides [14, 129] or social web systems [54, 109]. This diversity is becoming richer each year. One may think of providing adaptation in consumer devices or medical industry. In this section we will not cover every application area of AHS, moreover it is becoming even more difficult to capture the whole scope of constantly appearing systems and system approaches. We will rather focus on context issues that started playing an important role in AH systems. Context aware systems gain popularity, however context awareness is usually very field-dependent. Most of the time these are context-sensitive user interactions [6], providing context-based navigation or presentation support [107, 124] or context-aware adaptive process management [5]. Context awareness in some sense may replace the definition of application area or environment, allowing the system to be decoupled from a narrow field of application to a broader concept of context which may vary, providing system flexibility with different or evolving context. An Adaptive System therefore should be able to track this dynamic and evolving context which is taken into account in GAF and is captured within the Context Model (CM) (considering both user and usage contexts of adaptation).

Combining Adaptive architectures and Ubiquitous Computing results in a semantic interoperation based approach to creating context-informed adaptive applications that make maximum use of rich content as presented in [102], or context sensitivity within content-based filtering recommender systems like in [40]. This may also be a context-based recommender system which is based on different adaptation filters to recommend individual or collaborative activities to the users according to different type of user features, behaviour and usage context like in CoMoLe system [93]. Though context may not always be applied in terms of AHS, a reference model of such a system should be designed taking into account context awareness and sensitivity aspects.

As should be clear from the description above the term context applies to both the application (context) in which adaptation can be applied and to the environment (context) in which the application is used. The application-dependent adaptation decisions correspond to the question where the adaptation is done (this also conforms with the classification of AH methods and techniques from section 2.3), whereas the application-independent environment (e.g. the use of context adaptation: time, day of the week, network bandwidth, etc.) corresponds to the question when the adaptation is done. In *GAF*, because of this difference, the where and when questions thus belong in different layers thus providing the flexibility to capture different contexts of the adaptation process, help to analyze users' behavior and to some extent make the system inferences based on the information discovered. We continue the detailed discussion of *GAF* context and application models in section 3.1 where we put them in the context of the framework modeling and adaptation process description and in chapter 4 where the detailed architecture is described.

2.9 Dexter model revisited

In this section we revisit the Dexter Model in order to show a very important observation that hyperlinks are essentially represented by queries, which makes it possible to replace the navigation structure of the hypertext model by queries (and particularly considering search queries) instead of resolving navigation links which would be considered in the next chapters when discussing open corpus adaptation, web search compliance and other related issues.

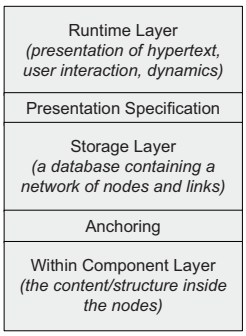


Figure 2.18: Layers of Dexter Model

In Figure 2.18 the layered structure of the Dexter Hypertext Reference model is presented. Here the storage layer emphasizes 'glueing' components and links together to form hypertext networks. These components are generic containers of data (where there is no difference between content types, graphical and textual components). On the other hand, the within-component layer of Dexter Model is concerned with the contents and structure within the components of the hypertext network.
The Hypertext system requires functions to refer to locations(items) within the content of an individual component. It is done by anchoring (e.g. to support span-to-span links). These anchors provide the aforementioned functionality while at the same time maintaining a clear separation of storage and within-component layers.
The basic addressability in the storage layer of the Dexter Model concerns the component. This component could be an atom, a link, or a composite entity which may be comprised of other components. Atomic components are primitives which are determined by the within-components layer. Atomic instances can be called 'nodes' of the hypertext system. Links here are entities which represent relations between other components. They are usually a sequence of 2 or more 'endpoint specifications' each referring to a component in the hypertext. A more detailed structure of the overall organization of the storage layer is shown in Figure 2.19, it includes specifiers, links and anchors.
Simplifying the model and considering only the Web model of linking, where only the 'TO' resolver exists (in terms of Dexter Model) we can see the complementarity of a linking and searching notions (Figure 2.20).

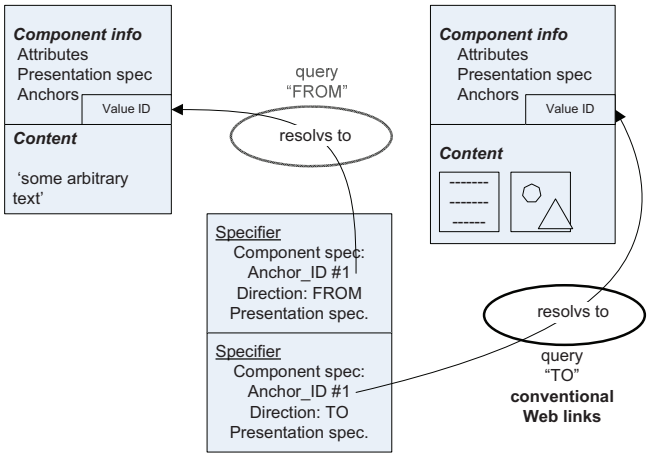


Figure 2.19: Dexter Model Storage Layer (incl. specifiers, links, anchors)

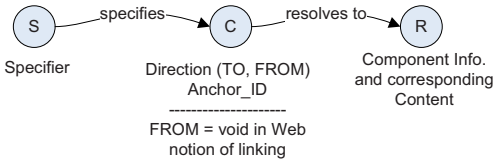


Figure 2.20: Dexter-based "Linking - Query" Model

In section 5.3 we are going to elaborate this idea and show how adaptation and which is more important the adaptation framework is compliant with the conventional web search process bridging the notions of querying and linking described earlier.

2.10 AHAM model revisited

The AHAM reference model [47] can be considered as an adaptive extension to the Dexter model. Therefore most of the layers in AHAM remain the same, while only the 'storage' layer has been modified to accommodate adaptation features. The adaptation required to split the 'storage' layer into Domain, User and Adaptation model (Figure 2.21) in order to facilitate adaptation to user attributes based on the conceptual structure of the domain, represented by the concept-link structure.
These major adaptation-enabling elements of AHAM do the following:

- DM describes how the information content of the application or is structured (using a conceptual representation of knowledge, concept hierarchies defining the trees of

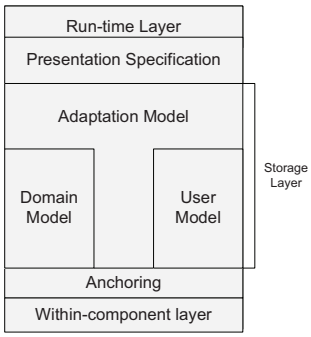


Figure 2.21: Layers of AHAM Model

concepts and types of relationships).

- UM represents user preferences, knowledge, goals, navigation and other relevant user aspects. It is devised as an overlay over DM concepts, storing user values for each visited concept/page.

- The presentation of content and link structure (from DM) is adapted to the user's behaviour as well as to the user's knowledge and interest (stored in UM). Thus an AE (implemented in the Adaptation Model) consists of adaptation rules. These rules define both the process of generating the adaptive presentation and that of updating UM.

2.11 Study conclusions: summarizing a vision of future generic AHS

Having presented a review of existing and some recent approaches to building an AH system, we summarized our vision on the future that will result in the *GAF* reference architecture in the coming chapters, highlighting key points, which will incorporate new ideas in AH research to provide greater adaptivity and flexibility of the system. Several of the items shown below are already showing up in AHS for a few years, but mostly in isolation, but now is the time when they all need to be present in a general-purpose AHS. Based on the investigation of Dexter and AHAM models we may look at *GAF* as an evolution of these two (Figure2.22).

- *Ontologies.* In many AHS authors create not only the information space but also the concept space for applications. In order to start combining the adaptation from different applications, taking advantage of what one AHS has learned about the user in another AHS, the meaning of the concepts must be agreed upon. Therefore, instead of arbitrary conceptual structures adaptive applications are becoming based on ontologies. Combining the user models and the adaptation from different applications

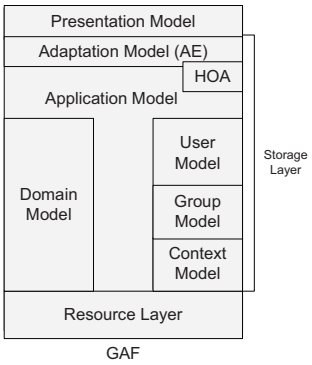


Figure 2.22: Evolving Layers of the GAF Model

based on the same ontology is a feasible problem, but when different ontologies are used, the problem of ontology alignment must be tackled first, making the reasoning on the Semantic Web [7, 11] within the boundaries of AH field more challenging. For example the AHAM reference model can handle the single ontology case (as it allows arbitrary relationships between concepts) but has no provision for dealing with multiple ontologies. In sections 3.10 and 3.7 we talk about Semantic Web and reasoning aspects correspondingly, considering mostly interoperability issues of the conventional AHS and semantic-web enabled. We also present a case-study of the *CHIP* system in section 6.4.

- *Open corpus adaptation.* Most AHS deal with a known set of information items, whether it is a single course, a 'bookshelf' or a whole encyclopedia. In such applications a concept space can be mapped onto the document space by the author. Even though open corpus is not a completely new research field, adaptive applications increasingly consider open corpus adaptation, where resources come from search results in large and dynamic LO repositories or from a Web search engine. In order to perform adaptation to an unknown document space, the mapping between concepts and documents can only be done at run-time, bringing the fields of hypermedia, databases, and information retrieval together. One of the strongest research threads in parallel to AH since the very beginning was the Open Hypermedia (OH) research aiming to separate links from documents in order to handle 'hyperstructure' separately from the media it relates to and trying to provide an alternative view of the AH from a contextually aware OH perspective [10, 9]. Recently defined in [28] open corpus adaptation in terms of, AH is receiving more and more attention providing new ideas and models in this area. Most of them introduce new approaches of adaptive navigation support in an open corpus space [25] or trying to model linked open hyperspace from open-corpus resources, providing indexing for open corpus resources (both manual or automatic keyword-based) or introduce some community-based or collaborative approaches. Currently no widely-used ap-

plications of content adaptation exist for open corpus applications (to the best of our knowledge). This is not surprising because it is difficult to foresee how the content (or presentation) should be adapted of pages that are newly found and are outside the control of the author of an adaptive application. In *GAF* we envision that content adaptation will become an integral part of open corpus applications. These approaches are covered in section 2.9 where Dexter model is revisited and linking and querying is considered as the same content retrieval approach on the modelling level.

- *Group adaptation.* With few exceptions AHS perform adaptation to individual users. However, this process can be significantly extended by taking into account actions undertaken by other users and the adaptation has been performed for other users, perhaps with a similar profile or belonging to the same (manually or automatically created) group. Determining the best partitioning of users into groups (that can be also done through collaborative tools adapted to each group features) and finally fitting this within AM is another challenge and subject of ongoing research. Although a few developments have dealt with automatic group formation (considering user features and actions) and adaptive generation of collaborative workplaces (e.g. COL-TANGOW) the main issue here for a new reference model is the existence of (group) models that are not associated with a (single) user, and rules for individual user actions generating updates of these models and of the models influencing adaptation performed for a user (belonging to the group), which is different from known stereotype AHS. In the ALS [3] project an extension to AHA! system was designed to deal with group formation and adaptation, which allows us to model users belonging to groups as well as groups consisting of users, without the need to create a new and separate way to handle groups versus users. *GAF* group modelling approach description and charts can be found in 3.8 whilst the process aspects and interoperability with other models are discussed in chapter 4.

- *Data Mining.* The behaviour of user groups may provide information that can be used to improve the navigation structure of an application. Data mining is a valuable tool in this respect. For example, clustering users into groups based on their navigational patterns can be used to automatically suggest hyperlinks or products to a user or customer, based on the common interests of the members of the group and has been considered long ago [140]. An overview of web mining for website personalization can be found in [52]. Similar research in this direction, providing hints for reorganization of sites, was described in [36]. The application of data mining in AH research has been started mostly in the area of e-Learning [116], but the need and potential benefits of data mining in the all of AHS areas are obvious. The main consequence of the introduction of data mining in adaptive applications is that the traditional AM based on ECA rules no longer covers all the possible ways in which the needed adaptation can be determined. Whereas ECA rules cover the calculation of the 'immediate' adaptation, data mining can potentially be used to capture

[3] Adaptive Learning Spaces project under the Minerva Program.

longer term effects. Ideally the outcome of data mining for adaptation would be the automatic generation or updating of the ECA rules that drive the AE. Therefore we provide an overall picture of Data Mining approaches that can be used implementing AH system or application in 3.9 besides we touch upon these problems in compliance and case-studies where the applicability of aforementioned approaches is explained.

- *Higher order adaptation.* As mentioned in 2.4 we are beginning to see applications that not only monitor the users behavior in order to perform adaptation, but also to decide to adapt the adaptation behavior. Monitoring the user and the adaptation process will allow systems to deduce either directly or indirectly (after data mining) how to refine existing rules or construct new ones. Higher adaptation orders will allow systems to do adaptation to more than one parameter at once, though considering several aspects of adaptation is inherently difficult because they may influence each other. In sections 3.9 and describing *GAF* architectural composition in chapter 4 we stress on a Higher-Order Adaptation (HOA) block and describe it within the structure of the framework.

- *Context awareness.* On the one hand shifting from Application Model to Context Awareness will help to decouple and make AH systems and applications less integrated with and dependent upon the environment in which they are used. On the other hand, considering a context model will allow the system to be sensitive and adapt in many other ways, rather than following a certain number of fixed adaptation rules. In this respect adaptation to context may also be referred to as a higher order of adaptation, providing monitored results to devise new rules in a particular context. Chapter 4 discussed not only the Context Model but distinguishes between the User and the Usage contexts of the framework. Besides a great variety of contexts is touched upon in the discussion of the conducted case-studies in chapter 6.

- *Multimedia adaptation.* Earlier in 2.3.4 we have discussed this important aspect of adaptive methods and techniques, mapped existing content adaptation techniques on multimedia content, which resulted in a level of technique abstraction, independent of a content type. Besides the part of *GAF* (described in section 2.3.5) concerns the content independence at every application level: authoring, AE, or presentation generation. It helps to generalize techniques and methods use and broaden application deployment.

As a result of these investigations, we foresee further developments and research strategies of AH as well as requirements for a modular composition of a new AHS reference model that captures all these new trends and adjoining technologies to support users within rich and diverse 'hyperspaces', bringing a new level of adaptation to the user experience.
In this chapter first and foremost we reviewed AH architectures, and defined a new taxonomy of adaptation techniques. Secondly, we showed that using the results of this analysis we have obtained requirements for a new reference model that we will discuss further and that builds on the experience gained with existing models including the Tower Model, the

AHAM reference model, the multi-layer LAOS model, and others, and that draws from the many new research ideas that show up in (prototype) adaptive systems.

Chapter 3

Modelling *GAF*

3.1 Introduction to building *GAF*

The research field of adaptive hypermedia and adaptive web-based information systems has been growing rapidly during the past ten years and this has resulted in new terms, concepts, models, prototypes and methodologies. The main existing reference model AHAM [139], developed in the beginning of this period, preceded many of these new developments. This chapter's aim is to show how these updated and new methodologies can fit into a new reference model of AHS, and provide a common reference in terms of both taxonomy, model and architecture.
In this chapter we are going to investigate different *GAF* modelling approaches, varying from process-oriented aspects to the reasoning and semantic web interoperability. The following published papers contribute to this chapter: [75, 78, 80, 64, 81].
The adaptation process is usually defined by techniques and methods of AH used in this process. Here we not only revisit the questions of the adaptation process, methods and techniques, but also address the issues of aligning all these question in a common structure, of a generic AHS and its process representation.
Adaptation techniques and methods refer to means of providing adaptation and their generalization correspondingly and comprise the core of adaptation process. Every adaptation method shows a clear idea of the approach, but at the same time each method can be implemented by a number of different techniques (for example adaptive navigation support can be implemented using guidance techniques or link hiding), likewise some techniques may be used to implement several methods using the same knowledge representation. This set of techniques and methods described in the section 2.3 comprises a toolkit of AH [22, 82]. Both techniques and methods can be applied to content, presentation and navigation adaptation. Here we distinguish adaptive presentation beyond the original content and navigation techniques and decide to differentiate the three forms of adaptation: content adaptation techniques, adaptive presentation techniques and adaptive navigation techniques in a single diagram (see figure 2.3).
But we can't really define the reference model only using the classification and description of methods and techniques. Hereafter we start building a reference model of a generic adaptation framework as a layered structure (the reason why the layered (or blocked)

structure has been chosen will be explained later in this chapter). As mentioned in section 2.4 reference models started having layered architecture with the Dexter Model, later on the Tower Model became an extensible data model for *Hyperdocuments* intended to serve as the basis for integrating hypermedia systems with other information sources (e.g. expert systems). Later the most well-known and important AHAM was developed, followed by other models and systems, such as LAOS (where the aforementioned layered approach was elaborated), APeLS, the Munich model, Hera, etc. In *GAF* we reconsider design issues for application independent generic AHS, review open questions of system extensibility introduced in the adjacent research fields, and try to come up with the generic approach to build up an adaptive framework.
The inspiration for the further *GAF* research and development is captured in Figure 3.1. It shows a set of blocs that will be used and described to devise the framework. On the left side we mention a new requirements for an up-to-date reference model, while on the right side the classification of AH methods and techniques in terms of questions that have to be answered. It shows that we are going to align the major adaptation questions and the corresponding functionality (layers or building blocks) with the new ideas outlined in the conclusions to the previous chapter (mentioned on the left side of the figure).

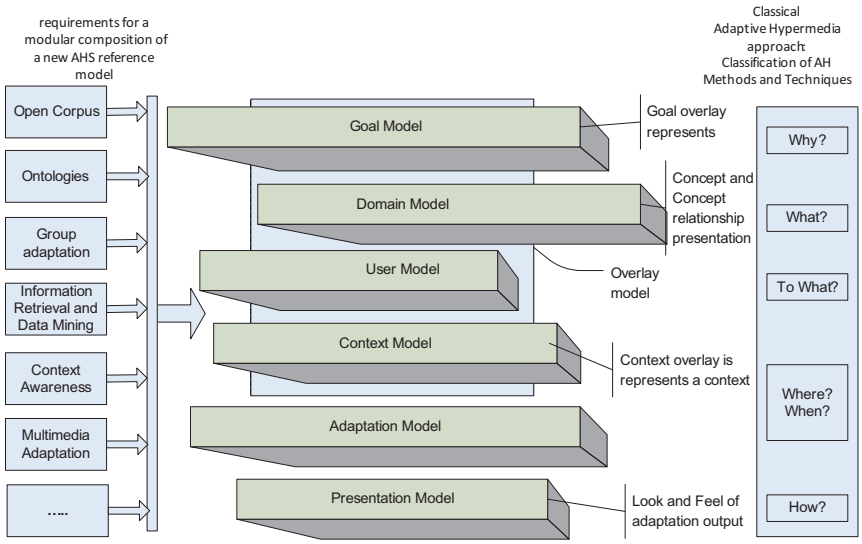


Figure 3.1: GAF Composition (a starting point for building the framework). (On the left side we list the requirements for the framework composition, where some of the requirements will be defined during the compliance studies. On the right side - major adaptation questions to be answered).

3.2 From Dexter Model, through AHAM, to GAF

In this section we show the evolution of the *Hypertext* reference models, from Hypertext to Adaptive Hypermedia to the new *GAF* which encapsulates most recent developments in AH and adjacent fields (Figure 3.2). A brief discussion of the Dexter model can be found in (section 2.9), so here we would like to concentrate on the adaptation features evolution and outline major differences of these systems.

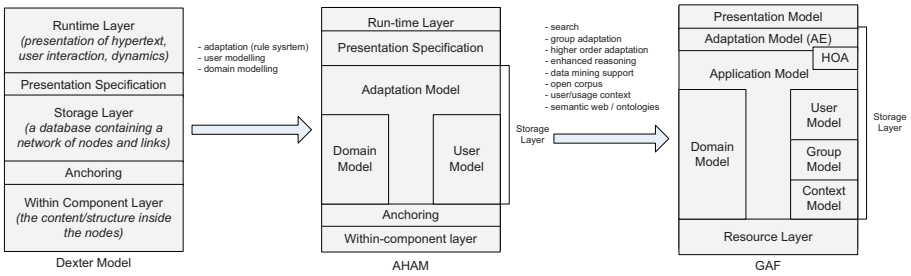


Figure 3.2: From Dexter, through AHAM, to GAF

The AHAM [47] reference model could be considered as an adaptive extension to the Dexter model. Major points of AHAM are:

- Any AHAM application must be based on *DM*, describing how the information content of the application or 'hyper-document' is structured (using a conceptual representation of knowledge).
- A *UM* must be devised and its sustainability should be maintained representing preferences, knowledge, goals, navigation and other relevant user aspects.
- The presentation of content and link structure must be adapted to the user's behaviour as well as to the user's knowledge and interest. Thus an *AE* should be defined consisting of adaptation rules. The rules define both the process of generating the adaptive presentation and that of updating UM.

In AHAM the Dexter Storage layer was split to support Domain and User modelling in order to facilitate adaptation to user attributes based on the conceptual structure of the domain, represented by the concept-link structure. The Adaptation Model (AM) encapsulates the Adaptation Engine (AE) functionality, the rule system performing adaptation based on the value of UM attributes.
Moving towards GAF, we enhance adaptation capabilities and include new methodologies and techniques, facilitating more elaborate adaptation. In Figure 3.2 (right model) we presented an extended draft architecture of GAF and briefly outline the enhancements (compared to AHAM). In Figure 3.3 we decouple GAF blocks in a way we used to describe the adaptation process and put them in a hierarchical structure to show what these major enhancements are and where they belong in the GAF block structure.

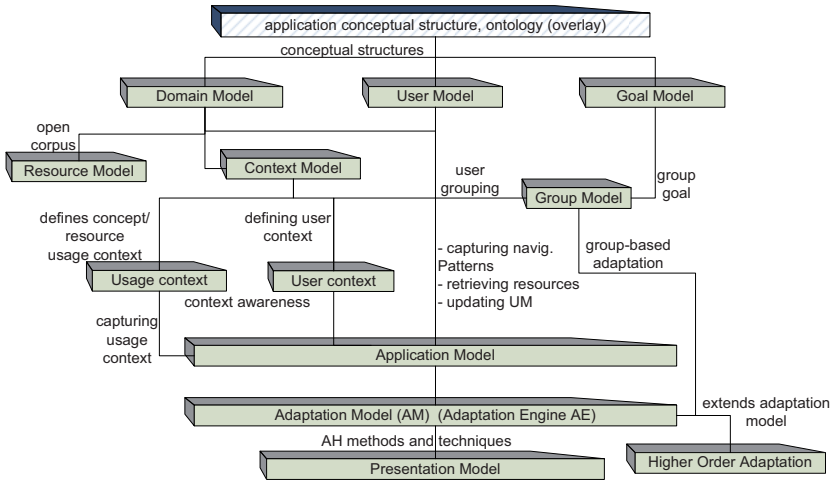


Figure 3.3: GAF Block Hierarchy

- *Ontologies* are used in order to provide interoperability in adaptive applications. These ontologies must be agreed upon, considering concept structures and meanings, therefore ontologies as a base concept structures are accepted in more and more research fields. A Domain Model based on an ontology makes interoperability feasible.

- *Open corpus adaptation* which is increasingly considered in adaptive applications is scrutinized. This is where resources come from search results in dynamic learning object repositories or from a Web search engine (see section 5.3).

- *Data Mining* is a valuable tool with respect to clustering users into groups based on their navigational patterns or capturing long term effects of adaptation rules.

- *Group-based adaptation* will extend the adaptation by taking group models into account. It determines partitioning of the users into groups and and adapting to the group model.

- *Higher order adaptation* will monitor the user's behaviour also to adapt the adaptation behaviour.

- *Multimedia adaptation* provides a content type independence at any application level, providing a generalization of adaptation techniques and methods to work with.

- *Context Awareness* allows system and application to be decoupled from the existing environment, and makes them more sensitive to adapt in many other ways rather than through a set of predefined rules. We consider usage and user context for GAF:

both capturing the context of user behaviour and Domain Model usage, allowing to adapt to user and concept contexts (e.g. environment settings).

The starting point however is the structure presented in Figure 3.4. It is based on the investigations from the previous chapter and outlines not just a basic set of blocks but basically the *GAF* skeleton. Thus from now on we will consider it as the *GAF* baseline and elaborate each and every block in the following chapter(s).

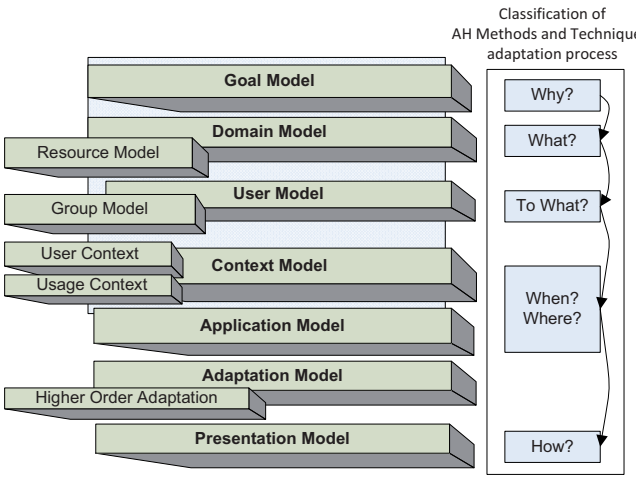


Figure 3.4: GAF Layered Model Schema (presents the overview of the framework's basic elements composition, aligned with the major questions defining AH methods and techniques described in section 2.3)

3.3 *GAF* Process Modelling

Hypermedia applications provide their users with the freedom to navigate through a large hyperspace whereas Adaptive Hypermedia (AH) offers personalized content, presentation, and navigation support. Throughout the development of the Hypermedia and later AH research field people have been trying to create *reference* models of these categories of systems. Major reference models have been favouring a layered architecture, starting with the Dexter Hypertext Model [60, 61], and later the Tower Model [46] (introduced as the Extensible Data Model for Hyperdocuments). Later this was continued in the adaptive hypermedia field with the most referenced AHAM model [47], followed by other systems and models, such as LAOS [44], APeLS [41], the Munich model [87]. However, these developments were mostly concerned with the structure and/or the data model, but not as much with the process underlying the adaptation.

Newly developed systems have brought in new terms, concepts, approaches, models, methodologies, prototypes and use-cases. But in general they have not led to a clearer picture on the "ideal" adaptation process and haven't even introduced such a notion. Although there is a larger variety of specialized systems than ever before, there is still no consensus as to what is the *ideal* generic (or general purpose) AHS would be, and in particular how its adaptation process should look like.
In this section we examine the issue of aligning the adaptation process, based on an extensive list of AH methods and techniques [82], with a layered structure of AHS. We show that to some extent the process influences and defines the composition and the ordering of such a layered structure in such a way that it partially arranges the order of the layers, defines couplings and determines the major transitions in the system. We show that the process driven approach gives more insight into AH development methods and the composition of the AH system.

3.3.1 Adaptation Process Modelling

Definition 1 The Generic Adaptation Process (GAP) *defines the interaction in AHS starting with the goal statement, exploiting features of the user (such as knowledge, interest, age, etc.) and domain models and adapting various aspects of the information and presentation to the user.*

Figure 3.5 shows this user modelling / adaptation loop as originally presented in [23]. It presents basic elements of the adaptation when the user data is collected and UM is constructed to be processed by the system (and AE in particular) to generate the so-called "adaptation effect" which can result in content, navigation or presentation changes.

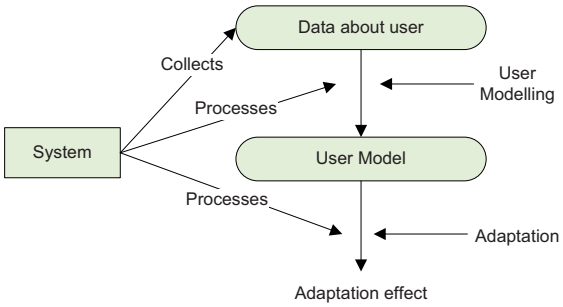


Figure 3.5: Classic Loop User Modelling - Adaptation

Considering a generic adaptive system one may think not only about defining a framework or reference (data) model but also about what the adaptation process within the system looks like, beyond what Figure 3.5 shows. Next two figures show extensions of the classical loop, taking into account that selection of user information or reasoning about the user

model to obtain answers about the user is an essential part of the adaptation process (Figure 3.6) and that either the user or an administrator (or both) need the ability to scrutinize the user model (Figure 3.7).

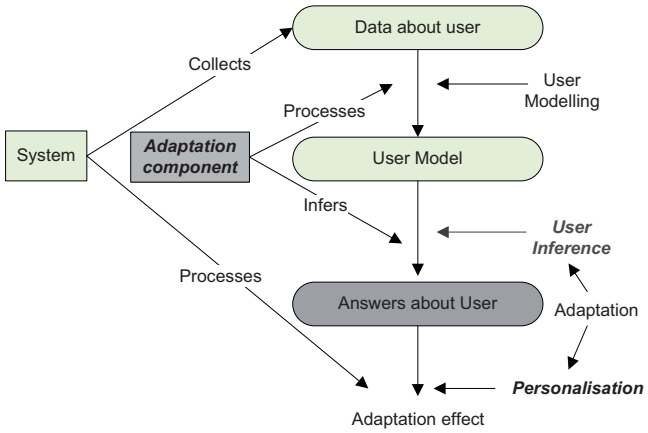


Figure 3.6: User Model Inference - Adaptation Loop

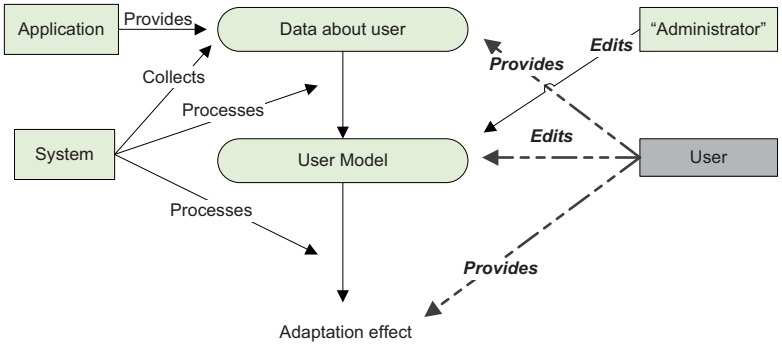


Figure 3.7: Collaborative User Model Editing (User and Administrator Involved)

These updated *user-modelling - adaptation* loops give a more extensive overview of the adaptation. In [82] we took a different approach: we extended the classification of initial AH Methods and Techniques with an adaptation process cycle to give the first insight of the *GAP* flow, shown in Figure 3.8.
Although coupling the AH methods classification with the *adaptation process* had a different purpose from what is shown in the classical (and later) loops of *user modelling - adaptation* our goal is to show the diversity of the adaptation process representation and

the possibility of aligning not only the *user-modelling - adaptation* loop into the adaptation process but the adaptation methods and techniques as well.
We consider that most of the AHS aspects can be aligned in the *adaptation process* which can serve as a reference process for AHS design, comparison and evaluation.

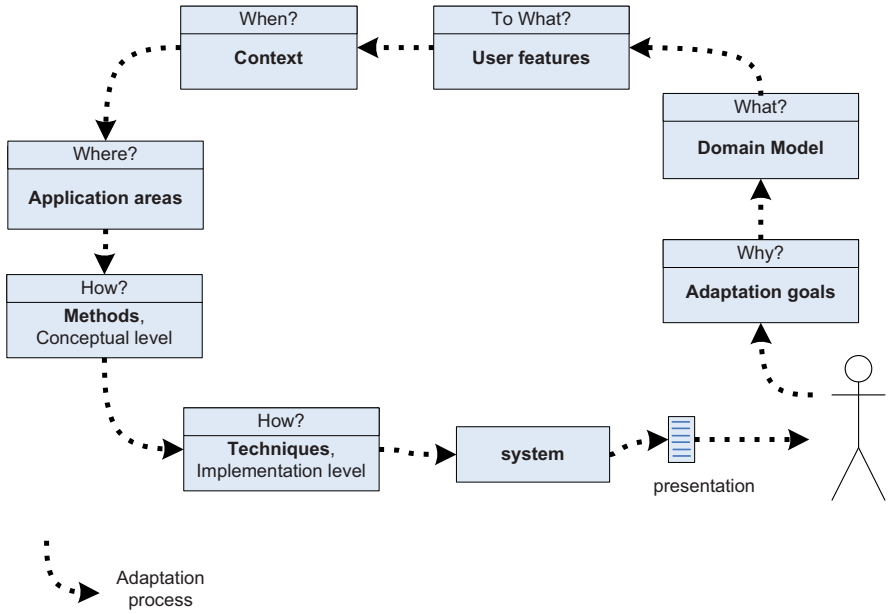


Figure 3.8: Classification of AH Methods and Techniques; Adaptation Process Highlights [82]

3.3.2 Reference Adaptation Process outlines

In this section we describe the reference adaptation process, aligning it with the traditional *adaptation questions* (section 2.2) and formalizing it in a single generic manner. In particular, we:

- *provide a flowchart diagram of a generic AHS*;
- *place the notion of the adaptation process in the context of a generic layered AHS*;
- *align the layers of AHS in a "sequence chart" and present the reference adaptation process*.

We introduce the notion of *GAP*, which consolidates most of the developments in the AHS field. Our approach starts from a methods and techniques classification, which leads to a layered-based model. We then match the adaptation process with the layered structure. We design a suitable layered structure for AHS [82] and get to the process representation by rotating these (normally horizontal) layers of the AH framework 90 degrees which eventually turns into an informal sequence chart (section 3.6). It shows inter-layer connections, information and control flows first of all in the AHS, and at the same time it easily matches the workflow representation. Thus we have a two-fold process presentation which can to some extent be derived from the AHS structure and which can also drive requirements for modular AHS composition.
Later on in section 5 we present a proof of concept of the *generic* process by showing 3 diverse use-cases (compliant with flowchart and sequence chart process representation). We show that our adaptation framework encloses both types of process representations in one go. The first example will present a classical Adaptive course [34, 21] which can be easily aligned with the generic adaptation workflow. The second example scrutinizes a different field and shows web search and the way we are used to query information on the Web being aligned with the adaptation sequence. In the third example we consider an on-line recommender compliance.
But first we consider the adaptation questions and related adaptation process insights and then *GAP* in details.

3.3.3 Questions of Adaptation and Adaptation Sequence

Adaptation can be defined by posing and answering six major questions:

- Why do we need adaptation? (*Why?*)
- What can we adapt? (*What?*)
- What can we adapt to? (*To What?*)
- When can we apply adaptation? (*When?*)
- Where can we apply adaptation? (*Where?*)
- How do we adapt? (*How?*)

This type of classification has been initially introduced in [23]. Revisiting these questions we address the issue of aligning them (also aligning the corresponding methods, techniques and respective modules (layers) of AHS) in *GAP* which can serve as a process guideline and framework for defining the way an AHS functions.
Figure 3.8 presents an order in which the adaptation questions could be asked (and answered), thus leading to the first informal definition of the adaptation process. The *classification* of AH methods and techniques is outlined by the solid lines representing the typical dimensions for the analysis of adaptive systems [118]; at the same time we join the same classification blocks considering the adaptation *process perspective* which is depicted by dotted lines. This process is usually initiated by the user stating the adaptation

goal and thus answering the *Why?* question. Then in the process we consider the *What?* and *To What?* questions, which emphasize Domain Model (DM) and UM descriptions. *When?* and *Where?* go next providing context and application area definitions. Lastly, the *How?* question describes methods and techniques at the conceptual and implementation level.

Taking into account user needs and system components (anticipating both core and optional components) we would like to present a process which explains the transitions, states, sequences and flows in a generic AHS. First we revisit a few such systems. Then, based on the research and summarization done in [82], we present the flowcharts of an AHS and finally come up with the conceptual sequence chart of a layer-structured *adaptation framework (GAF)*.

Considering the adaptation process in other systems we mention a few examples of how the authors tried to capture an idea of defining the adaptation processes (both implicitly or explicitly) in their systems and matching processes with the layered structure of their systems.

In the GOMAWE system [12] (Figure 3.9) the authors tried to fit the adaptation process in the general ontological model of the system they designed. Though there is still much to be considered in terms of the real inter-layer transitions, we can already observe a few basic transitions such as the Event Interface which either triggers the *Push Reasoning* or provides the data for the *Pull Reasoning* interfaces of the Reasoning layer. Here *Push* is responsible for transforming user events into UM updates which (events) happen when users interact with the system, and *Pull* retrieves the UM state. Moreover these connections tie different layers of the designed system together.

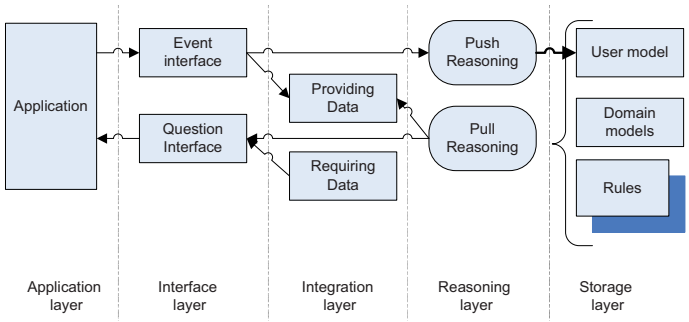


Figure 3.9: Overview of the General Ontological Model for Adaptive Web Environments *GOMAWE* (has a layered structure and presented as a sequence chart)

Another Generic Adaptivity Model was presented in [49] (Figure 3.10). It provides a cycle view of the adaptation process trying to match it with a layered structure. However, it doesn't represent any of the AHS adaptation functionality.

The Munich model defined by Koch in [87, 88] presented the lifecycle model of adaptation in the UML formalized notion (Figure 3.11).

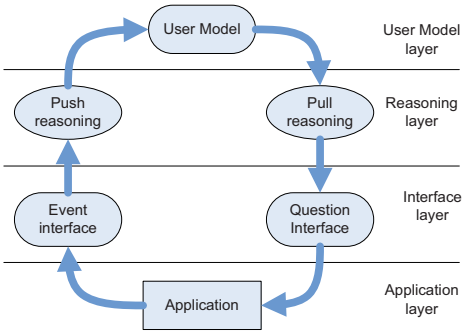


Figure 3.10: Overview of the Generic Adaptivity Model (presented as a layered structure with a loop explaining user-system interaction)

It defines the following 'layers' or components or states to be tied by these process loops: presentation, interaction, user observation, and adjustments of the systems (which include Adaptation itself and UM updates). These cycles start with an initial presentation and a default UM. Stereotypes are usually used to provide the information for the initial UM. Then the following steps of adaptation cycle follow [87]:

- **System Interaction** - which describes how to react to certain user action(s), resulting in the termination of this cycle and adaptive continuation;
- **User observation** - in which the evaluation of the information retrieved from UM is being done;
- **Adjustments** - comprising the two sub-states: User model update — in which UM attributes are updated; System adaptation - in which the adaptation is performed (adaptation of presentation, content or navigation) utilizing the state of UM.
- **Presentation** - when the system presents the adaptable elements taking into account what the information system knows about the user; the system remains in this state until the user starts interacting with system again.

To some extent most of the adaptation loops fall under this classification. The interactions are continuous and recursive when the user continues using the system and explores the knowledge base in depth. We should also mention that here we don't consider any concurrent loops that may happen and influence each other in every aspect.
The General Meta-Model for AHS, presented in [119], captures input-output processes of an AH "meta-model" (which essentially connects classic UM-adaptation loop with the navigation and presentation models) (Figure 3.12). Though these input-output parameters sometimes don't connect particular blocks in each model represented in the figure, they denote the process flow of the system, e.g. User Context is required as an input for AE to enhance the adaptation results, or the adaptation functions update UM. Thus it might be

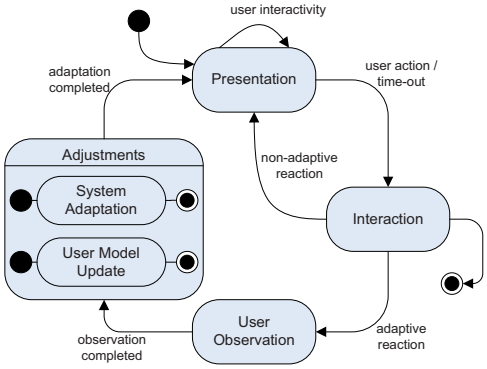


Figure 3.11: Lifecycle Model of Adaptation (Munich Model)

useful to show the data flows of the system in terms of the input/output of each component as presented in Figure 3.12.

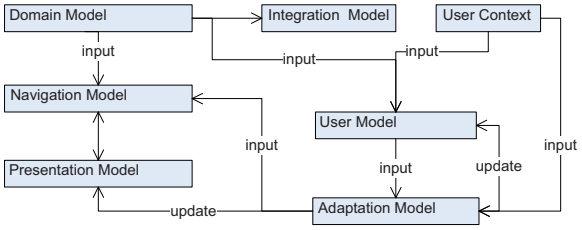


Figure 3.12: General Meta-model for AHS

3.4 Adaptation Process Flowcharts

In this section we summarize the procedural knowledge of the information flows in AHS and come up with a generic representation of AHS processes. We present the adaptation process flowcharts generalizing the functionality of the AHS. In fact these flowcharts follow the system properties summarization presented in sections 2.6 and 2.5. Based on the summarized (and generalized) functionality we devise these adaptation process flowcharts.
We distinguish the following flowcharts:

- abstract adaptive process flowchart (Figure 3.13);
- goal acquisition and adaptation (Figure 3.14);
- adaptation functionality (Figure 3.15);

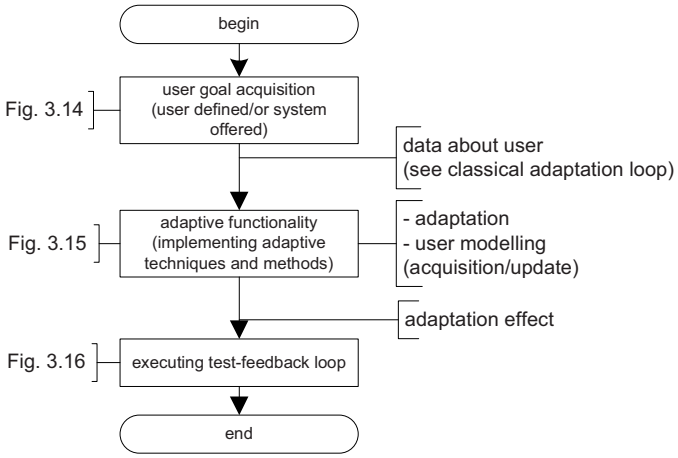


Figure 3.13: Generic Adaptation Flowchart (aggregation of figures 3.14 - 3.16)

- test-feedback functionality (Figure 3.16).

Each flowchart (except Figure 3.13) represents a certain aspect of the adaptation process, annotated to give more insight in the description of some blocks. On the right side of each chart we link parts of the process to the layers of the GAF model. The communication between the layers is illustrated in Figure 3.22. We also mark with numbers the exact correspondence of Figure 3.22 calls and transitions with the outlined blocks on the flowcharts (Figures 3.14 - 3.16) in order to show the conformity of the sequence and flowchart approaches which is discussed later in compliance study section 5.
In the *goal acquisition and adaptation flowchart (Figure 3.14)* we start with the group analysis, thus assigning the user to a group or acquiring group properties in order to take them into account while choosing the adaptation goals. Here we also make assumptions that the user may belong to one group and may not change the group within a session. The user may have his/her own goal or be advised by the system, as well as proposed to use the group goal. In any case goal suitability is checked to determine whether the user can follow it. All suitable goals are elaborated in a sequence of concepts or the most appropriate *project* (defined set of concepts to study, e.g. as it was used in KBS-Hyperbook) is chosen.
The *Adaptation functionality flowchart (Figure 3.15)* presents the main AE functionality in a sequence of concept-content adaptation steps for a particular user. In general we analyze conditions for a particular step and execute adaptation rules which apply adaptation techniques and perform presentation, content and navigation adaptation. After that UM attributes are updated accordingly and the user proceeds with the next concept either on a *one-per-click* or project-organized basis. This figure looks very similar to what was done

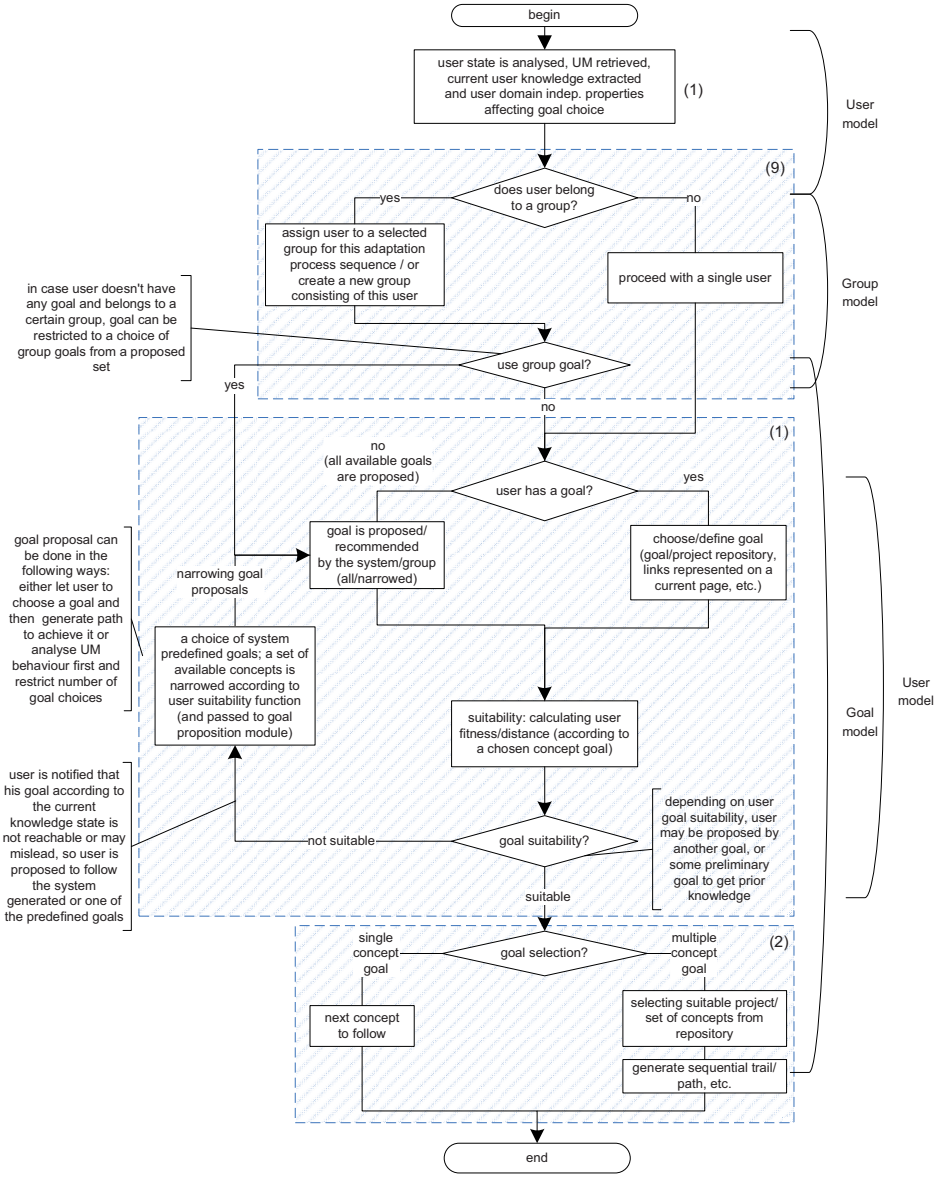


Figure 3.14: Goal Acquisition and Adaptation Flowchart

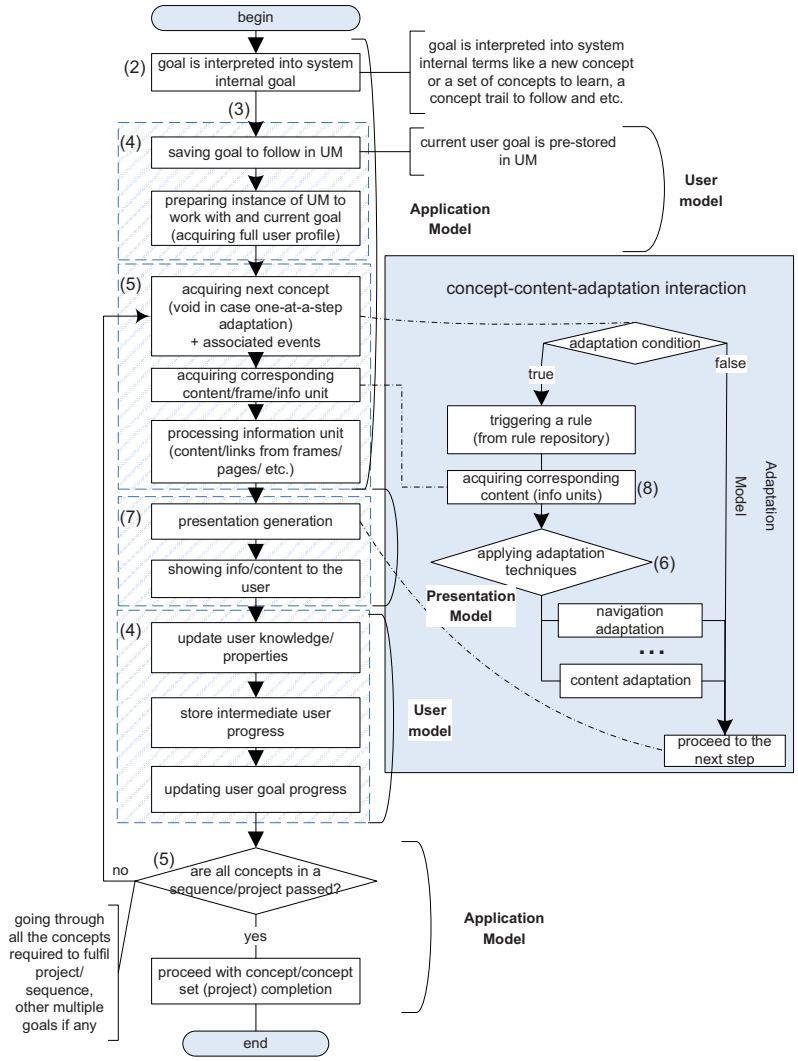


Figure 3.15: Adaptation Functionality Flowchart

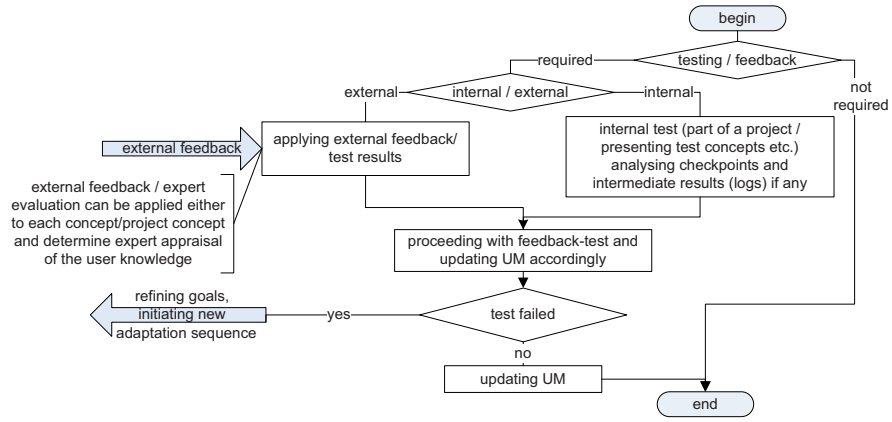


Figure 3.16: Test-feedback Functionality Flowchart

the in IMMPS model [17], presenting a reference architecture for intelligent multimedia presentation systems where the knowledge server was separated from the main flowchart in order to separate and retain the knowledge base from other system functionality. For the same reasons to separate AE functionality we have the distinguished *concept-content adaptation interaction* block. The flowchart presents the case where adaptation is done and then UM is updated. However, an AHS may also decide to first update the UM and then perform adaptation. AHA! [45, 48] and GALE [134] do it this way.
Figure 3.16 represents Test-feedback functionality. Here if such a feedback is required the user continues either with the external evaluation or internal assessment which could be part of a project or a separate questionnaire or test instance. If a user failed the test, user goals might be refined and he/she could be requested start all over again.

3.5 AHS functionality explained in terms of process representation

In order to explain the process representation we would like to show a number of previously investigated systems from chapter 2 considering their adaptation process perspective. We show the adaptation process of these systems as it has been decomposed using *GAP*. We have studied AHAM (Figure 3.17), (AHA!) (Figure 3.18), APeLS (Figure 3.19) and KBS Hyperbook (Figure 3.20) Adaptation Engine processes in depth. Numbers in each figure show the sequence of the adaptation steps and present only one adaptation iteration of the engine. It is easy to follow after considering the Figure 3.13 and its detailing.

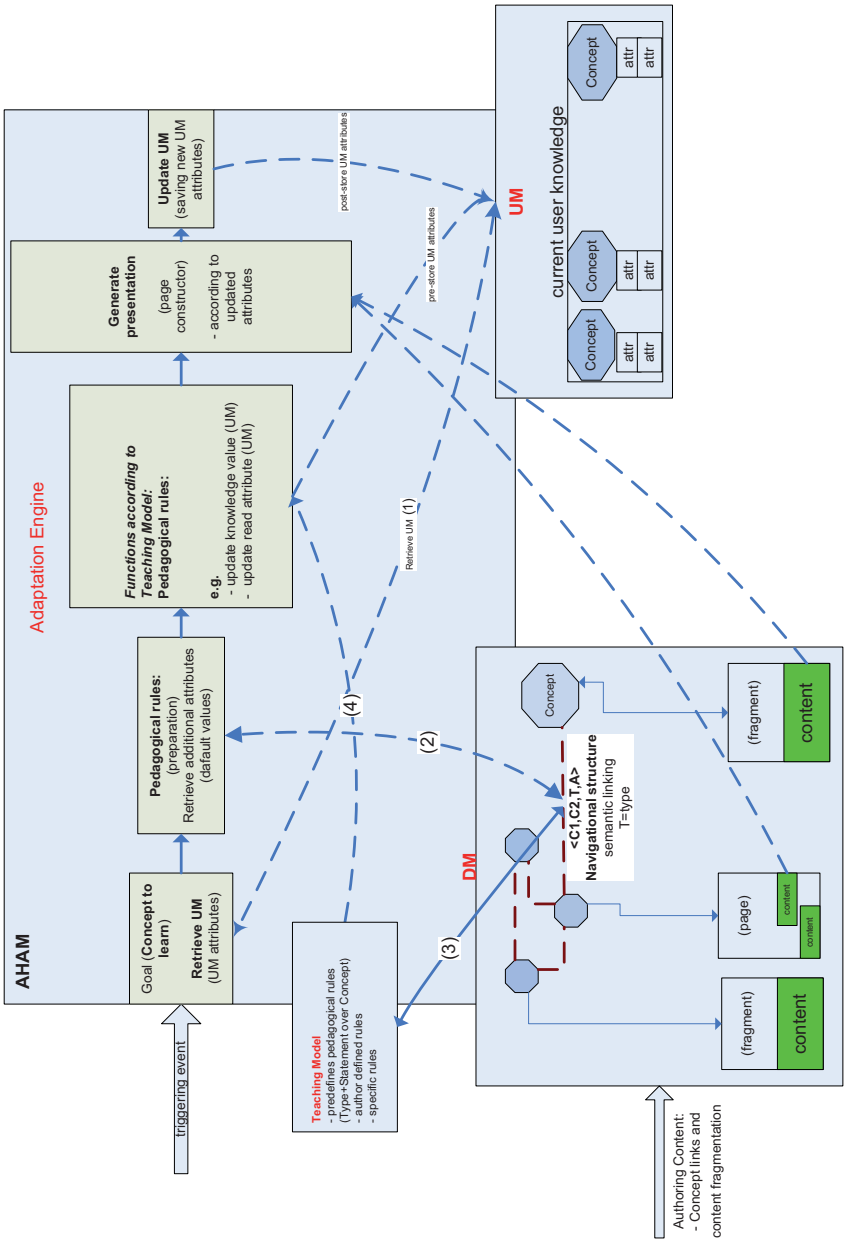


Figure 3.17: AHAM Adaptation Engine Process Representation

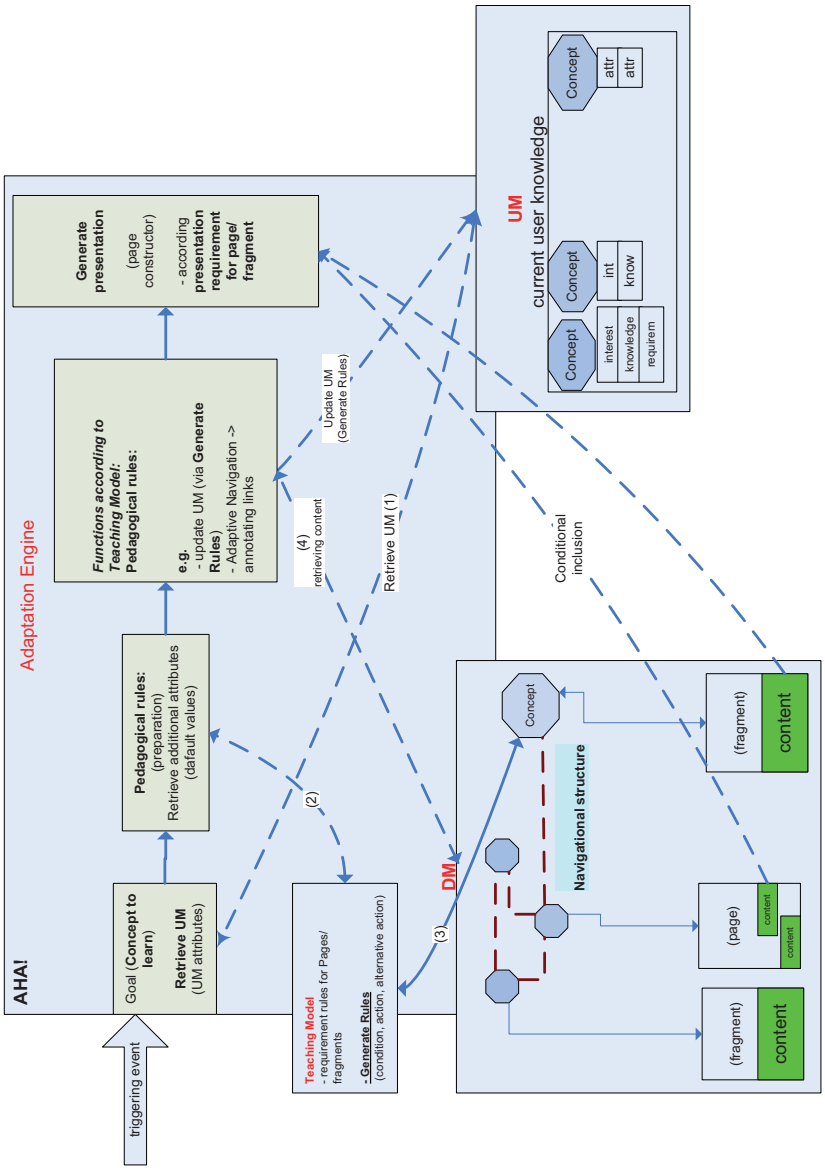


Figure 3.18: AHA! Adaptation Engine Process Representation

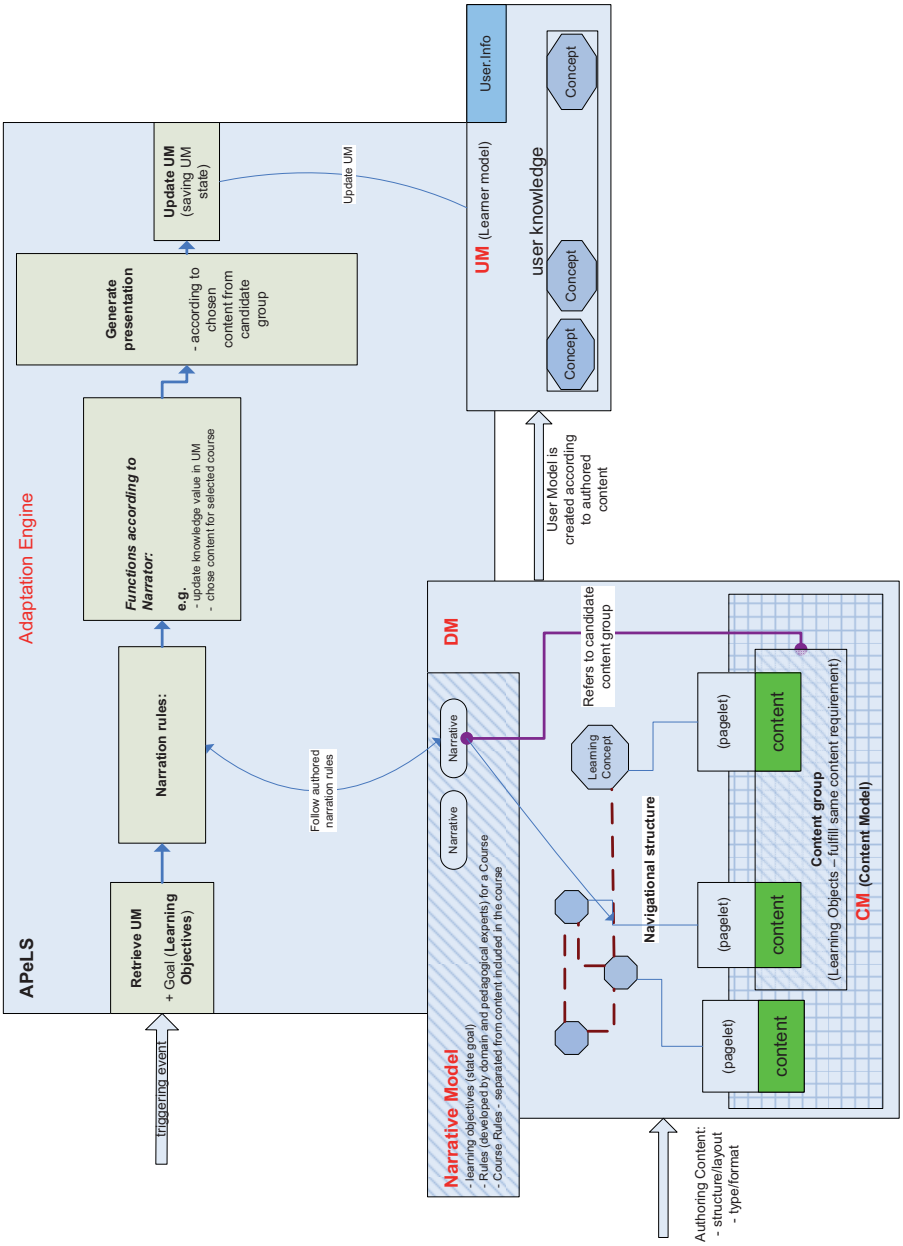


Figure 3.19: APeLS Adaptation Engine Process Representation

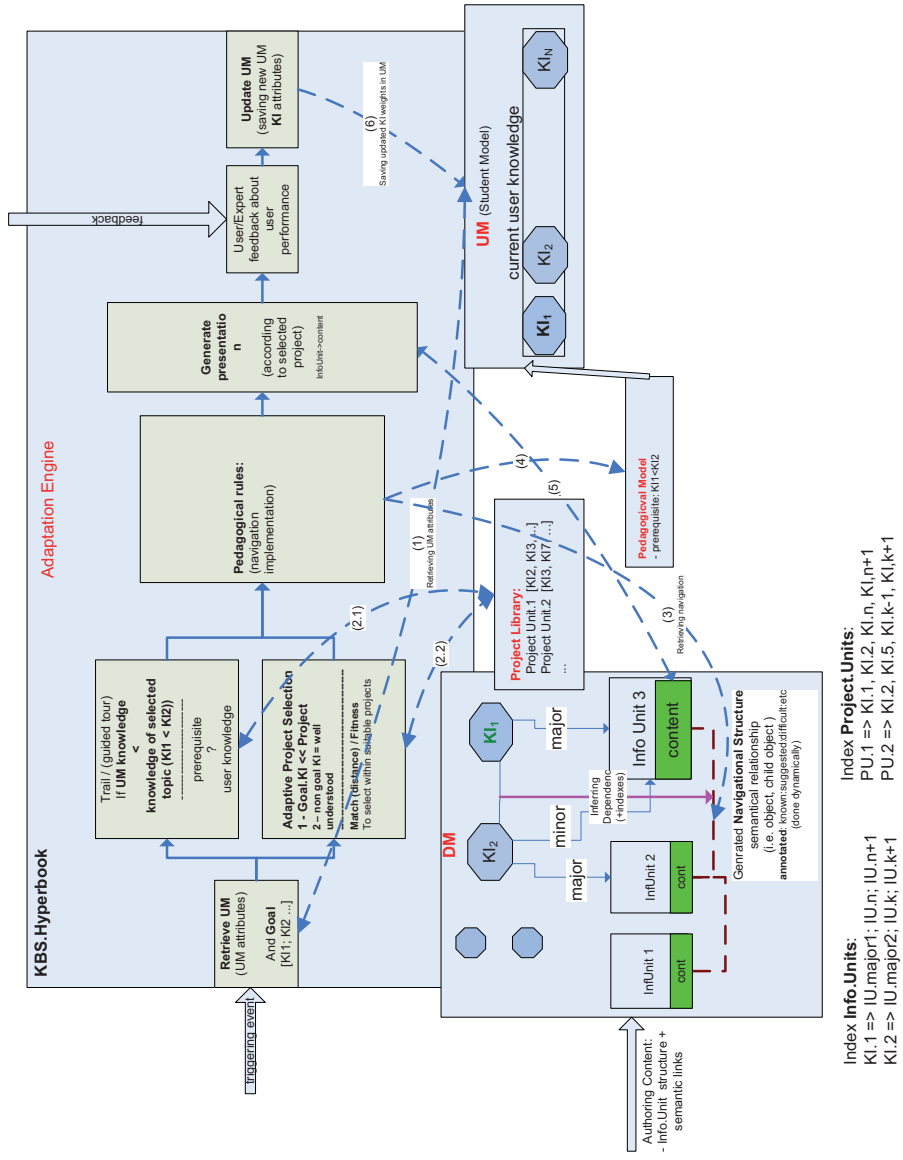


Figure 3.20: KBS Hyperbook Adaptation Engine Process Representation

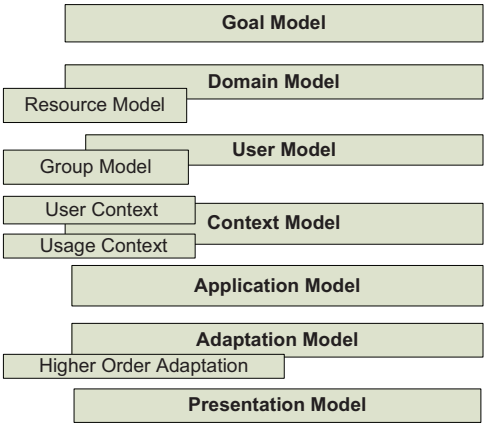


Figure 3.21: GAF Conceptual Schema (set of framework building blocks)

3.6 'Rotating' the Layers of AHS: Adaptation Process and the Layered Model

Figure 3.21 presents the conceptual structure of GAF [76] in which we align the order of the layers in the system according to the classification of AH methods and techniques (Figure 2.2). Though this order represents the basic understanding of the adaptation questions every particular system may vary or even omit some of these, thus leading to a different composition of the system layers determined by the different adaptation process. Now considering the generalized adaptation process flow-charts presented in (Figures 3.13 - 3.16) and the layered nature of AHS [82] we present GAP within the layered GAF model. We believe that in order to couple, align, sort and arrange the layers of such a system (both the generic model or some particular domain focused implementation) one should keep in mind an adaptation process sequence that will partially determine the layer arrangement and to some extent will define the mandatory and optional elements and drive the system design[1]. Figure 3.22 shows such an abstraction of GAP in terms of the system layers. (It appears rotated back as it only fits on the page in landscape orientation.)
We have marked the communication arrows with numbers to set up a correspondence with the flowcharts, where respective blocks are outlined and marked with the same numbers. This is done to show the coherence of the sequence and flowcharts. We should also note that not every connection in the adaptation process sequence exists in the above-mentioned flowcharts due to the more extensive description of the GAF process sequence chart.

[1]Thus we decided to rotate the anticipated layered structure representation (which normally has horizontal layers) by 90 degrees counter-clockwise and match it with the adaptation process flowchart.

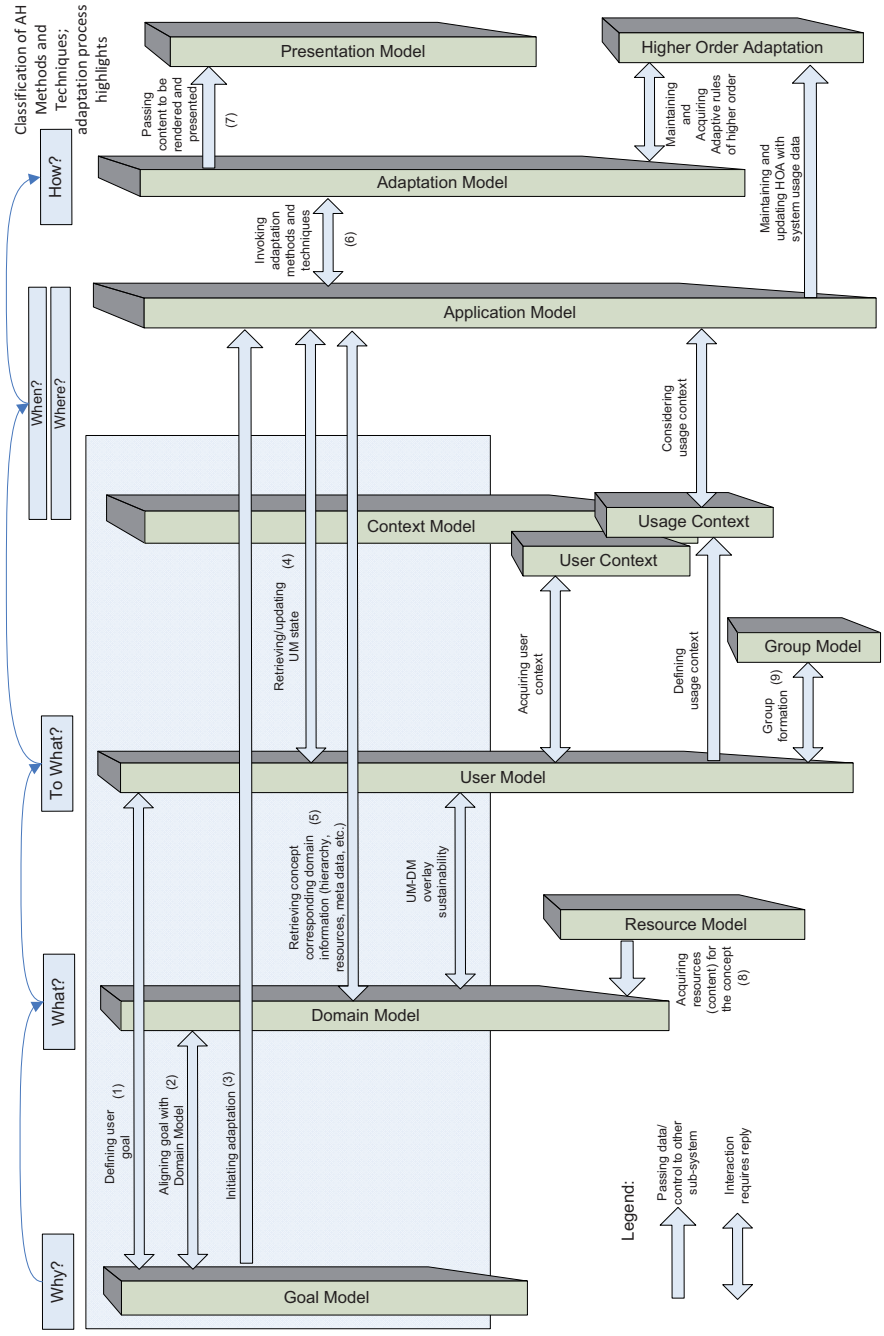


Figure 3.22: Conceptual Generic Adaptation Process Sequence Chart

The marked connections here are:

- (1) User goals are defined. In case user doesn't define any goal it can be proposed by the system or a group goal can be used;
- (2) User goals are aligned with DM, considering the conceptual structure of the domain. According to the selected goal a suitable set of concepts to follow is chosen;
- (3) Adaptation is initiated and control is passed to the Application Model (AM);
- (4) Operations of UM properties such as acquisition (which is performed before actual adaptation) and update (which may be done before as well as after the actual adaptation execution depending on the system implementation) (corresponds to a few places on the flowchart);
- (5) Operations mainly concerned with working with the concepts from DM;
- (6) Appropriate adaptation methods and techniques are invoked;
- (7) Retrieved content is passed to the Presentation model to be rendered/generated and presented to the user;
- (8) Corresponding content (for concerned concepts) is retrieved from the Resource model and handed over back to AM;
- (9) Group related operations (assigning users, retrieving group properties, defining new groups, etc.).

3.7 Reasoning in GAF

In order to perform adaptation based on domain and user knowledge an "author" should specify how the system interaction results in information presentation (based on the information retrieved from DM). For instance in AHAM, this is done by means of an AM which consists of adaptation rules. AE runs and interprets these rules, handles link anchoring and generates the presentation specification. It is based on ECA rules which describe the UM updates and the "adaptation processes" to some extent.
In section 2.6 we have thoroughly investigated the questions of the conventional AH reasoning approach and compared a number of state-of-the-art systems from the reasoning perspective. These were the AE in the AHA! systems using the aforementioned ECA rules, as well as KBS Hyperbook, which also introduces the deduction rules (based upon the conceptual modeling 'Telos' language). The APeLS system used JESS rules representing facts that make certain rules applicable and assert the rules (complies with the ECA reasoning type). But ECA rules are low level and sometimes they are difficult to understand, therefore, AHS used to provide authoring tools offering higher-level instruments to author the adaptation (e.g. in AHAM using "concept relationships" and "concept relationship types"). There were a number of issues associated with ECA reasoning. In [139]

those concerning problems of termination and confluence were considered, and a static analysis of rules and a simple strategy for dynamic enforcement has been proposed.
In the following sections we revise the reasoning approaches used AHS and adjacent fields and introduce a generic AH ECA-type reasoning framework in section 3.7.1. Moreover, we briefly consider a number of new to AH field approaches: Association Rules usage in the AH context (section 3.7.3) and Case-Based Reasoning applicability in AHS (section 3.7.4).

3.7.1 Combining Technologies for GAF-ECA Reasoning Framework

In order to provide the flexibility, interoperability and generic manner of AH reasoning within GAF we introduce a semi-formal reasoning framework of the ECA nature to facilitate AH system functionality interchange and provide the reasoning basics in AH field. As discussed in *MARS* [101] the best way to facilitate reasoning interoperability (we would follow) the way of discriminating different sub-parts of the ECA rule (here distinguishing *Query* and *Test* phases in the *Condition* part of the ECA alike rule) and consider them first as a separate instances combining into a single GAF reasoning framework. In fact we would like to generalize and show that AH reasoning complies with the ECA type and can be represented in a single generic framework combining different techniques to implement each of the instance.
In Figure 3.23 we present ECA rules classification based on the investigation done in the previous chapters. It aggregates a different types of reasoning approaches varying from those used in a conventional AH systems to a more elaborate Semantic Web-enabled systems and custom developments.

3.7.2 Restricting the EQTA Transitions in GAF

It is always difficult to predict the final result in a rule-based system, as the execution of the rules may result in infinite iterations or in an unpredictable end point which results in problems of termination and confluence correspondingly. These problems have been covered in [139]. However in order to restrict meaningful transition between these four states (E-Q-T-A) (condition is considered as a query and a test) we propose a state diagram describing all the possible transitions and states in *GAF* ECA framework in Figure 3.24. Each state presented may be recursive. We also discriminate reactive and active states, however would like to restrict the reactivity of some of the states in order to avoid infinite loops and consequent termination problems (e.g. invoking action from the action).
Unconditional actions as well as unconditional tests can be triggered by the event without querying for any particular value, these result in testing some predefined and/or statically stored values which don't require any querying. On the other hand and in case of generalization we consider that whether the query is required or not it is executed anyway as a void call. Queries can also cause a new event (e.g. querying a value might trigger an event to update an access attribute) but can't cause any new actions without a test procedure. In a simple case this test call can be a binary check assigned to the action and checking the necessity of a designated action. Test can query a new variable values using the query

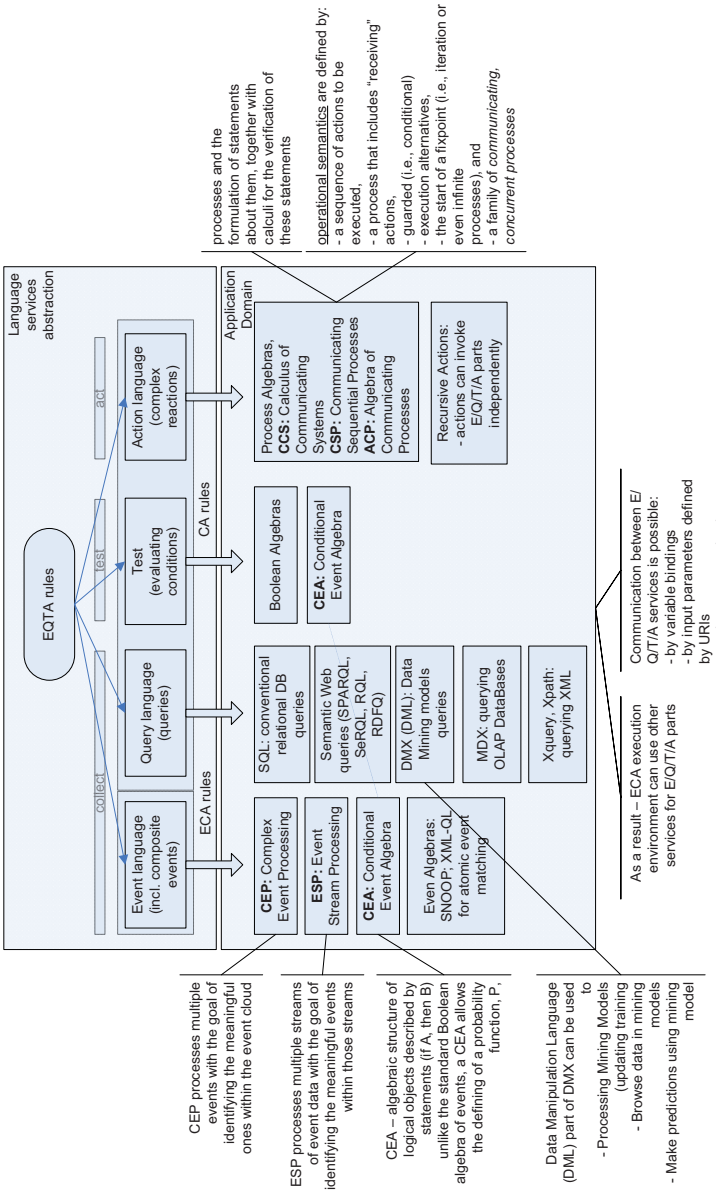


Figure 3.23: ECA Rule Breakdown Classification
(presents ECA (EQTA) rule breakdown and lists associated methodologies that can be used to replace any E-Q-T-A part in order to compose a complex rule structure; explanations of the blocks can be found in the left and the right side of the figure)

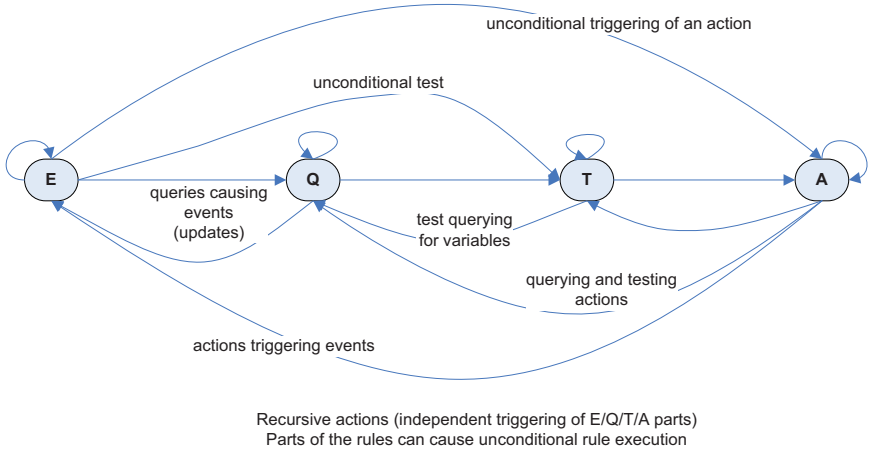


Figure 3.24: GAF E-Q-T-A Rules State Diagram (transition graph)

"sub-system" or just calculate the value to be tested (e.g. as in backward reasoning). A test can be recursive or composite (require a calculation of more than one test variable) It can also trigger more than one action at a time or composite action (e.g. update UM variables and render the presentation). Action in general can be nested and trigger other events (e.g. as in a sequence of concepts to follow adaptation).

3.7.3 Association Rules in GAF

Association Rules can be used to relate entities (concepts, pages, links, etc.) that are referenced together thus association mining facilitates the AE rule refinement process. It can be used to relate pages and concepts that are most often visited together by a user or by a particular user group. It may also help the needs of the domain experts or users. To some extent it may help driving the adaptation process (especially that one of a higher order) which can be done automatically if the confidence is enough or suggest some solutions to the domain expert that can be used for the system refinements and adjustments. Association rules amongst others comprise the *GAF* reasoning toolset.
In the web usage mining context association rules refer to sets of pages that are accessed with a support value exceeding a certain threshold, moreover the pages may not be connected (with the direct hyperlinks) (e.g. *apriori algorithm* discovers a correlation between users who visited a page containing a certain product and users who access a different page with a different product). Alongside marketing purposes, these rules can be helpful for web experts, including (AH) researchers to restructure DM (including conceptual and navigation structures). Additionally the rules may help with a heuristics for pre-fetching documents which helps to deliver user personalized content in AHS.

Association rule learners (such as Apriori methods) have a number of metrics which could be re-interpreted in AH systems in a following way:
Confidence (Accuracy) - rules have an associated confidence, which is the conditional probability that the consequent will occur given the occurrence of the antecedent. This metrics can be used by the Bayesian Network reasoners, since it can manage uncertainty in knowledge observations and their conclusions. For example to make conclusions and reason over them, the estimation of a conditional probability of the student knowledge being on a certain level of expertise is made.
Support - indicates how frequently the items in the rule occur together. This may help to cluster rules with the similar antecedents and consequences together and facilitate new rule inference in such a way to make this rules more stronger in order to increase the system stability or follow the supported trend. Both Support and Confidence identify the rule validity, however there are exceptions showing that even when both metrics satisfy the validity condition the rule can be still useless. (e.g. students who follow the courses A also follow the course B with a 75% confidence. The combination of these two courses A and B has a support of 30%).
Lift - is used for rule quality evaluation, and shows the rule strength over the random co-occurrence of the rule antecedents and consequences. Discovered rules with the importance less than 1 dont show the opportunity of knowledge gaining and have to be proceeded in a different way or discarded. In a formal way lift can be expressed as follows:

$$Lift = (RuleSupport)/(Support(Antecedent) * Support(Consequent))$$

3.7.4 GAF meets case-based reasoning

The goal of this section is twofold: to generalize the idea of combining Data Mining (and in particular pattern discovery) and case-based reasoning (CBR) in the domain of Adaptive Systems, and to show the possibility of other than conventional ECA reasoning in order to enhance AE with case-based capabilities.
Attempts to combine Knowledge Discovery (KD) with the CBR were done in numerous fields, including medicine [55] but as a result the hybridization of these systems was very focused in the domain without providing an overall picture on how it can be done in a generic type of a system. At the same time there were some systems aiming to combine CBR with other adjacent fields (COMPOSER [110], CADRE [69]). Figure 3.25 on the one hand presents the possibility of using the notion of CBR in AHS, and on the other hand utilizes the usage patterns to create cases for the adaptation. It shows the way the 'adaptation' cases are created (and the types of cases that can be utilized by AHS), processed and handled by the AE.
The main idea of this schema is to show the possibility of these techniques (IR, DM and CBR) to be combined to serve the needs of adaptive hypermedia. Moreover it contributes to the *GAF* reasoning framework.

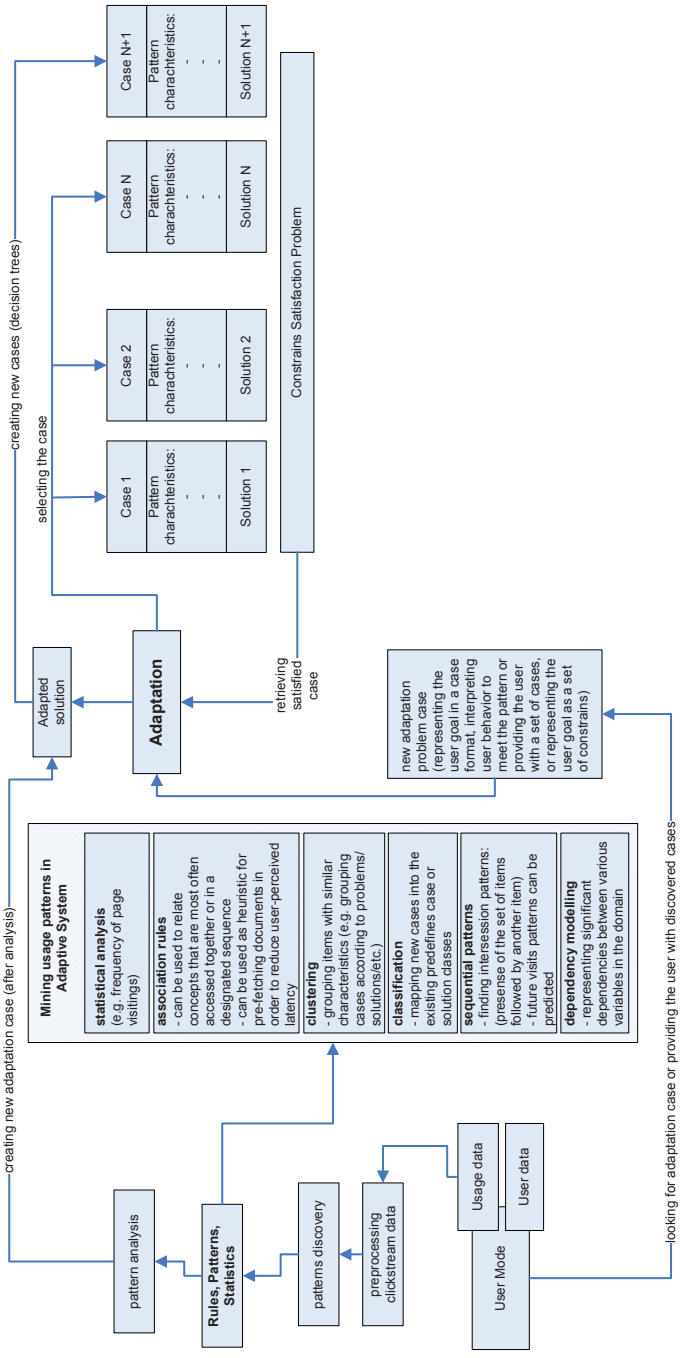


Figure 3.25: GAF meets case-based reasoning

3.8 GAF groups modelling insights

Most AHS perform adaptation to individual users, however we can extend this by taking group adaptation into account. Determining partitioning of the users into groups and applying this in the context of application model is a challenge. Based on the investigation in chapter 2 we present a generic group modelling sequence. Schema in Figure 3.26 shows the users' division into groups based on the concerned property (e.g. their interest, task, relation to other users, etc.). In order to show how the users are assigned to a different groups we present Figure 3.27. It covers all major group formation scenarios according to [13] and is meant to elaborate block number *(9)* from Figure 3.14 (section 3.4).

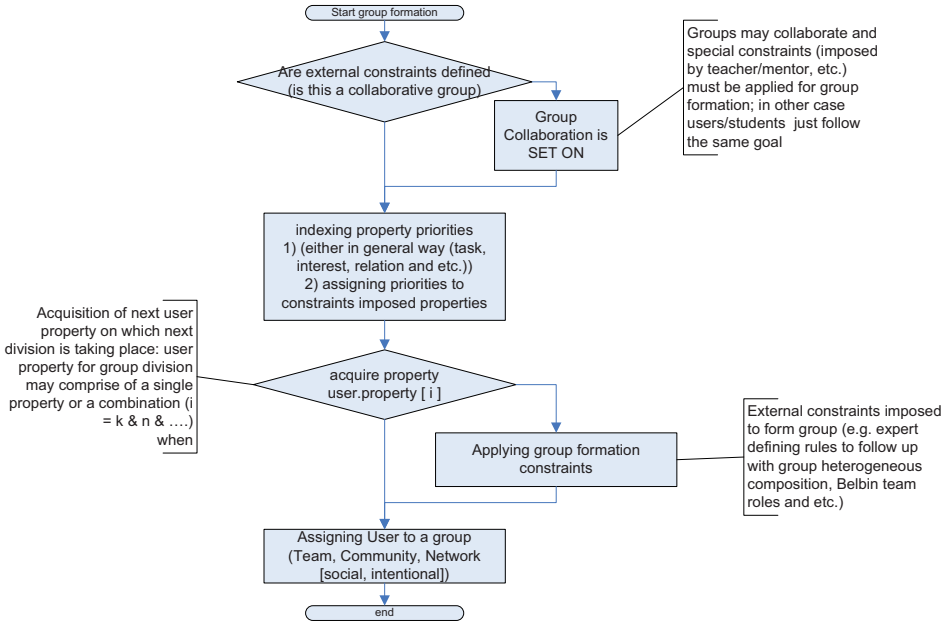


Figure 3.26: GAF Group Modelling Sequence

3.9 Towards data-driven toolset in GAF

In this section we summarize "data-driven" ideas we had so far (mostly data mining and information retrieval approaches applicable in the AH context) and present a "data-oriented" toolset in Figure 3.28. We don't discuss this figure in details here (as it mostly concerns data mining and information retrieval techniques in general) but rather use this toolset to discuss and explain *GAF* compliance and case studies in chapters 5 and 6 accordingly.

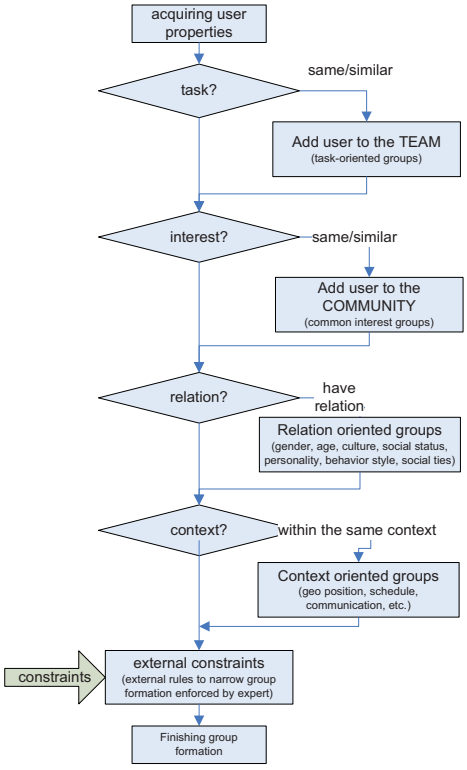


Figure 3.27: GAF Group Modelling Scenarios

3.10 Semantic Web insights

In this section we give some insights regarding conventional AHS and semantic web enabled systems side by side, investigating complementary properties of the systems. In the tables 3.1 - 3.6 we envision and explain interoperability issues in the *GAF* context.

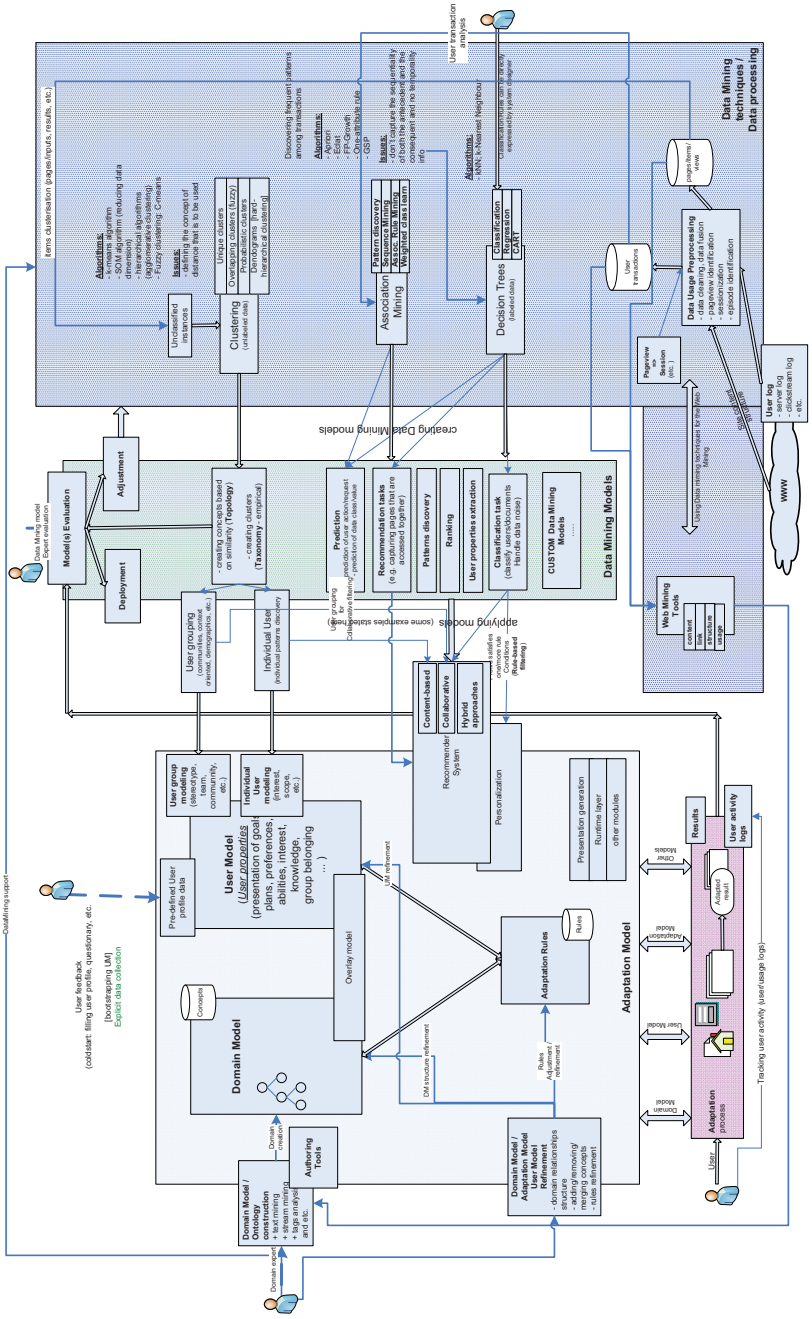


Figure 3.28: GAF "Data-Oriented" Toolset

Table 3.1: AHS—SW GM Properties

Property	conventional AHS	Semantic Web	Interoperability / comments
Goal Model	A set of concepts, presented in a form of sequence (or one at a time), representing a knowledge (other) base that needs to be handled in order to reach a designated result (gain knowledge in (complete course), etc.)	Adaptation Goal ontology (shared with DM ontology) or query builder specifying semantic search query (goal ontology should be aligned with UM to provide feasibe goal selection)	n/a
User Goal / objective	Overall (learning) goal stated by interaction with user (stretched over the whole interaction session of the user with the system)	Can be a short or a long term goals	n/a
User goal statement	User chooses link (available on the current page), project to study, etc.	Following a link(s) (and essentially linked concepts), choosing an available concept to interact with, getting a suggestion from the system (related concepts) or posting a query	Mapping on the conceptual level of DM
System internal objectives	Single concept / set of concepts (project/sequence/index of concepts, etc.)	Translated into SW query language (querying concepts and relationships)	Ontology (concept) alignment to

Table 3.2: AHS—SW DM Properties

Property	conventional AHS	Semantic Web	Interoperability / comments
Description	representation of a set of concepts within a domain and the relationships between those concepts	formal, explicit specification of a shared conceptualization (ontology provides a shared vocabulary, which can be used to model a domain)	shared representation in SW
Concepts and hierarchy/relationships	is a tuple (SP, SC, SCR) SP is the set of pages (content), SC is the set of concepts and SCR is the set of concept relationships (of a type: link; prerequisite; inhibit; part (compositional), etc.)	Individuals: instances or objects Classes: sets, collections, concepts, types of objects Attributes: aspects, properties, features, characteristics, or parameters that objects (and classes) Relations: ways in which classes and individuals can be related to one another Function terms: complex structures formed from certain relations that can be used in place of an individual term	Interoperability by means of formal ontology description or alignment in SW (OWL, RDF/S, etc.) (usually manual or semi-automatic)
DM construction	Construction is usually individual for an application domain, mostly using DB or Active DB to support basic reasoning capabilities or more generic solution like XML. DMs are usually pre-authored by domain experts and are rarely re-usable by different parties.	Most ontologies describe individuals (instances), classes (concepts), attributes, and relations. Ontologies are constructed using ontology languages (usually declarative and usually generalization of frame languages (frame (that can be fulfilled) is applied to the structuring of language properties))	essentially any given DM (in conventional AHS) is a very simple ontology (directed tree)
Technologies used	query language used within DB (by DBMS) which stored the domain information	Traditional ontology languages: CycL, F-Logic, LOOM, OCML, etc.; Markup ontology languages: DAML+OIL, OIL, OWL, RDF/S, etc.	re-using technologies from the SW side, (converting ontologies, extracting, re-using), usually done on a syntax level (e.g. XML)

Table 3.3: AHS—SW UM Properties

Property	conventional AHS	Semantic Web	Interoperability / comments
Description	usually represented as an overlay of the Domain Model structure, where attribute values for that concept are stored	expressed in terms of ontology (which can be shared with DM); GUMO; GenOUM; (user model dimensions are represented with different model classes: Physiological, Characteristics, Emotional, Mental, Motion, Role, nutrition, etc) - should be opened and extensible model	n/a
Domain independent (experience, preferences, learning style, grouping, environment)	User properties can be represented by UM attribute values learning style can be authored using additional ECA rules	User properties are modeled via UM ontology, representing users' classes and properties (attributes); Grouping can be done by sharing ontologies between users; ontology of user environment cab be also shared/overlapped with UM ontology.	UM classes and properties interoperability can be done via AHS-SW systems ontology alignment, re-using a sharing a common ontology
Domain dependent (knowledge, interest, results, objectives)	Knowledge and interest are represented as a vector of concept-value pairs, showing the level of knowledge reached in each concept Results are mapped on the same domain knowledge vector of concept knowledge-value pairs Objectives are use usually a set of concepts to be learned (often - a sequence of concepts) stated by the user (chosen from a list), or suggested to the user	— ” —	— ” —

Table 3.4: AHS—SW UM Properties (cont.)

Property	conventional AHS	Semantic Web	Interoperability / comments
User Observations	clickstream can be transformed in usage patterns	user observations in a form of statements (including triggering events, queries, sequences of actions, timestamps, subjects, and other arbitrary actions)	usage patterns can be derived (mined) in both cases
Other UM properties (short/long term properties)	usually a custom properties defined by the system authors/experts	expiry-classification (can change within days/month/years/not at all/etc.)	custom, depends on the types of the system, each property should be aligned separately
Distributed UM	custom build UM, usually refers to storing different parts of the model separately or storing the same UM in multiple locations and synchronizing it (e.g. client-server models, UM caches)	user model distributed approach reflect features taken from several standards for a user modelling	n/a

Table 3.5: AHS—SW AE Properties

Property	conventional AHS	Semantic Web	Interoperability / comments
AE functionality	offers page construction facilities; performs adaptation by executing rules (ECA); constructs pages according retrieved values and condition; accordingly updates UM;	SW query engine exploits semantics of DM (uses semantics to get more relevant results); includes similar classes (e.g. subClassOf, EquivalentClass, etc.), searches equivalent constructs; processes language variations; contains a proof layer (to retrace the derivation of the answers and its explanation);	on the level of rules and rules composition (E-C-A parts of the rule structure) (explained in section 3.7.1);
AE Output	updated UM attributes; links adjustment; page construction; next page is shown to the user; goals adjusted; groups refined;	concept and associated relations with explanation on a search result (how this results were deduced) - may be in a form of a concept map/graph with related explanations	AE output can be propagated from the Rule execution layer;

Rules Interoperability	interoperable only of authored using the same semantics and language (e.g. CLIPS, JESS, LISP, etc.)	SWRL (Semantic Web Rule Language combining OWL and RuleML), SiLRI, DQL, RQL, N3, Squish	RuleML, SWRL allows interoperability with major rules systems such as CLIPS, JESS
			`http://www.ruleml.org/`
			`http://2006.ruleml.org/group3.html#3`
			Rules structure interoperability (based on E-C-A type of rules) explained in section 3.7.1

Reasoning Issues	include termination and confluence	include assumptions of descriptive logic limiting SW reasoning capabilities	n/a

Table 3.6: AHS—Semantic Web Adaptation Methods and Techniques

Property	conventional AHS	Semantic Web	Interoperability / comments
Modelling	set of techniques, described in respect to a different content type and input parameters	Adaptation Methods and Techniques ontology (a set of techniques and methods relationships (incl. types describing inheritance, n-ary relationships, inclusions, etc.)	(Adaptive Methods and Techniques Ontology) ontology will allow to introduce interoperability of techniques and methods in application to different system concepts and representation types (e.g. AH techniques taxonomy)
Content and Presentation	conditional text stretchtext fragment variants page variants frame-based techniques	the same techniques can be used in terms of content rendering techniques	types-vs-methods ontologies - ontologies matching techniques and methods in respect to content types and structures (which techniques suit best which content type and structure, which methods should be used for structured and unstructured data (detailed in section 2.3.5))
Navigation	direct guidance (next best, page sequence, trails) link sorting (similarity sorting, prerequisite knowledge) link hiding link annotation (traffic light metaphor, traffic lights and hiding) link disabling link removal map adaptation link destination	semantic web relationships	interpreting semantic relationships and applying navigation techniques, in the most simple case the links are propagated from the

Chapter 4

GAF Reference “Architecture”

In this chapter we present the *GAF* reference “architecture”. It is hard to define as a conventional architecture (e.g. software architecture), since we capture a number of different aspects in one go: data models, data and control flows, scenarios and even some technical aspects that are tightly connected with a particular feature(s). To avoid the confusion we will refer to this schematics as the GAF ‘architecture’. To come up with this representation we elaborate the GAF reference model and the generic adaptation process model (see Figure 4.1 top and bottom figures respectively) and describe each component separately (sometimes event a part of it due to the scaling reasons) depicting its (model’s) internal structure, data representation, technologies and interfaces. One may consider this schematics to be an overlay of different models and architectures. For each decomposed part we summarize its structural description and the functionality in the tables. Therefore each section in this chapter consists of a schema and a table.
In these Figures one can find the major structural blocks of the models and their connections. Colored blocks mean that they are involved in the similar use-case/scenario (e.g. using overlay model across a number models) or a compliance case (these compliance cases are described in chapter 5). Block may be connected by a number of incoming and outgoing connectors. Colored connectors mean that they perform a similar functionality (e.g. context related operations or information handling by AE). Major connections are annotated and have a respective interface descriptor (e.g. *AE_Retrieve(Dm/conc.)*) and is discussed in the table. Tables consider the description of the corresponding functionality, structure (of a block or a sub-block), relationships within these structures, advantages (essentially outlining the advantages of the model in the *GAF* context) and major interfaces. We didn’t include all of the connections and sub-structures in the discussion on purpose, since most have been previously considered in the *GAF* modelling chapter and/or self-explainable.
Figure 4.2 shows an overall picture in order to give the idea about the scale of the schema.

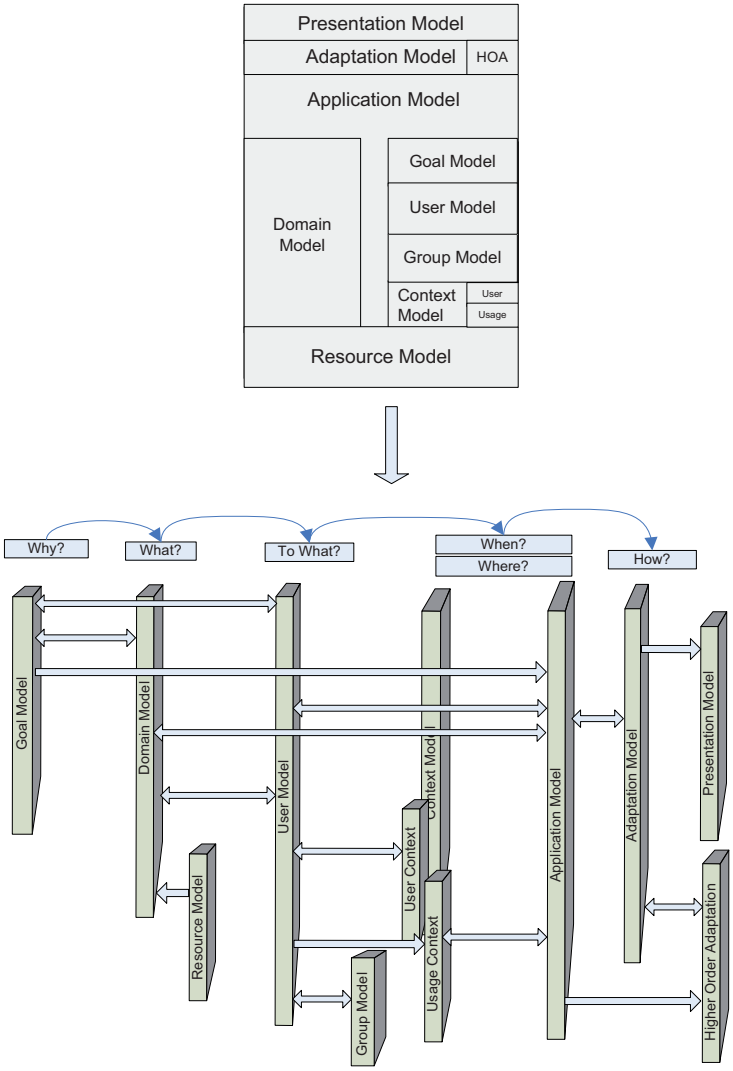


Figure 4.1: *GAF* Architecture Evolution: top - GAF model; bottom - GAP 'sequence chart'

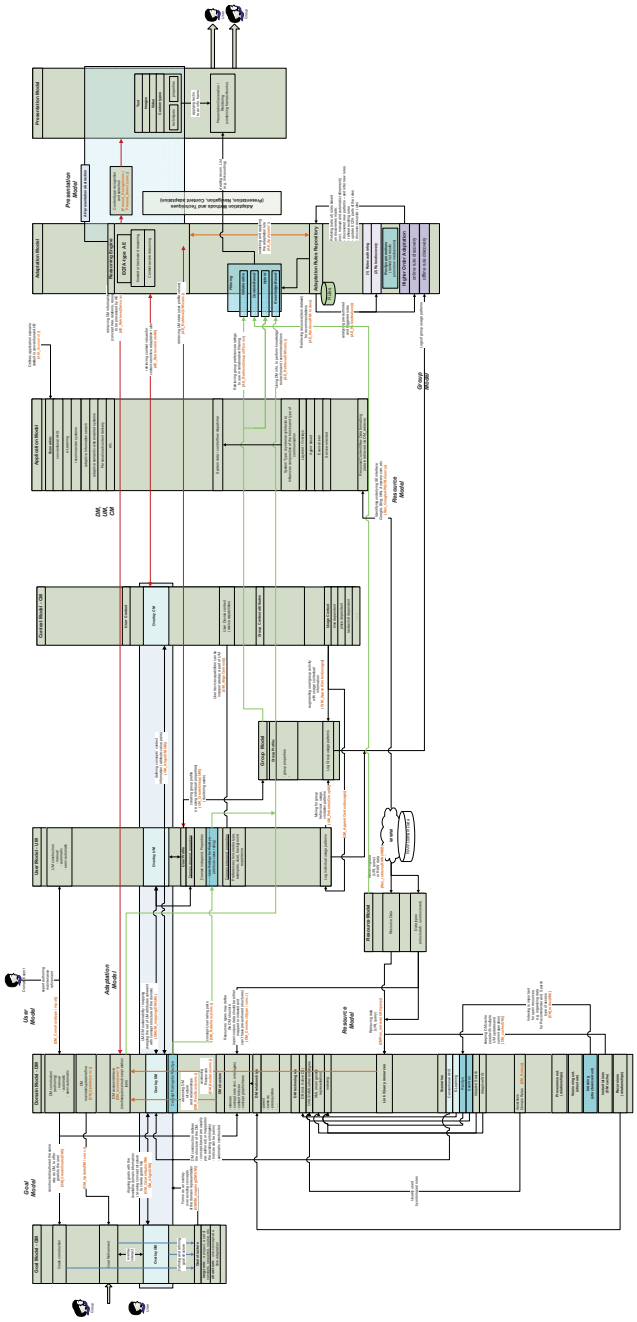


Figure 4.2: *GAF* detalization overview (this picture is presented here in order to show the scale of the GAF schema)

4.1 Goal Model (GM)

In this section we start discussing the *GAF* architecture with the Goal Model (GM), its reference architecture, interaction with the other models, major and minor components, functionality and the important interfaces of this model. In Figure 4.3 we show GM separately (with the respective connections and necessary explanations) and discuss its functionality and the details in Table 4.1.

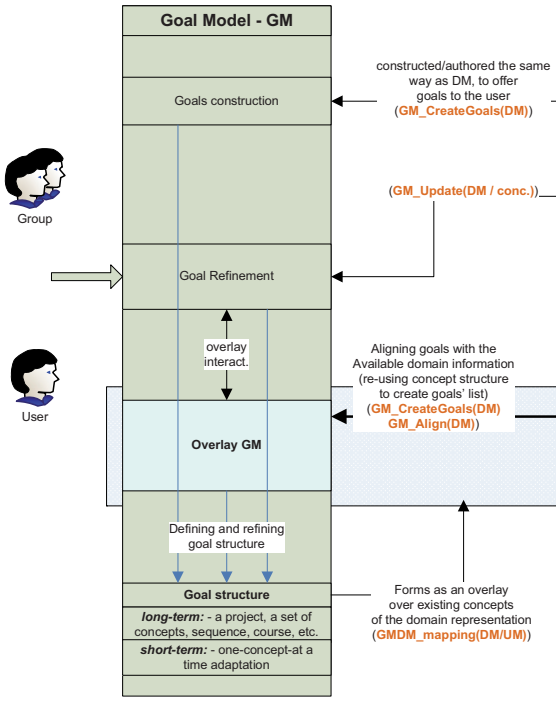


Figure 4.3: *GAF* Goal Model architecture. Explains basic internal reference structure and major interactions (such as creation, alignment and update) with DM and the corresponding dependencies (e.g. in terms of interfaces' parameters)

Table 4.1: GAF Goal Model

sub-component (question)	functionality	structure	relationship	advantages	interfaces / details
n/a	n/a	hierarchical structure of goals, objectives, tasks, requirements, etc.; index or a set of DM concepts to follow; overlay over DM structure;	usually the same relationships used in DM (mostly representing the sequencing in complex goal structures)	- re-use of the Goals; - deal with more than one DM; - provides group goals to individual users (helps on a cold start)	*GM_CreateGoals(DM)* creates goal(s) based on the conceptual structure present in DM *GM_Align(DM)* aligns GM and DM conceptual structure (map goal concepts on the existing DM concepts) *GM_Update(DM/conc.)* updates goal model based on the changed DM or a particular concept
individual goals	describes the individual goal of the user in terms of domain concepts	long-term goals: - a project, a set of concepts, sequence, course, etc. short-term goals: - one-concept-at a time adaptation	defined by the system	each user has his/her own goal which is derived from the system, offered or recommended to the user	*GMDM_mapping(DM/UM)* maps individual user goals (based on the DM) onto UM attributes (presents the user only the reachable goals)
group goals (what are the goals of others?)	describes the group goal	the same as for individual goals with an additional group attribute	— ” —	re-use of individual goals and vice versa	group goals are usually assigned to a group; there is some additional interaction for the user-group assignment described in 4.4

4.2 Domain Model (DM)

The *GAF* reference architecture functionality is presented in Figures 4.4 and 4.5. The first figure concerns major DM interactions with UM, AM and GM, while the second shows mostly RM within model interaction. Both figures are explained in Tables 4.2 and 4.3 and respective tables discussing GM, UM, RM and AM from this chapter.

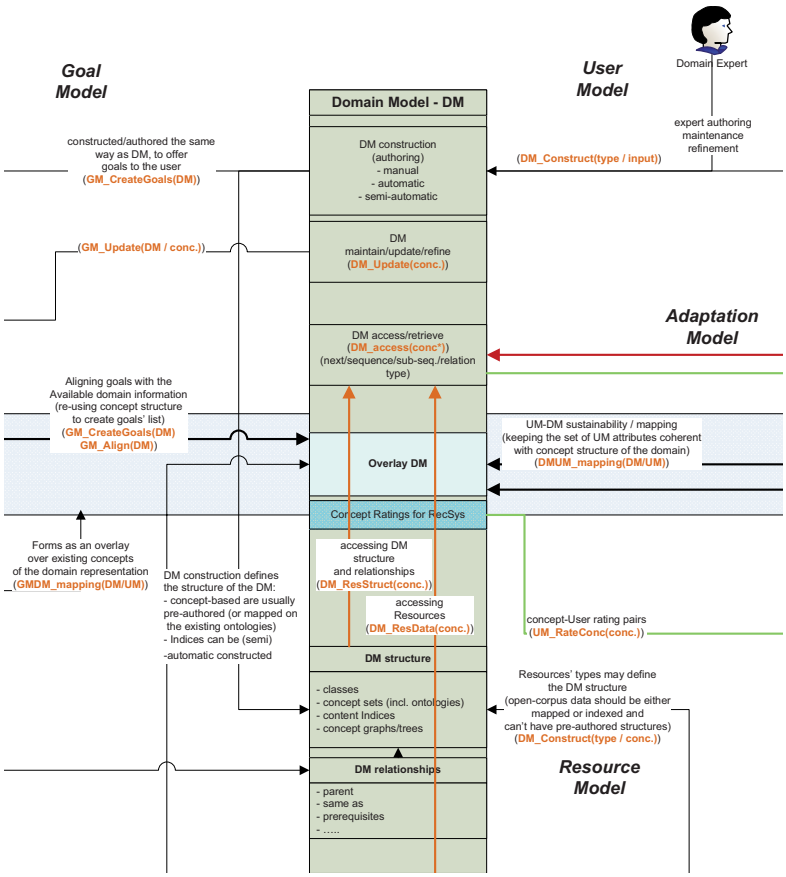


Figure 4.4: *GAF* Domain Model architecture (part.1 incl. UM, GM, AM (and some RM) interactions). There are interfaces defining model construction, major overlay functions keeping the integrity of the overlay and block defining the conceptual structure (incl. relationships) and access functions.

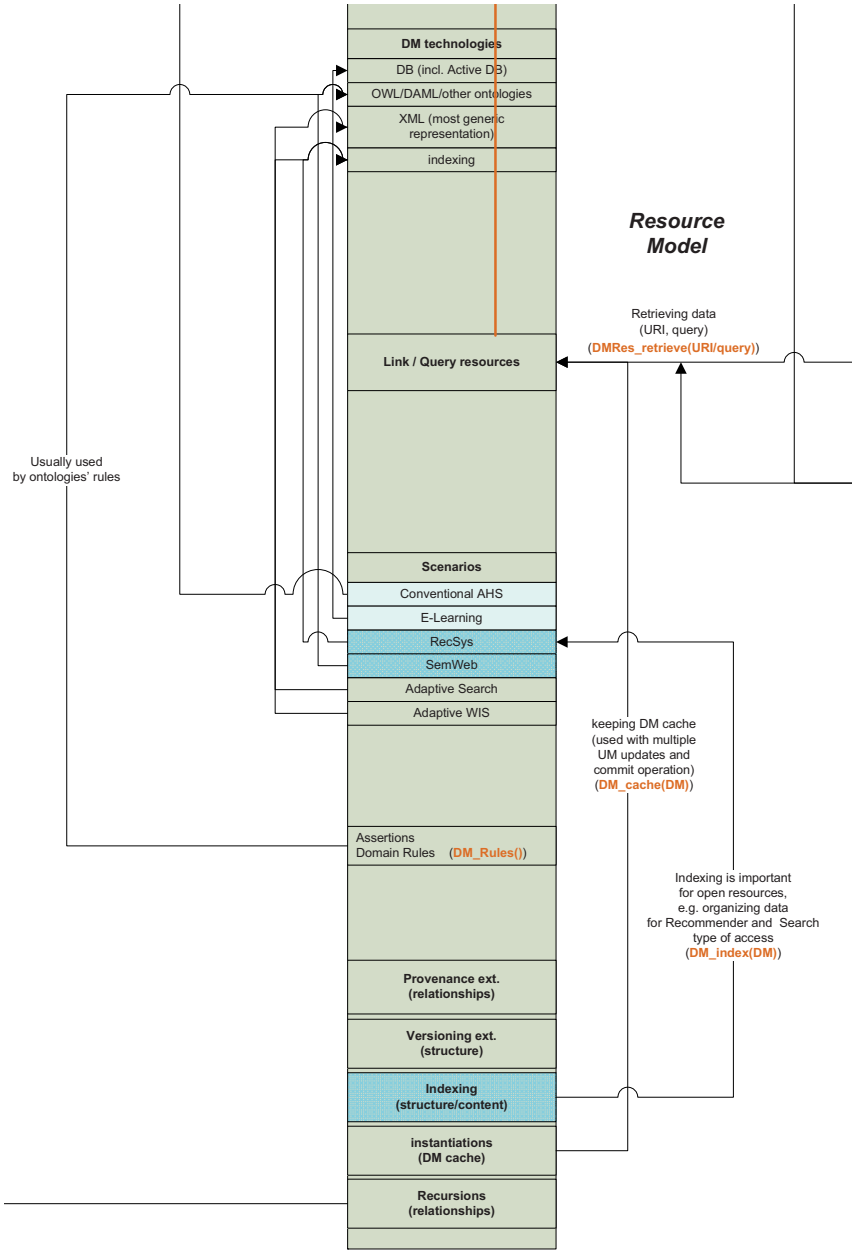


Figure 4.5: *GAF* Domain Model architecture (part.2 use-cases, RM and within model interactions). This part of DM reference architecture shows

Table 4.2: GAF Domain Model

sub-component (question)	functionality	structure	relationships	advantages	interfaces / details
structure	functionality of the system using DM defines its structure, different types of the systems require different DM representations (concept hierarchies, ontologies, indices, etc.)	each structure has an associated technology (implementation)	n/a	a variety of structures and approaches gives the flexibility to the system and allows (in case of a proper interfaces implementation and structures alignment) to deal with multiple sources of information (interoperability)	*DM_Construct(type/input (conc.)* - DM authoring (construction) by a domain expert or determined by RM used which may define the *type* of the structure (e.g. with open corpus data or RecSys DM where it is presented as an index) and the *input*; *Input* here defines the type of date from which DM is constructed: raw open corpus data, structured labelled RecSys data, conventional SW ontology that has to aligned or a hierarchy of concepts (as in AH); RM may also contribute to the automatic or semi-automatic DM authoring when the expert has to map the existing concepts on the ones retrieved; *DM_Update(conc.)* - performs internal DM updates (usually during the refinement), and propagates the changes onto UM and GM in case of the overlay and dependencies;
concepts	knowledge representation of a concerned domain	structure may be defined by the ontological attributes, arbitrary properties usually used in conventional AH systems, or for instance feature space (labels) in RecSys case	n/a	n/a	*DMUM_mapping(DM/UM)* function performs the conceptual alignment between DM and UM, keeping the overlay sustainability and updating the changed (refined) concepts correspondingly; *DM_access(conc.)* function accesses a particular concept or a first concept in a set of concepts (e.g. in case of indexed concepts) and may use the two following functions: *DM_ResStruct(conc.)* retrieves a sub-structure of a particular concept (a set, a sub-tree or an ontological class); and then *DM_ResData(conc.)* retrieves the corresponding content associated with the concept from RM;

Table 4.3: GAF Domain Model (cont.)

sub-component (question)	functionality	structure	relationships	advantages	interfaces / details
relationships	define the relationships between concepts of the domain knowledge	usually presented in a form a concept hierarchies (directed acyclic graphs) with prerequisite and parent-child relationships; uses ontological relationships	n/a	n/a	concept access functions should not only retrieve a particular concept (an entry concept in the set or an index) but the corresponding relationships as well (retrieving child concepts (or at least identifying that it is a composite concept) or a relationship to a next concept in a set)
content	retrieved from RM section (see 4.3)	n/a	n/a	n/a	*DMRes_retrieve(URI/query)* retrieves resources from RM with a provided resource URI (conventional AHS when resource URI is stored in the concept attribute) or queries the resource (open corpus systems or an adaptive search engine); *DM_index(DM)* creates an index of the concepts and associated resources to use used by search and recommendation functionalities of the system *DM_cache(DM)* keeps the cache of DM in case it is required for intermediate updates and commits (e.g. in versioning context (see 7.2)) *DM_Rules()* triggers domain specific assertions (usually used with ontologies)

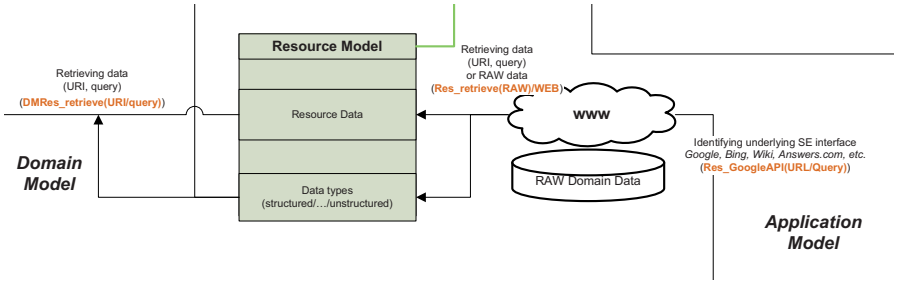


Figure 4.6: *GAF* Resource Model architecture.

4.3 Resource Model (RM)

RM presented in Figure 4.6 shows the basic structure of the model and the respective connections to DM and AM which define the resources retrieval by DM using an URI or querying information when there isn't any particular resource linked to a DM concept. A connection to AM defines an underlying external interface (which is usually a search engine) and it identifies a use-case of a particular framework "instance" (adaptive search, conventional AHS, RS, etc.). More details are provided in Table 4.4 and chapter 5 elaborates a number of usage of RM cases.

Table 4.4: GAF Resource Model

sub-component (question)	functionality	structure	relationship	advantages	interfaces / details
("What" content is adapted?)	originally intended as a storage layer (Dexter, AHAM)	defined by the type of the system, may be structured as in SW and AHS applications or unstructured as in SE thus defining data types used by the model	n/a	distinguishing RM helps to separate pure conceptual concerns of DM and the actual content which can be stored and handled separately by RM	*Res_retrieve(RAW)/WEB* retrieves the actual resources from RAW data (unstructured), from Web, from any adjoint resource repository *Res_GoogleAPI(Domain URL/Query))* identifies SE used by the system (might implement any other than Google interfaces)

4.4 User and Group Models (UM and GrM)

In Figure 4.7 we combined the presentation of UM and GrM since they are of a very close functionality. Besides, in the ideal case these two can be modeled the same way, essentially Um can be derived from a void GrM and the other way around when GrM can be considered as Um with aggregated properties of the group. Table 4.5 describes the details and interfaces of both UM and GrM.

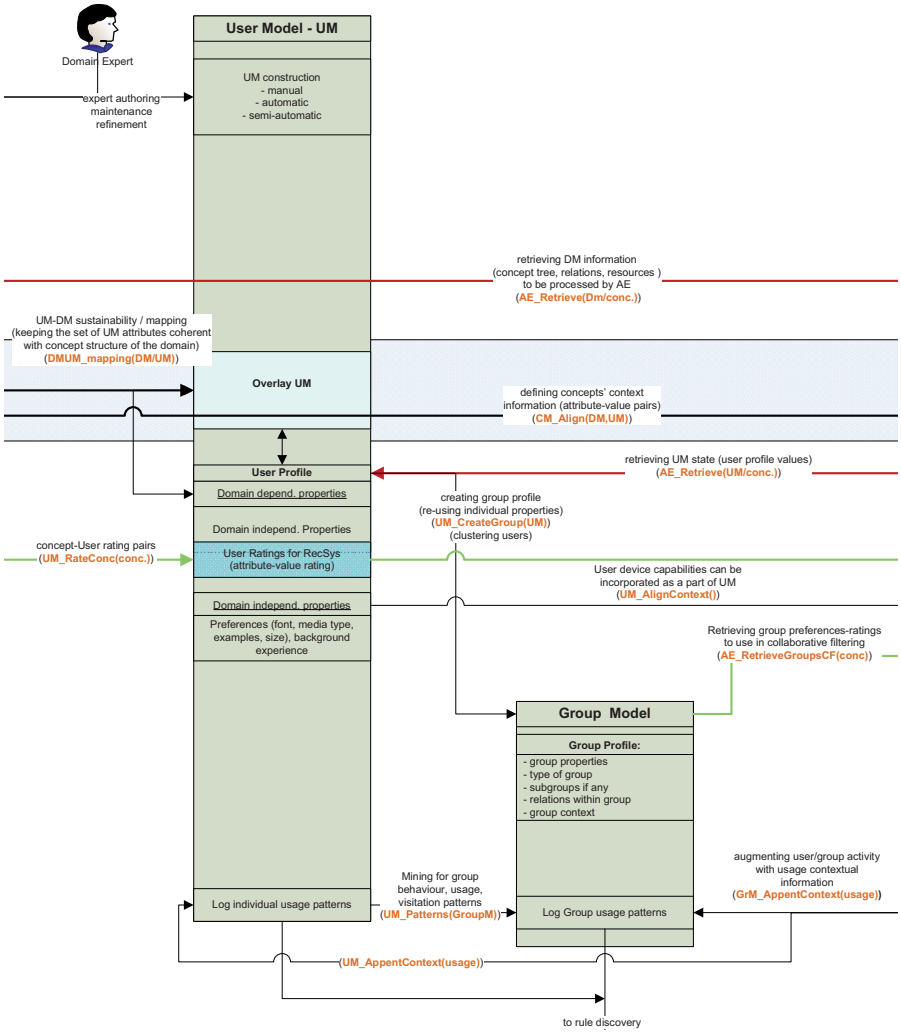


Figure 4.7: *GAF* User and Group Models architecture

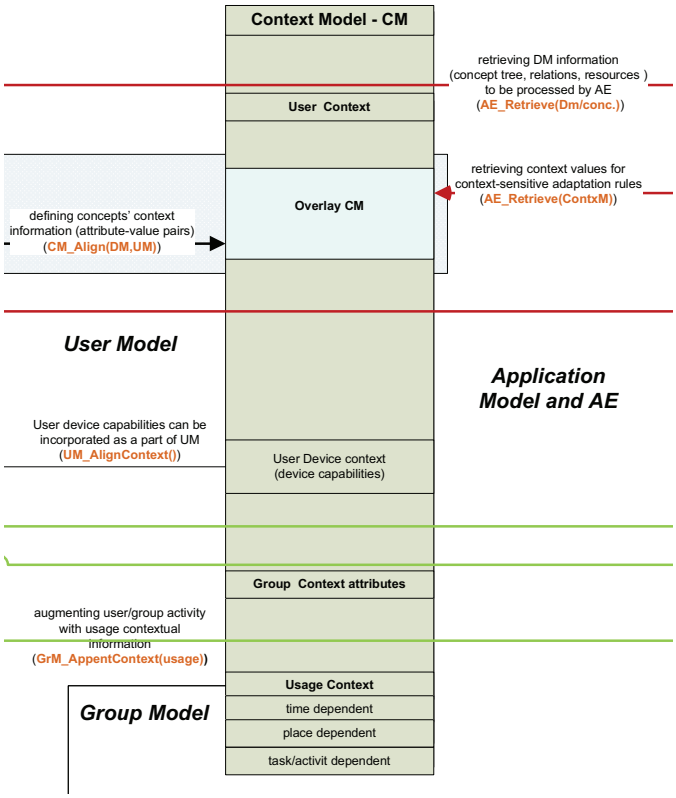


Figure 4.8: *GAF* Context Model architecture

4.5 Context Model (CM)

In this section we don't elaborate on the details of the context modelling but rather give an insight on how *GAF* can be helpful in resolving some of the issues. Moreover in chapter 6 we investigate some of these issues in particular by decomposing and analyzing real systems and application in terms of the *GAF* architecture. Figure 4.8 gives an overview of the basic structural elements and connection of CM while Table 4.6 describes its details.

Table 4.5: GAF User and Group Models

(sub-)component (question)	functionality	structure	relationship	advantages	interfaces / details
individual user profile (model) ("To What" is adapted)	represents individual user attributes (both domain independent and domain dependent (discussed in section 2.6))	structure of the user is determined by the application and is usually authored by an expert; when a pre-defined scenario is used UM can be automatically derived and includes application specific attributes (e.g. RS - user ratings; e-Learning - user knowledge and interest (overlay model is implied), etc.); common properties breakdown can be found in section 2.6 and the application scenarios are elaborated in chapter 5;	users' relationships are described within a user group (see section 3.8)	depends on the system implementation, however GAF considers UM and GrM separately and essentially means that any UM can be derived from GrM or the other way around when GrM can be created by aggregating UM properties of a group; UM and GrM can share the same 'overlay' properties;	in RS case *UM_RateConc(conc.)* allows users to rate individual concepts in order to identify interesting and relevant items (concepts); *UM_AppendContext(usage)* appends usage context to the user profile (e.g. time, place) (required by AM and AE)
group model	determining partitioning of the users into groups (Fugure 3.26)	describes group profile including group properties, types of the groups, relationships within group and the group context	types of users' relationships within the groups can be found in the scenario (Figure 3.27)	generic structure of the user and the group models (both instances of the same generic profile class) allow to apply the same type of reasoning and the same results representation;	*UM_CreateGroup(UM)* creates a user group (Fugure 3.26); *UM_Patterns(GroupM)* looks for a group behaviour patterns (e.g. needed to refine adaptation rules); *GrM_AppendContext(usage)* appends the current usage context to a group of users (e.g. when they are working on the same task);

Table 4.6: GAF Context Model

(sub-)components (question)	functionality	structure	relationship	advantages	interfaces / details
answer questions: Where, Who's, When, etc.	refers to a contextual information virtually of any model in the system as well as the usage of these models	taxonomy of context-aware features (mainly user and usage contexts and domain dependent and independent context features defined by the application usage/expert)	each context feature relates to certain GM, UM, DM, RM property and contributes to a contextual adaptation, contextual resource discovery, contextual presentation and link augmentation, etc.; in ideal case CM can be represented as an overlay over all UM and DM related features, which means that essentially every concept may have an arbitrary number of contextual attributes which are handled by AE;	separating CM from UM or any other context sensitive models allows to handle context features independently; decouple and make AH systems and applications less integrated with and dependent upon the context (incl. environment, device) in which they are used and more generic; adaptation to context may also be referred to as a higher order of adaptation, providing monitored results to devise new rules in a particular context;	*CM_Align(DM,UM)* aligns CM properties with the corresponding domain and user features (e.g. maps contextual features on the properties of a particular user group which require a special environment (place, time, etc.)); *AE_Retrieve(contxtM)* AE retrieves (task, user, usage, concept-related) contextual information required by the engine;

4.6 Application Model and Adaptation Engine (AM, AE)

In the coming chapters we conduct and discuss compliance and case studies where we show that depending on the application AM and AE might show a similar behaviour and complement each other in many different ways. Thus we show and discuss these two models in the current section and present them in Figure 4.9. We don't elaborate much on AM structure and interfaces since we come back to this discussion and the next chapters and investigate a number of scenarios and systems showing the diversity of the model. Table 4.7 summarizes major AM and AE aspects.

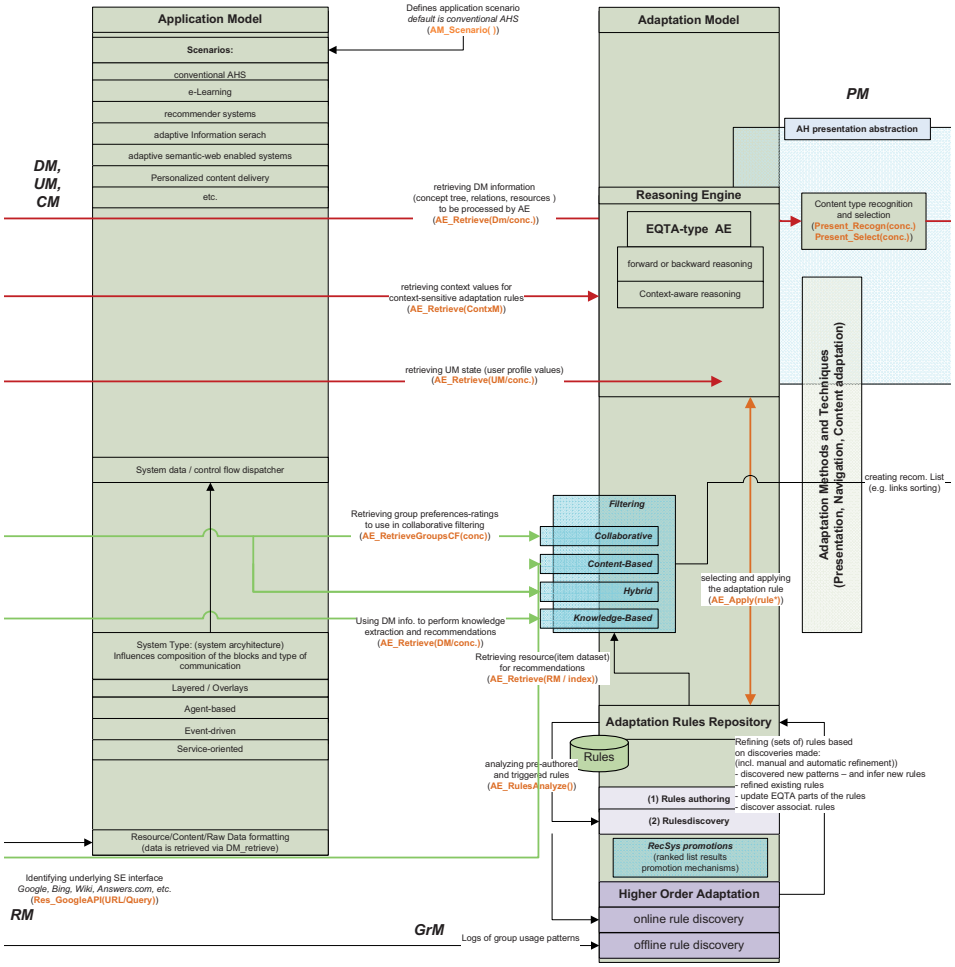


Figure 4.9: *GAF* Application Model and Adaptation Engine architecture

Table 4.7: GAF Application Model and Adaptation Engine

(sub-)component (question)	functionality	structure	relationship	advantages	interfaces / details
Application Model	describes the adaptation functionality of AHS at an abstract level (irrespectively of its actual content, domain); describes the scenarios' workflows of the application (usage pattern, user interaction, etc.)	depends on the system implementation and scenario (layered/tower, overlays, event-driven, agent-based, service-oriented, etc.);	couples together other layers of the system; describes the relationships in terms of work and data flows in the system (sections 3.3 and 3.4);	mediator for the other layers, as shown in chapter 6 may represent application framework, back-end or a server side handling data and control flows of the application;	*AM_Scenario()* defines the main application scenario which guides system composition and usage (e.g scenarios described in chapter 5)
Adaptation Engine	defines the adaptation functionality, resolves UM updates and presentation specification;	reasoner details about the structure and functionality insights can be found in section 3.7; the model also accommodates rule repository(s) and rules discovery facilities (Higher-Order Adaptation) employing users' and groups' usage patterns as well as triggered rules and AE updates analysis to derive new rules;	n/a	combines a number of reasoning approaches by distinguishing different parts of the E-Q-T-A rules and handling them separately; (detailed in section 3.7.1)	*AE_Retrieve(N)* where N is DM concept, Context, UM value, RM instance. This AE function retrieves corresponding values from all around the system in order to handle the reasoning procedure. Depending on the system may employ scenarios rom AM (e.g. retrieves group preferences from GrM to use in RS CF algorithm); *AE_Apply(rule)* applies a chosen rule (from Rule repository) to a concerned condition and values and triggers corresponding value updates; *AE_RulesAnalyze* in data-driven system is used to analyze and discover rules (e.g. association rules or user behavior patterns (based on the triggered rules));

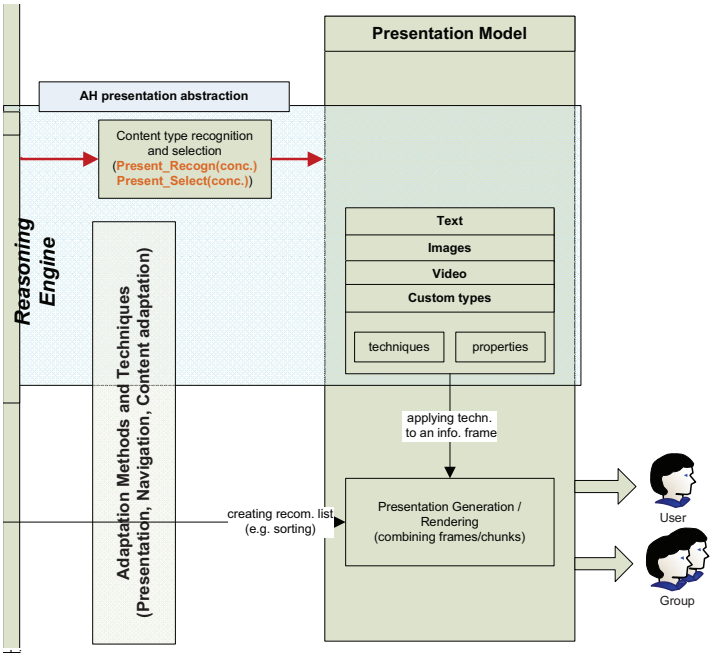


Figure 4.10: *GAF* Presentation Model architecture

4.7 Presentation Model (PM)

This sub-part of the framework has been discussed in details in section 2.3.5 where 'Adaptive Presentation Techniques Abstraction Layer' has been introduced. It essentially comprised the major part of the model therefore in Figure 4.10 we mainly show the connections to AE. These functions recognize the type of the content and select the associated content. based on the content properties and pre-authored system behaviour and appropriate adaptation technique is chosen and then the presentation is rendered and presented to the user or the group of users. In RS scenario there is no need to apply this while a particular set of pre-defined techniques to create a sorted list (considering a retrieved link or a snippet to be an atomic 'page' to be adapted) can be chosen.

Chapter 5

GAF Compliance Studies

The main goal of the *GAF* framework is to provide a reference architecture of AHS. *GAF* describes building blocks (both essential and optional) of a web-based adaptive system, defines the criteria to distinguish between these elements (pursuing the idea of the separation of "adaptation" concerns), describes their functionality and interaction. This modular structure (framework) can be used to describe applications and bootstrap the application development so that they satisfy different adaptation and personalization needs. This framework has a layered structure where layers match the original classification of AH methods and techniques and provide per layer functionality separation (in other words — separation of concerns). Using this classification we describe different functionalities within the same adaptation system layers depending on the requirements of the application and thus contribute to the system extensibility and heterogeneity. In this chapter we touch upon the questions of compliance and show the correspondence of AHS modeling (GAF layers in particular), adaptive courses, web search, recommender systems and other similar approaches.
This chapter covers a generic compliance study which precedes the idea of the framework validation in the more concrete system studies (from chapter 6) and is based on the definition of the *GAP* adaptation process explained in section 3.5. This chapter is based on research previously published in [80].

Definition 2 GAF compliance *is defined as the conformance between the GAF framework layered representation and description and any other given model, system or application considering their functionality decomposition (breakdown), as well as the correspondence of the adaptation process steps (both flowchart and the 'sequence' chart representation which is described in chapter 3.3). A system is compliant with GAF if it can be expressed and described in terms and definitions of the* GAF *framework, including functional building blocks (layers) and connections which comprise the adaptation functionality. Moreover we don't see the compliance as a one step procedure, it is an iterative process and each time we encounter a new (and generic) functional block in the evaluated system, it is added to the* GAF *reference architecture executing a* GAF *evaluation-architecture feedback loop.*

Investigating a number of scenarios in this chapter we not only describe them in *GAF* terms but also increase *GAF* vocabulary and thus reference functionality we will use in real system case-studies in chapter 6.
Hereafter this definition is used to perform the compliance study of a generic e-Learning application (an adaptive course) (section 5.1). A study of different recommendation methods is done and different types of RS filtering approaches are presented in terms and layers of *GAF* showing how different RS components match layers of the framework (Domain, User, Adaptation models, etc.) (section 5.2). And finally we show a conventional web search process and the way it is described by GAF. This allows to enable adaptation functionality and make the search personalizable (section 5.3). We deliberately use these three versatile examples in order to show essentially three different contexts of adaptation. These scenarios vary from an adaptive e-Learning application to a web-search. In each case we show that *GAF* can be used to capture and unify the adaptation functionality or define the methods how a system can be augmented with personalization features. More concrete system case-studies follow in chapter 6 but these compliance examples help to understand the idea of bringing together and expressing different types of systems and features in a single framework.

5.1 Adaptive course compliance study

The first *compliance* proof is presented in Figure 5.1. It describes an adaptive course use-case and discusses adaptation steps covered in the scheme. These (generic) adaptation steps were previously discussed in the generic adaptation block-schema in Figures 3.14-3.16.
The scheme in the figure covers the following steps (from top to bottom):

- Goal selection starts with the link selection, so that the possible goals correspond to the links the learner sees in the presentation; So a user selects the goal by following the first link and this goal may need to be elaborated in order to proceed with adaptation: In case when no goal elaboration is needed the system will proceed with content adaptation according to user knowledge and goal (next iteration); In case of a composite goal, goal elaboration will be done according to the goal repository structure, projects structure and so on until the most suitable goal sequence is found to fulfill user requirements (each iteration going one level deeper within the hierarchical structure if required).
- Having decided upon the goal (or a sequence of goals) it will be mapped on the DM concept structure to start with content adaptation.
- The current UM state will be acquired to proceed with the next step of rule parsing, acquiring the corresponding content and performing presentation generation until the system will go through all concepts designated to be learned by the user.

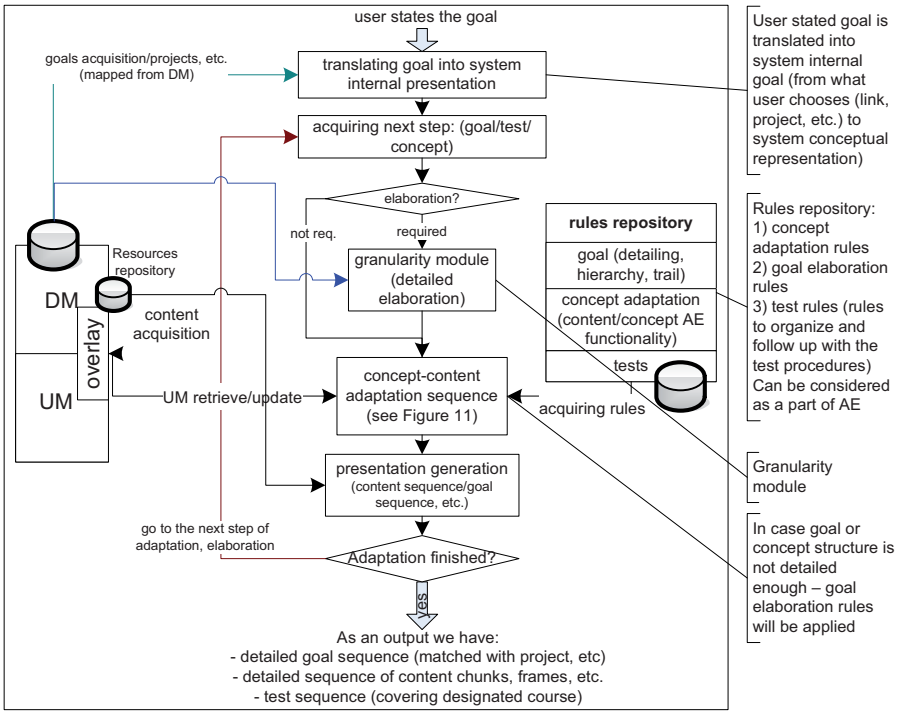


Figure 5.1: Use-Case: Adaptive Course Flowchart

5.2 Recommender System Compliance Study

In this section we will first summarize the applicability of GAF approaches with respect to the different types of RS and then explain the framework layers and corresponding functionality to show the *compliance*. In order to have a clear picture about what we're going to discuss in the section, basic elements of the framework and corresponding recommender functionality can be found in Figure 5.2.

Figure 5.2 presents the picture of compliance of a recommendation [114] and an overlaying *GAP* sequence chart. Here the generic process chart is constructed by coupling the layers of a general purpose AHS as in the previous case, however the functional purpose is different. We consider recommendation steps in terms of a single framework layer or a transition in the system and discuss this compliance presented in the figure further in this section. Though we were facing certain issues distinguishing Recommendation Engine functionality, in particular filtering and ranking mechanisms (in this respect Application and Adaptation models can be treated accordingly), we align recommendation and describe its functionality (in terms of aforementioned models) with *GAF* terms.

Considering RS we would like to mention the major types of RS and give a brief insight on the *GAF* structure and functionality related to these types of recommendations:

- *Collaborative Filtering (CF)* is usually based on UM and a Group model (GrM) (the same as in AHS) (we assume here both user-to-user and item-to-item aspects of CF). The recommendations are generated based on a comparison of user profiles from the GM and UM. In this case a Recommender UM is represented by a vector of items and corresponding ratings, which essentially can be turned into a set of *interesting* items or concepts (for this user) and associated attributes with corresponding rating values [89]. Those values are updated as the user interacts with the systems, browses through these items, rates them, and gets recommendations. In general CF allows to deal with many different types of objects where essentially only ratings matter. It becomes possible by separating ratings and item sets within the *GAF* Domain, User and Resource models.

- *Content-based Filtering (CBF)* RS recommends an item to a user based upon a description of the item - feature database (e. g. using words in a text as the textual feature or book genres as a library properties) and a user profile (essentially UM). Having learned features of these items (which can be expressed in the Resource Model (RM)) rated by the user, the RS infers new recommendation suggestions. As well as CF, CBF recommenders can also handle different types of objects as long as there is a common feature space.

- In *Knowledge-based Recommendation (KBR)* systems a domain expert knows which types of recommended items should be assigned to which types of users. In fact this is in line with current state of the art in AHS (i. e. there is a domain expert who needs to author UM, DM and Adaptation Rules mapping these two). It combines content-based filtering performed on the features of the concerned dataset with the explicit user query which is used to make inferences about needs and preferences of the the user. Thus it is possible to relate how a particular item meets user needs and thus to do the reasoning about possible recommendations.

- The *Hybrid type Recommender System* is the most commonly used type. These systems employ different approaches (simultaneously) to achieve better results, both combining techniques from collaborative, content-based and knowledge-based methods and providing different type of hybridization: mixed, weighted, cascade, etc. Our AHS framework allows us to combine both advantages of content and collaborative filtering, having all necessary building blocks available (e. g. user and group models, ranking mechanisms, reasoning engine) and will also help to handle heterogeneous data sources.

Hereafter we are going to summarize the *compliance* of the recommendation process with the reference AHS structure and explain the building blocks and the respective interactions presented in Figure 5.2:

- The user states the goal thus formulating a new recommendation query for which inferences over the user's preferences are made (particularly interesting in KBR

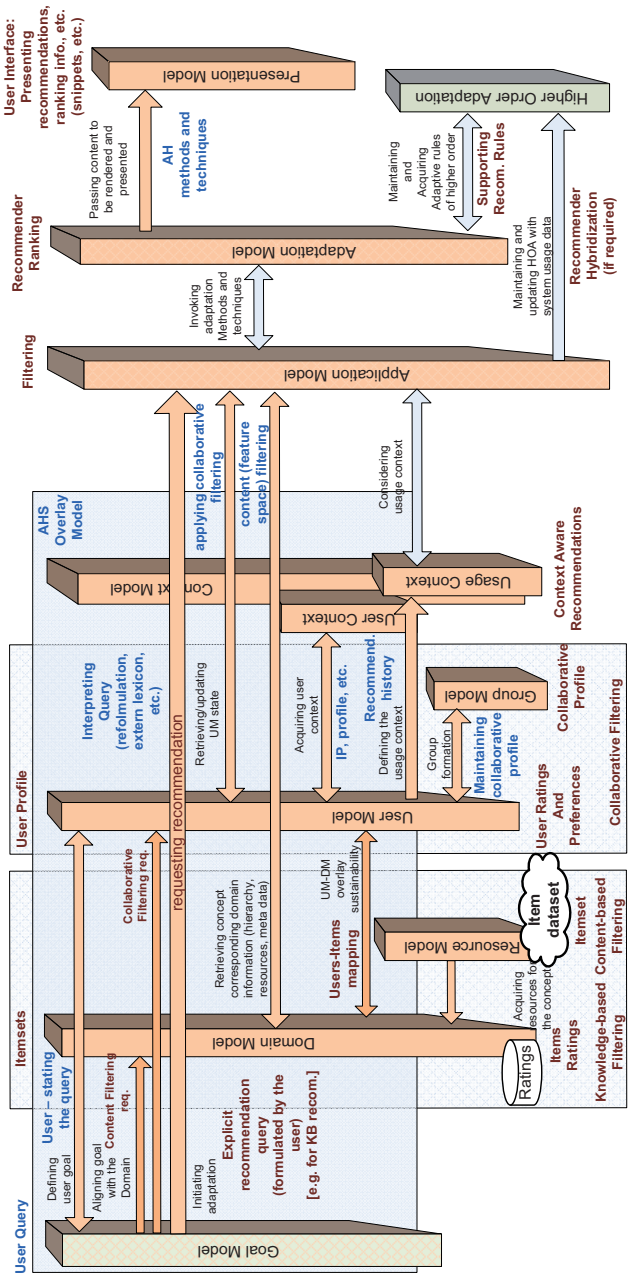


Figure 5.2: Recommender System Compliance With Generic Adaptation Process (GAP)

systems). This step can be considered as stating a goal or choosing a particular concept (set of concepts) to study/visit in an AHS. The goal can be interpreted and aligned with DM (availability of concepts, concept structures and hierarchies, etc.) and UM (considering user competencies, preferences, experience, interests, etc.). The same way the recommendation query can be reformulated, refined or aligned to match with the related user preferences using semantically related terms to get better recommendations by inferring closely related items (e.g. name of a favorite film director relates to a certain movie genre); UM is significantly important in collaborative filtering and corresponding recommender systems.

- DM is defined by the knowledge structure of the system domain, representing keywords and terms together with the relationships which can be used to facilitate fast and reliable information retrieval and filtering of the concerned itemspace, where the resources are defined by RM. On the other hand DM may represent the feature space in case of content-based recommendations where content is again retrieved from RM. It happens more often in recommender systems that the domain is represented by an ontology [32] in order to facilitate more elaborate reasoning over the items relationships and present more accurate recommendations.

- CM defines user and usage context properties such as IP address, time, other activities of the user, etc. Modelling Context gives a possibility to consider context-aware recommendations and adaptation, both from the usage and the user point of view.

- GrM refers to maintaining a collaborative profile of the user(s) or stereotyping filtered results by location or user age group and gender, which later can be used to rank and recommend results for a particular user or mediate user models associated with different groups [15]. The Group Model in general may represent and serve heterogeneous user groups by looking up commonalities in profiles to form groups or similarities in the group system usage (usage patterns) to recommend next best items both in the context of recommender and adaptive system/application for the users within the same group.

- Retrieving and updating UM refers to storing and accumulating users' rankings and recommendation history which can be used to reformulate system queries or retrieve personalized recommendations by finding similar patterns in the users' system usage.

- AM and AE refer to the Recommender Engine involving Filtering and Ranking mechanisms, however it may not be entirely clear how to distinguish some particular parts of those between layers of AHS. Here we would refer to the AE for ranking and recommendation rules. AM then couples other layers and dispatches information in AHS and Recommender respectively and performs a corresponding filtering method retrieving information from UM and DM for collaborative and content-based filtering respectively. Usually search and recommender engines are more robust and flexible for introducing or discovering new rules compared to the Adaptation Engines. However the rule systems which are conventionally used in

AHS can easily facilitate reasoning in recommender systems (e.g. ECA type of rules to determine static recommendation filters such as gender or location aware, or at the same time serve as the basis for semantic reasoning to look up for a related concepts that the user might be interested in). AHS will also provide the so-called *higher-order adaptation* capabilities. They will monitor the user's behaviour also to adapt the adaptation behaviour by discovering new or refining old rules.

- PM renders recommendation and adaptation results in such a way that a recommendation list is presented in the form of a ranked list, snippets, additional ranking information, result groups, etc. By applying AH presentation and navigation techniques [82] we may present not only ranked or sorted lists, but the whole new spectrum of interaction becomes available to enhance user experience in recommender systems (e.g. (de)emphasizing results in the list, summarizing results, navigating through the list, presenting contextual and non-contextual links, annotating, etc.)

In section 6.1 we will discuss a social recommendation system *HeyStaks* in details in order to fully cover *GAF* and its RS compliant functionality.

5.3 Adaptive Search Compliance Study

In this section we would like to discuss a generic web search process, investigate the possibilities to capture adaptation insights of this process and present it with the help of the same layered structure and functional description of *GAF* we used for describing RS or e-Learning systems.
Figure 5.4 presents a picture of a search process compliance with an overlaying *GAP* sequence chart. *GAP* represents the process chart constructed by coupling the layers of GAF which were introduced in section 3.6.
Here we assign search process steps from Figure 5.3 to a single layer or a transition in the system. Though we were facing certain issues discriminating Recommendation Engine functionality, in particular Search Engine and Ranking mechanisms (in this respect Application Model (AM) and Adaptation Model/Engine (AE) can be treated accordingly) we could align the search process and describe its functionality (in terms of the aforementioned model) with *GAF*. On the one hand this proves the generic nature of *GAF*, and on the other hand it opens new horizons to facilitate search aspects in the AH field and vice versa.
The search process complies with the reference structure of AHS (GAF) and the generic process sequence (GAP) as follows:

- *The user states the goal* thus formulating a new search query, which can be considered as stating or choosing a particular concept (set of concepts) to follow in AHS. It can be interpreted and aligned with DM (availability of concepts, concept structures and sequences, etc.) and UM (considering user competencies, preferences, experience, etc.). Thus this search query can be re-formulated, refined and enriched (e.g. matching it with the common lexicon or using semantically related terms).

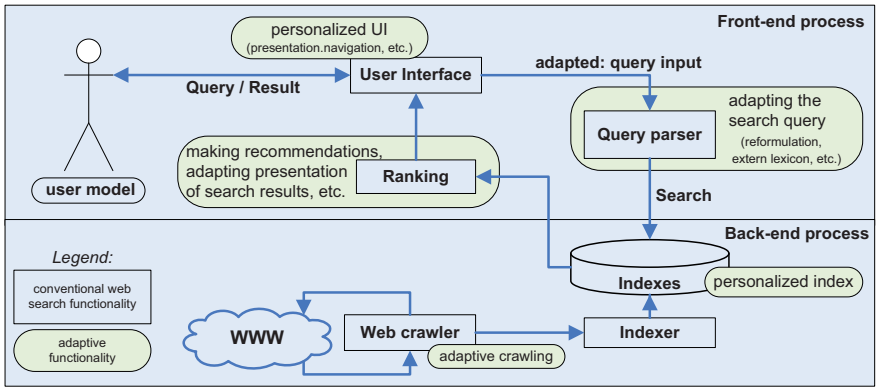


Figure 5.3: Web Search Schema

- *DM* is defined by the search index, representing keywords used to facilitate fast and reliable information retrieval, which is acquired from the resources *(RM)* (and essentially WWW). The index information is obtained from the web *by means of crawling / scraping* which is similar to the process of resolving content information of a concept in AHS.

- *CM* defines user and usage context properties such as IP address, user profile / stereotype, or search and result history of the users accordingly.

- *GrM* refers to maintaining a collaborative profile of the user or stereotyping search results by location or age group and gender, which later can be used to rank and recommend results by AM.

- Retrieving and updating UM refers to *storing and accumulating UM search history* which can be used to reformulate queries and/or retrieve personalized results.

- *AM and AE* may refer to the Search Engine and Ranking mechanisms, however it may not be entirely clear how to distinguish some particular parts of those. Here we would refer to the AE for Ranking, since they both to some extent perform adaptation of the results. AM then serves as the core of the system: coupling other layers and dispatching information in AHS or performing routines of a Search Engine.

- *PM* renders search results and presents a ranked result list, snippets, additional rank information, groups result (e.g. pictures, videos, text, etc.).

Summary: in order to make the use-cases from this chapter more convincing and outline further development of the framework compliance schema we continue with a set of system case-studies in the next chapter.

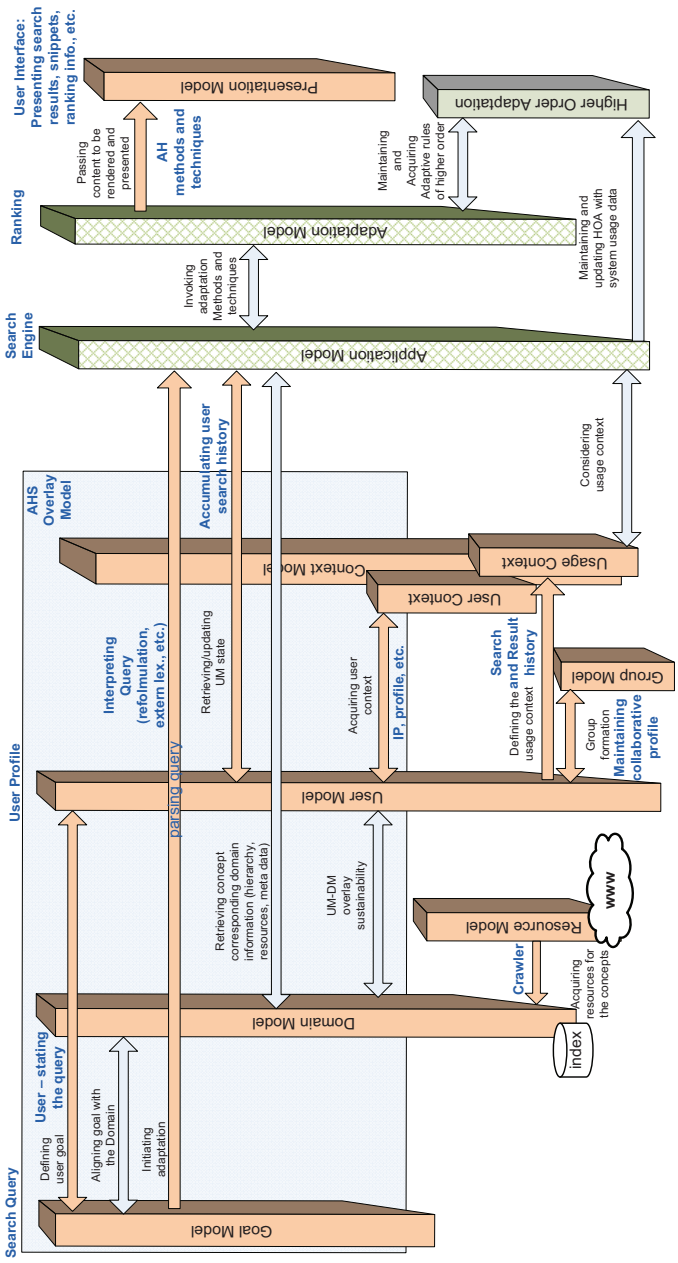


Figure 5.4: Web Search Compliance With GAP

Chapter 6

GAF Case-Studies

In this chapter we conduct a number of system case-studies and perform a detailed breakdown of systems' functionality blocks and elements in order to investigate them in terms of the *GAF* layered structure and the *GAP* process and conclude with our vision on formalizing AHS analysis and evaluation approach providing a toy example.
Case-studies in this chapter are based on the compliance studies presented earlier in chapter 5 and published in [80]. One of these studies (from section 6.2) has been previously published in [64] and the insights on the RPM case (section 6.3) have been presented in [135].
In this chapter's studies we have taken a number of personalization-enabled systems from different domains in order to show the diversity of *GAF*. We show that it may not only cover the functionality of the conventional AH systems as it has been investigated in chapters 2 and 3 but it can be also extended to cover the functionality of web-based data-driven systems such as recommendation systems (section 6.2) or personalized web search (section 6.1), applications from a medical domain responsible for a personalized content delivery (section 6.3) or semantic-web enabled systems (such as online tourist guides) (section 6.4)). We didn't include conventional AHS or (adaptive) e-Learning applications' studies in this chapter. One can find examples of AHAM, AHA!, APeLS and KBS-Hyperbook model and systems in section 3.5, where their adaptation functionality has been discussed in details and presented in the context of the *GAP* process. In the second part of this chapter we introduce AHS architecture analysis approach and present our thoughts on how to proceed analyzing systems taking into account the reference architecture and the notion of the system compliance.
And we will see in the conclusions that not all of the specific features can be covered by *GAF*, but as a matter of fact in the very beginning we couldn't even predict that it would be possible to combine such a variety of systems in one go (and describe them by means of the *GAF* framework). Besides, *GAF* is a dynamic structure and allows flexibility (as described in chapter 4) thereby the following case-studies helped us to identify those weak points of the framework and provide valuable feedback for the architecture (see the feedback loop in figure 1.1).

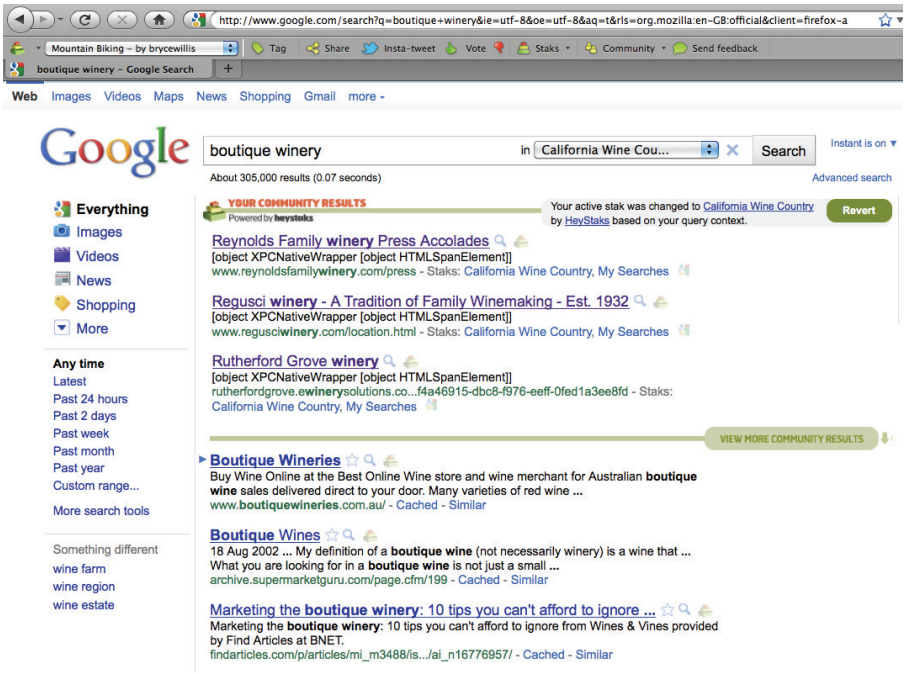


Figure 6.1: HeyStaks Browser look

6.1 *HeyStaks* case-study

HeyStaks is a social search personalization service that helps users to organize and share pages they find on Google and other search engines (Bing and Yahoo) and provides personalized search results (see Figure 6.1). Considering diverse aspects of the system and its functionality which is represented in the *HeyStaks* architecture overview in Figure 6.2 we decompose it in a way that forms an overlay of the generic model of an adaptive system, explains the functionality of the system using terms and definitions from the adjacent research area, and foremost brings a custom system to a common denominator by means of the GAF reference model.

6.1.1 *HeyStaks* process

Figure 6.3 presents a picture of 'HeyStaks' and the Generic Adaptation Process 'sequence chart' compliance. The GAP process chart is constructed by coupling the layers of a general purpose AHS as described in [85]. Recommendation steps are assigned to a single layer or a transition in the system. Hereafter we discuss 'HeyStaks' and GAF compatibility presented in the figure. Though we have faced certain issues distinguishing parts of the Recommendation Engine functionality, in particular the *filtering* and *ranking* mecha-

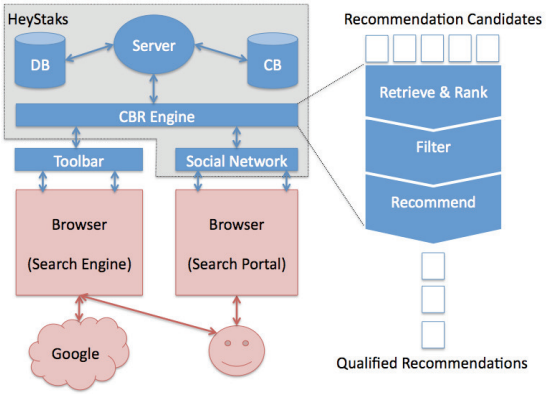


Figure 6.2: HeyStaks architecture

nisms (in this respect AM and AE can be treated accordingly) we could align 'HeyStaks' functionality with GAF terms and identify gaps and possible system extensions. On the one hand the mapping proves the genericity of the GAF framework, and on the other hand it opens new horizons to facilitate and generalize recommendation aspects, bringing adaptive techniques into place, extend information fusion and heterogeneity possibilities [26] of such a system encapsulating features of both Recommender and Adaptive Systems.

6.1.2 *HeyStaks* study: compliant and complimentary features

Below we summarize compliant and complementary 'HeyStaks' features and the reference structure of GAF and explain the building blocks and interactions presented in Figure 6.3, augmented with numbered references on the figure and in the discussion below. Of particular interest here are the remarks regarding AHS functionality (shown in GAF) that can be used to further extend 'HeyStaks', but also a few instances where 'Heystaks' functionality suggests further extensions to the GAF model.

- The User states a goal thus formulating a new recommendation query whereby some inference over the user preferences is made (*(1)* in Figure 6.3) (which is particularly interesting in knowledge-based RS). This step can be considered as stating or choosing a particular concept (set of concepts) to follow in AHS. The goal can be interpreted and aligned with DM (availability of concepts, concept structures and hierarchies, etc.) and UM (considering user competencies, preferences, experience, interests, etc.). The same way the recommendation query can be reformulated, refined or aligned to match with the related user preferences using semantically related terms to get better recommendations by inferring closely related items (e. g. name of a favorite film director relates to a certain movie genre). There is no explicit User Goal statement in 'HeyStaks', however users imply goals by choosing a *stak*

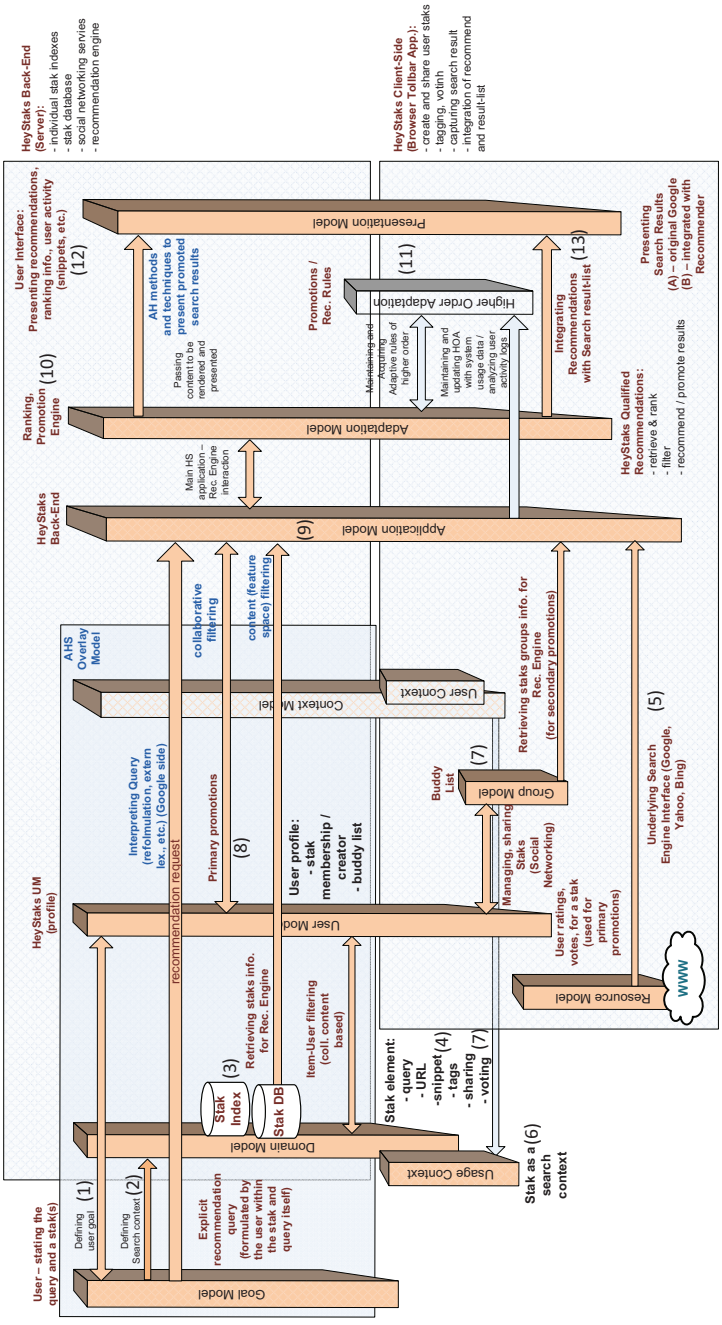


Figure 6.3: GAF - HeyStaks functionality overlay

in which they want to perform their searches *(2)*, thus narrowing down the outcome by getting promoted results within the *stak*. These could be the user's personal *stak* or a *stak* the user joined. (Staks are explained next.)

- *DM* is defined by the knowledge structure of the domain, representing keywords and terms together with the relationships which can be used to facilitate fast and reliable information retrieval and filtering of the concerned item space, where the resources are extracted from RM. On the other hand DM represents the feature space in the case of content-based recommendations where content itself is again retrieved from RM. It may also happen that the domain structure is defined by an ontology [32] in order to facilitate more elaborate reasoning over the item relationships and present more accurate recommendations. There is no explicit domain structure in 'HeyStaks', but the domain information is structured as a set of *staks (3)*. Each *stak* consists of elements representing search results, which can be shared, tagged, voted for, moved to another *stak* or deleted *(4)*. Together with the *stak* activity it represents a complete eco-system encapsulating domain information that is used to personalize search results, as well as the clear activity picture, easily understandable and manageable by the users.

- *RM* in AH describes the underlying resources of DM from where the actual content that needs to be adapted is retrieved. RM is tightly connected with the idea of the Open Corpus Adaptation. This is where an AHS operates on an open corpus of documents, e. g. a set of documents that is not known at design time and, moreover, can constantly change and expand [24]. In other words - Open Corpus AH systems can operate on the web scale set of resources, just like 'HeyStaks' uses Google or Bing search results to show personal promotions before other results *(5)*. Open-corpus systems in general provide the flexibility of search, recommendation and navigation in one go.

- *CM* defines user and usage context properties such as location inferred from the IP address, time or device used. Modeling Context gives a possibility to consider context-aware recommendations [3] and adaptation, both from the usage and the user point of view. 'HeyStaks' doesn't take contextualization into account. However users' and staks' activities are rich and apart from those manual actions that are provided to a user (e. g. share, move between staks, tag), they can be automatically analyzed in the future to find some stable usage patterns or classify user activities in or outside the search *stak*. On the other hand, considering just the user interaction with the system every search query has a context, which is a *stak* used to narrow down the results *(6)*.

- *GrM* refers to maintaining a collaborative user profile. It usually clusters results by location or user age group and gender, and uses it to rank and recommend results for a particular user or mediate user models associated with different groups [15]. The Group Model in general may represent and serve heterogeneous user groups by looking up commonalities in profiles to form groups or similarities in the group

activities or system usage (usage patterns) to recommend next best items both in the context of recommender and adaptive system/application for the users within the same group. Though there is no explicit group modeling involved in 'HeyStaks', it uses a simple but at the same time elaborate way to encapsulate group related information both in the UM and *stak (7)*. Each *stak* has a creator and a list of members, while each user has a profile and a list of buddies. Thus all the users can be grouped based on their queries, *stak* tags, buddy lists, and their collaboration activity (e. g. adding each other, sharing *staks* and re-using search queries of each other within publicly available *staks*).

- Retrieving and updating UM refers to storing and accumulating users' rankings and recommendation history which can be used to re-formulate or re-use system queries or retrieve personalized recommendations by finding similar queries. 'HeyStaks' does this all by keeping the user activity logs *(8)*.

- Application and Adaptation Models (*AM* and *AE*) may refer to the Recommender Engine involving Filtering and Ranking mechanisms, however it may not be entirely unambiguous how to distinguish some particular parts of those between layers of AHS. Here we would refer to the Adaptation Model for ranking and recommendation rules, since they both to some extent perform adaptation of the results *10)*. The Application Model then serves as the core of the system: coupling other layers and dispatching information in AHS and Recommender respectively and performing a corresponding filtering method retrieving information from UM and DM for collaborative and content-based filtering respectively *(9)*. Usually search and recommender engines are robust and flexible for introducing or discovering new rules on the fly, which is not always the case with Adaptation Engines in AHS. However the rule systems which are conventionally used in AHS can easily facilitate reasoning in RS (e. g. ECA type of rules to determine static recommendation filters such as gender or location aware, or at the same time serve as the basis for semantic reasoning to look up for a related concepts that the user might be interested in). AHS will also provide the so-called 'higher-order adaptation' capabilities *11)*. They will monitor the user's behavior and also to adapt the adaptation behavior by discovering new or refining old rules. The 'HeyStaks' recommendation engine uses a rich history of user search experience and profiling techniques (where the *stak* serves as a profile of the user search activity) to promote result candidates. *Stak* usage data provides an additional source of data that can be used to filter results and generate the final recommendations.

- *PM* renders recommendation and adaptation results in such a way that a recommendation list is presented in the form of a ranked list, snippets, additional ranking information, result groups, etc. By applying AH presentation and navigation techniques [82] this may be presented not only like a ranked or sorted list, but a whole new spectrum of interaction becomes available to enhance user experience in RS (e. g. (de)emphasizing results in the list, summarizing results, navigating through the list, presenting contextual and non-contextual links, annotating and explain-

ing the choice, etc.) *(12)* On top of search results 'HeyStaks' does some of this *(13)*: it sorts according to the personal *stak* preference or provides the possibility to preview. More than that it gives a possibility to provide immediate feedback on the result (vote, share, tag) which changes your UM and influences subsequent searches and ranking of results.

6.1.3 Conclusions

We mapped AHS layers (GAF in particular), Recommenders and HeyStaks functionality and presented it in the Figure 6.3. As previously outlined in Figure 5.2 for different types of RS, here we retain the structure and building blocks of the system and show the architecture as well as the process overlay of the 'HeyStaks' social RS.

6.2 *Twittomender* case-study

In recent years a lot of progress has been made in the field of RS. New efficient models and algorithms have been developed [114], heterogeneous and hybrid systems are gaining wide use. Amongst those RS personalized search and twitter are getting into place [63, 128].

Since its first emergence on the scene in early 2006, Twitter has grown from an early microblogging engine into a real time web behemoth. In the early days some 300,000 tweets were produced by early adopters of the system per month, contrast this with some 140 million tweets estimated to be produced per day in march 2011[1]. With this obvious information increase, creating ways to use this information appropriately and intelligently has become a hot topic for many researchers [108, 53]. In Twitter the main producers of this content are the users themselves who have subscribed to the system. Obviously, not all the information produced by each user is to everyone's preference, so finding the producers, the so-called 'diamonds in the rough', is an interesting research challenge. And, with the *Twittomender* system we have chosen to frame this as a recommender systems.

Most of these AH models, including the emerging and discussed in the thesis GAF model focus on a layered architecture and discuss the adaptation of the content and navigation to the user properties as well as recommending the links to follow based on the user preferences and knowledge, thus bringing the fields of AH and Recommendation closer together. In fact we show that a web-based RS can be treated as AHS and therefore expressed in terms of a generic AH framework, which we backup by the conducted case-study of the *Twittomender* system.

In the section we focus on describing RS functionality and *Twittomender* [63] in particular in terms of AH systems, in order to show complimentary features of RS and layers of a generic AH framework (GAF) which aims to develop a new reference model for the adaptive hypermedia research field by considering new developments, techniques and methodologies in the area of AH and adjacent fields (including but not limited to RS) [80].

[1] C. Penner, #numbers, mar, 2011, http://blog.twitter.com/2011/03/numbers.html

We show framework compliance (conformity in structure and partial functionality overlay) with the *Twittomender*, a Twitter based recommender system.
Besides *Twittomender* is a very interesting example, covering topics such as personalization, recommendation, search and collaborative web experience which means that we can bring and fit all these features together in the framework. As a result we also achieve an evaluation of how generic the GAF framework could be on the one hand and whether the description of one particular RS fits in the framework, and helps to evaluate the real system on the other.

6.2.1 *Twittomender* recommender overview

The *Twittomender* system's main function involves syncing a user's account and producing followee recommendations through a range of collaborative and content-based strategies. However for this to work efficiently, users must be active on Twitter, i.e. they must follow a number of other users, must have some followers themselves and must have produced some content (through tweets). Although this functionality is great for Twitter users who wish to increase the number of appropriate user streams they follow, it does not perform satisfactorily for new Twitter users. These users have not produced much content through tweets, nor are they following or being followed by enough users for collaborative or content-based followee recommendation techniques to perform as expected. For this reason we also provide a search capability to *Twittomender*, which allows users to explicitly type search queries. For our collaborative and content-based strategies we evaluate 9 different profiling and recommendation strategies based on the different sources of profile information, in isolation and in combination. To begin with we implemented 4 *content-based* strategies that rely on the content of tweets as follows:

1. (*S1*) users are represented by their own tweets ($tweets(U_T)$);
2. (*S2*) users are represented by the tweets of their followees ($followeestweets(U_T)$);
3. (*S3*) users are represented by the tweets of their followers ($followerstweets(U_T)$);
4. (*S4*) a hybrid strategy in which users are represented by the combination of tweets from $tweets(U_T)$, $followeestweets(U_T)$, and $followerstweet(U_T)$;

In addition we implemented 3 *collaborative* style strategies, in the sense that we view a user profile as a simple set of user ids.

5. (*S5*) users are represented by the IDs of their followees ($followee(U_T)$);
6. (*S6*) users are represented by the IDs of their followers ($follower(U_T)$);
7. (*S7*) a hybrid strategy in which users are represented by the combination $followee(U_T)$ and $follower(U_T)$;

Additional 2 strategies are:

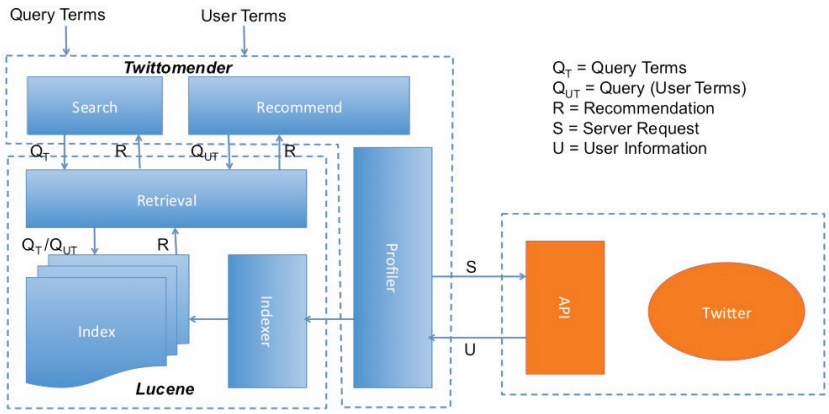


Figure 6.4: Twittomender architecture

8. (*S8*) the scoring function is based on a combination of content and collaborative strategies S1 and S6;

9. (*S9*) the scoring function for this strategy is based on the position of the user in each of the recommendation lists so that users that are frequently present in high positions are preferred;

6.2.2 Recommender System and AHS: Twittomender Study

Considering diverse aspects of the system and its functionality we decompose it in a a way that forms an overlay of a generic model of an adaptive system, explains the functionality of the system using terms and definitions from adjacent recommender systems research area, and foremost brings a custom system to a common denominator by means of the GAF reference model.
Figure 6.5 presents a picture of *Twittomender* and the Generic Adaptation Process (GAP) *sequence chart* compliance. Here the GAP process chart is constructed by coupling the layers of a general purpose AHS as described in [80]. Recommendation steps are assigned to a single layer or a transition in the system. Though we have faced certain issues distinguishing parts of the Recommendation Engine functionality, in particular the *filtering* and *ranking* mechanisms (AM and AE accordingly) we could align *Twittomender* functionality with GAF terms and identify gaps and possible system extension. On the one hand the mapping proves the genericity of the GAF framework, and on the other hand it opens new horizons to facilitate and generalize recommendation aspects, bringing adaptive techniques into place, extend information fusion and heterogeneity possibilities [26] of such a systems encapsulating features of both Recommender and Adaptive Systems.

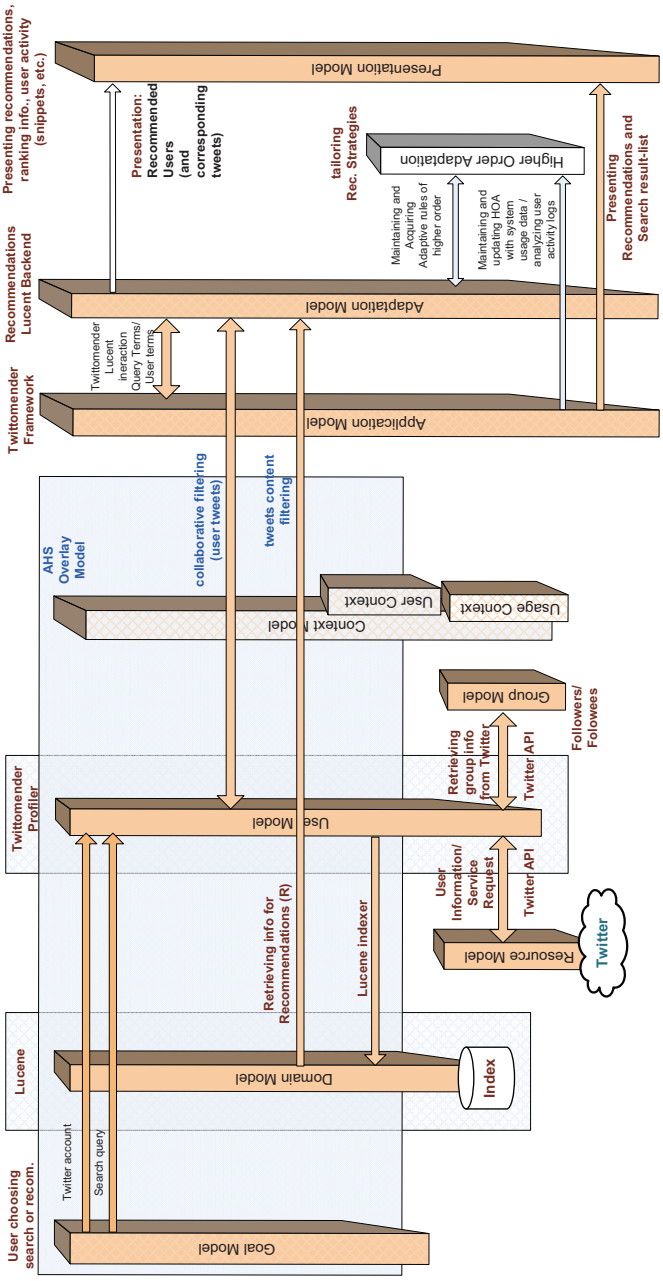


Figure 6.5: Twittomender compliance with Generic Adaptation Framework

Further we summarize compliant and complementary *Twittomender* features and the reference structure of GAF and explain the building blocks and interactions presented in figure 6.5. Of particular interest here are the remarks regarding AHS functionality (described in GAF terms) that can be used to extend *Twittomender*, but also a few instances where *Twittomender* functionality suggests further extensions to the GAF model.

- Users start system interaction by choosing whether to get recommendations directly by logging in with their Twitter profile or by entering a search query they are interested in. This refers to *GM* of GAF. Internally goal is represented by the immediate query input by the user or constructed from the indexed content of the users tweets when he or she log-ins into the system.
- Twittomender Profiler serves both as *UM*, by associating each user with the corresponding group of followers and followees, and at the same time as user information mediator which requests tweet content information from Twitter services and provides this information to the Lucene indexer which forms the index of user tweets and such forms the domain model to be used in recommendations.
- *GrM* refers to maintaining a collaborative user profile is already provided by Twitter services. It clusters results by location or user age group and gender, and uses it to rank and recommend results for a particular user or mediate user models associated with different groups. To some extent Twitter services provide this possibility by maintaining the groups of followers, followees of any given user.
- *DM* of the *Twittomender* is represented by the index which is stored by the Lucene (backend).
- *CM* (both user and usage models) are not considered.
- *AM* is represented by the *Twittomender* framework. Mainly it serves to query terms from the Lucene and retrieve corresponding ranked lists of users and related tweets. *Twittomender* framework also provides interfaces to PM.
- *AE* as described in a generic Recommender system use-case is represented by indexing and actual querying solution, Lucene. Its Information Retrieval module provides querying interfaces to *Twittomender* and return recommendation lists upon querying (both User Terms and search Terms as indicated in GM. The actual index is stored in DM providing flexibility of the system and at the same time decoupling Lucene as a stand-alone Query/Retrieval mechanism.
- *PM* generates ranked list of users recommended to follow and corresponding cloud of indexed terms that are relevant to the user activity in Twitter.

6.2.3 Extending *Twittomender*

Based on the case-study and some of the conclusions we immediately thought of improving the *Twittomender* system. According to the separation of the functionality concerns

that we could distinguish using the *GAF* framework we wanted to extend the user modelling and personalization capabilities of the *Twittomender* system and combine then with existing information from the personal Google profiles that many users already have.
From the Google personal profile we could extract the personal history (which is essentially versioned and takes up the last 7 days, 30 days, a year or all time historical information) as well as personal search trends (which are also provided by the Google infrastructure including top queries, top sites and top links). Based on this retrieved RSS history feed, we create an new personalized index which is used by *Twittomender* to provide recommendations.
The retrieved Google-based record includes the *title* - which is either a query string or a title in case of queries or result pages respectively; *description* (denoting the number of results) which can be used as a weighting factor in RS; *unique ID*; and *categories*: *web query* interpreted the same way as it is used to query relevant and interesting terms in *Twittomender*, and *web results* treated the same way as the Tweets index (in *Twittomender*) of a particular user. After parsing this retrieved RSS history feed we construct the input for the *Twittomender* in a form of an index, which from this point on can be considered as a personalized index and will help to deliver user related twitter recommendations and to a certain extent overcome a cold start problem by retrieving and constructing a personalized index from an existing Google profile. Besides that users should feel more comfortable getting a result relevant to what they have been previously searching on the web using Google search engine. The basics are shown in figure 6.6.
To summarize the extension does the following:

- it provides recommendations taking (Google profile) user modelling and time changes into account;
- ideally user should be able to log into *Twittomender* using their Google credentials;
- *Twittomender* treats the Google constructed UM (input result index or/and query);
- each UM profile is constructed using a versioned search/query term based on the day/week/month time window;
- the same time alignment is done in twitter;
- versioning criteria appear to be very useful since both Google and Twitter use time-windows to treat historical information and we consider these two aspects complimentary;
- since we are using the existing *Twittomender* recommendation mechanism, there is almost no impact on the existing system, we only feed a personalized index into the system.

Essentially Google profiles and personal search history are used to construct the personalized index to be used in order to provide *tweets* and *followers* recommendation in the *Twittomender* system.

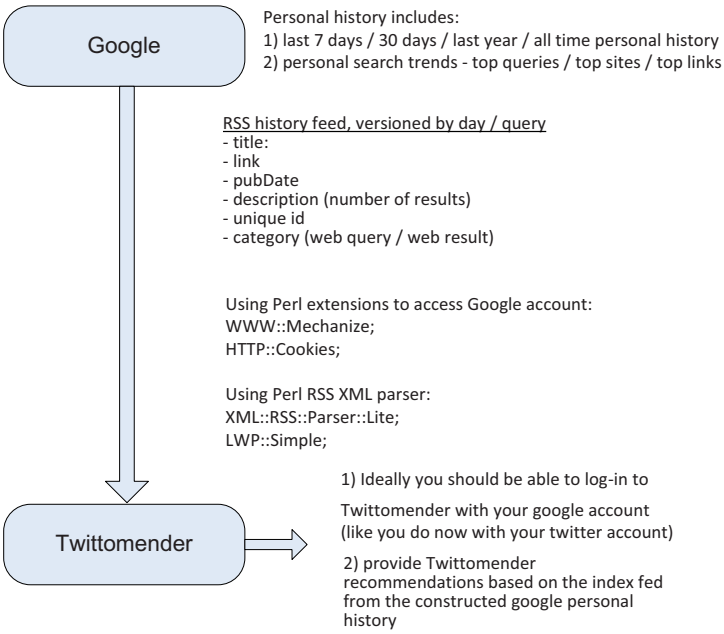


Figure 6.6: Twittomender extension schema

6.2.4 Conclusions

First and foremost this case-study helped to identify possible *Twittomender* improvements and extensions. It is planned to make some further developments to *Twittomender*, one avenue which we are exploring is a mechanism to focus on users individual personal traits. What topics do user's talk about? What types of people do they follow? It is also planed to extend the *Twittomender* platform to cluster similar users based on these traits. This will allow Twitter users to quickly navigate to the types of people they would normally tend to gravitate towards or conversely show them the topics they would be clustered into e.g Sports, Technology, etc. And this is what we could already foresee by looking into the system through the *GAF* "lens".

6.3 Remote Patient Management system (investigation) case-study

Remote Patient Management Systems (RPM), besides monitoring the health conditions of patients, provide them with different information services that currently are predefined and follow a one-size-fits-all paradigm to a large extent. In this section we focus on the

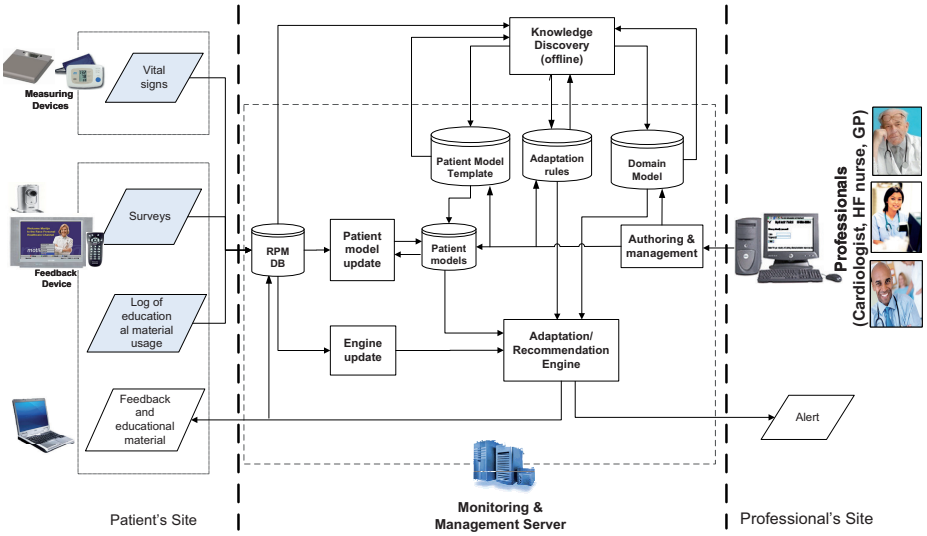


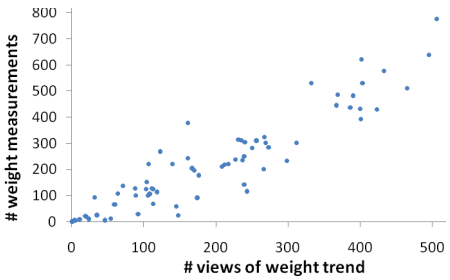
Figure 6.7: RPM systems schematics: current state and history

problem of knowledge discovery and patient modeling by mining data, motivational and instructional feedback provided to patients within the RPM system.

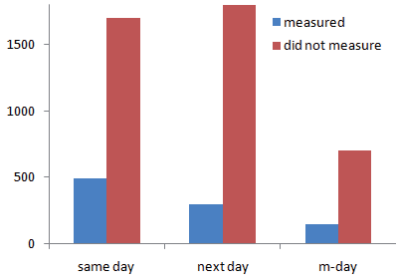
A general picture of such a system is presented in figure 6.7. But here we would like to focus on the design of the framework aimed at facilitating the adaptation and personalization of the information services by discovering actionable patterns from educational material usage data as well as motivational and instructional feedback, and linking those with the conditions and quality of life of the home-monitored patients. We sketch an RPM framework based on the *GAF* architecture (and considering separation of concerns principles of the *GAF* adaptation), that can be implemented and provide motivating examples based on our preliminary study of one real RPM dataset.

We would like *GAF* and AH technology in general to become an integral part of an RPM systems. The output of the knowledge discovery process will be utilized for patient modeling and providing input for the adaptation engine. One type of practically relevant questions related to patient modeling and adaptation includes e.g. "what kinds of patients are likely to weigh themselves regularly if they review their weight charts"; "what is the relationship between the patients reviewing the charts and watching educational videos or reading motivational messages", "do the patients (or what kind) restart weighting after receiving a message that they forgot to do so". Two examples drawn from the real RPM dataset, collected during a clinical trial, are shown in figure 6.8.

The preliminary results of our exploration study suggest that there is a potential of building upon the *GAF* framework to facilitate data-driven patient modeling and motivate the shift from the one-size-fits-all approach currently employed in the development of RPM systems to personalization in providing educational materials, motivational support and



"Is the patient who reviews his weight charts also more likely to weight himself regularly?"



"Do the patients restart weighting after receiving a message that they forgot to weight?"

Figure 6.8: RPM motivational example charts

informational content to the users. An anticipated architecture of the RPM system build upon GAF is presented in figure 6.9. In this figure rounded rectangles and data base symbols represent basic modeling elements (Domain, Resource, Patient and Professional, Adaptation and Presentation models). All the connections are annotated according to the interaction processes taking place. We also elaborate key elements of the patient models (UM) here as the main focus group of the RPM system.

Further work here has been discussed with the RPM system experts and may include the many-sided analysis of the RPM usage database with the focus on the educational content and usage data. The particular focuses include subgroups discovery and identification of signatures describing well-doing home-monitored patients and those who require more assistance of the medical staff, which is entirely covered by the facilities offered by GAF framework thus facilitating modern RPM systems development and deployment.

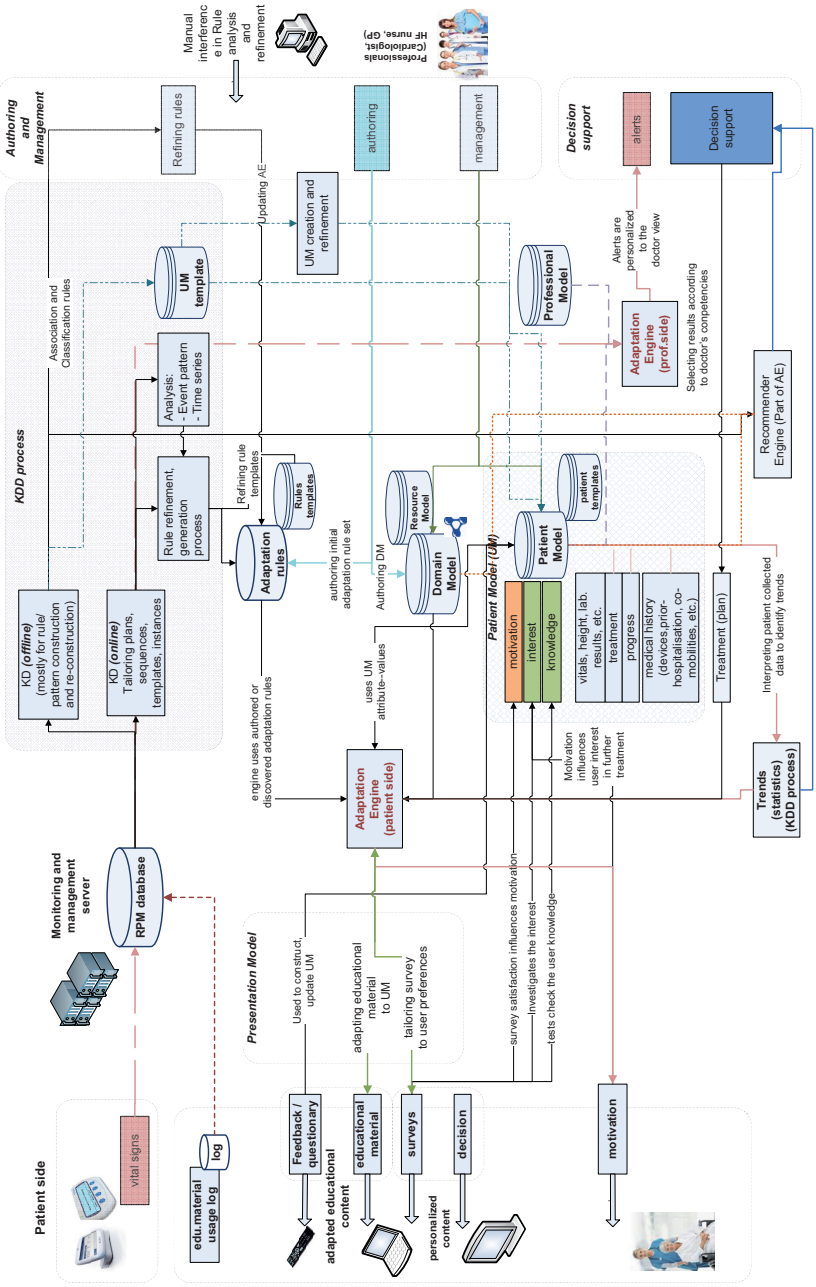


Figure 6.9: GAF-based framework for personalization of information services in RPM systems

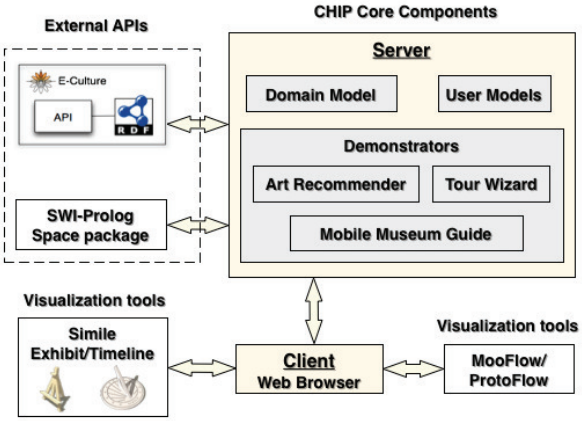


Figure 6.10: CHIP architecture

6.4 *CHIP* art recommender case-study

Employing Semantic Web techniques to combine user, domain and adaptation models from different applications based on the same ontology is a feasible problem, but when different ontologies are used, the problem of ontology alignment must be tackled first, making the reasoning over such a structure more challenging. AHAM presented earlier can almost handle the single ontology case but has no provision for dealing with multiple ontologies, while dealing with multiple CHIP ontologies require manual mapping. In section 3.10 we have provisioned modelling and interoperability questions and hereafter would like to discuss a particular case-study. It concerns the compliance of a semantic web-enabled recommender system *CHIP*[2] and as in previous use-cases described the CHIP system in terms of the *GAF* blocks.
The prototype CHIP architecture is presented in figure 6.10. It shows a composition of the system and based on the client-server architecture which makes use of two external APIs (including access to the E-Culture RDF storage to support the semantic interoperability of the system). Schema in figure 6.11 elaborates CHIP functionality and decomposes it from AH perspective.
GAF-CHIP modelling components are:

- *GM* in CHIP is presented to the user through Demonstrators (Tour Wizard, Mobile Guide, Art Recommender) and can be interpreted a rating goal, viewing or a tour offered to a user. All these goals generate events which are translated in the corresponding UM updates and system interaction.

- *DM* is implemented using a semantically enriched 'Rijksmuseum' collection stored in RDF/XML format and maintained by Sesame Open RDF memory store. It uses

[2]http://chip-project.org/

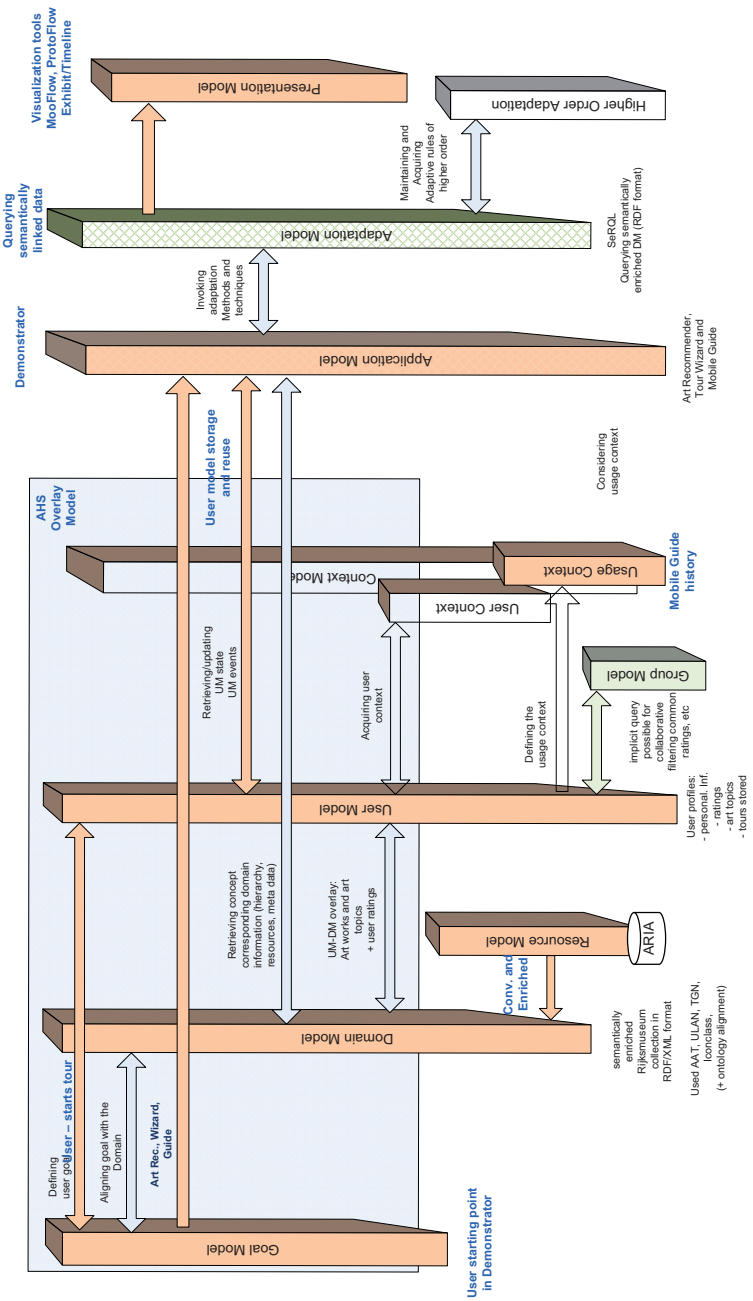


Figure 6.11: CHIP-GAF compliance

the corresponding query language SeRQL to retrieve the information from DM and associated resource base (RM).

- *UM* represents information about CHIP users such as personal information, ratings of the artworks and art topics, tours (stored in 'Turtle' format). It is implemented as an overlay over the DM structure and consists of attribute-value pairs representing the aforementioned user properties and thus fully resembles conventional UM overlay modelling used in AH. As mentioned earlier user goals are translated into UM events. These events are: Rating - when the users rates an object (artwork or an interesting topic) with a value from "-1" to "1"; Viewing - an event which comprises a Tour event; and the Tour event which orders user added art objects.
- *RM* in CHIP is represented by a converted ARIA data base (conversion to RDF was done within 'Topia' project[3]) and is queried using SeRQL query language as well as DM. ARIA provides a web interface to manage the data base and is used to provide the information for the general visitors of the museum website.
- *CM* is not entirely considered in CHIP, however a few steps in distinguishing mobile and desktop applications are done. This can be treated as a contextual adaptation to a device capabilities.
- *AE* - basic AE model functionality is based on querying (using SeRQL) DM. It retrieves only those elements from DM (and corresponding descriptions) which are interesting to the user (rated positively by the user). Thus only these items are presented to the user and only relevant items are recommended.
- To present information in CHIP *MooFlow* and *ProtoFlow* technologies are used. They render the lists of artworks in a carrousel in the Art Recommender and Mobile Museum Guide applications. Simile Exhibit and Simile Timeline javascript web applications serve the Web-browser client and are applied for data presentation. Thus the presentation is based only on the decision made by AE and is always provided in the similar way, varying in the end-user applications only (mobile vs. full screen application).

6.5 GAF System Architecture Analysis Approach

The evaluation of any AHS and essentially any type of the adaptive web system plays an important role. Compliance and system studies have shown this. Described in [106] the layered evaluation provides a description of the system functionality and helps to solve many related problems. In this section we consider a more formalized and specific system analysis approach by taking up systems' block composition scenarios and interfaces described in chapter 4 which might significantly help in further system evaluation and analysis. We defined the dependencies between models, methods they use to communicate and particular implementations (based on the usage scenarios). As a starting point for

[3] https://www.cwi.nl/Topia

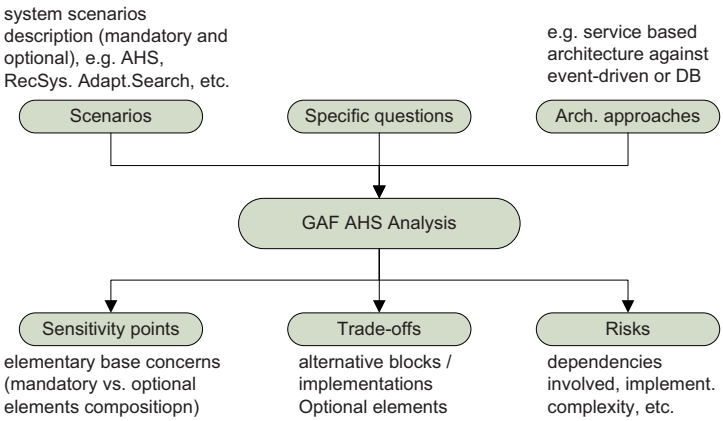


Figure 6.12: AHS analysis approach

our analysis we took an approach from [73]. The main steps of this analysis are presented in Figure 6.12. By scenarios we mean framework use-cases (adaptive search, adaptive eLearning, recommender system presented in chapter 5), which were published in [80]. We also consider system aspects and AHS building blocks composition which impacts the architecture (see chapter 4). The introduction to this AH architecture analysis method has been previously published in [79].
As a result we will have an elementary concerns of how to build AHS, explaining mandatory and optional building blocks of the system, trade-offs available (mostly concerning optional elements of AHS) and the dependencies involved. We have essentially elaborated this approach in the use-cases.

6.5.1 GAF Analysis Approach (overview of DM Example)

In this section we elaborate the analysis approach and consider DM as a sample model. In Figure 6.13 we show an overview of the model interface dependencies. Analyzing it further we comprise the dependency table of building blocks' interfaces (such as Domain, Use, Resource, Context models), scenarios of how these models are used and which type of a system is being described (AHS, Adaptive eLearning, Recommender System, etc.), possible technologies used to implement it (Data Bases, OWL ontologies for semantic web enabled systems, TF-IDF index for search, etc.). This idea of how this table should look is presented in Table 6.1. As a result we'll have a detailed picture of the system components, evaluated against the reference model (*GAF*), which helps to identify system's pros and cons.
Considering any arbitrary DM properties and interfaces of any given system, we analyze them against the following properties and methods of the DM reference structure (see Figure 6.14 for details). The major division here concerns methods and properties of the abstract Domain Model class. Further we distinguish classes (like sets or collections of

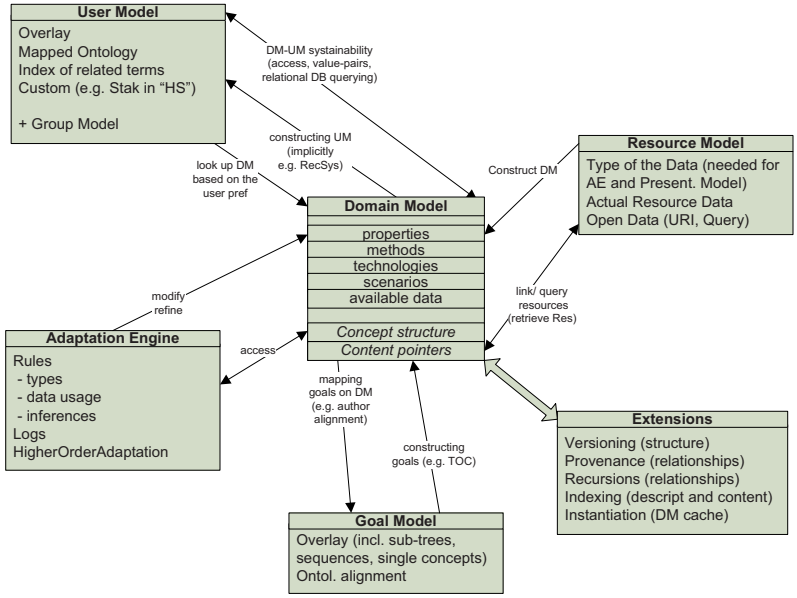


Figure 6.13: Domain Model interface dependencies

Table 6.1: partial GAF blocks high-level dependencies: DM example

DM properties and methods	Scenario	Resource Model	Adaptation Engine	User Modelling
concept tree	conventional AHS eLearning	content pages/frames	ECA reasoning, prerequisites relations	UM overlay
feature space	recommender system	datasets	promotions and ranking mechanisms	implicit user profiling
index	adaptive search	WWW	ranking	implicit user profiling

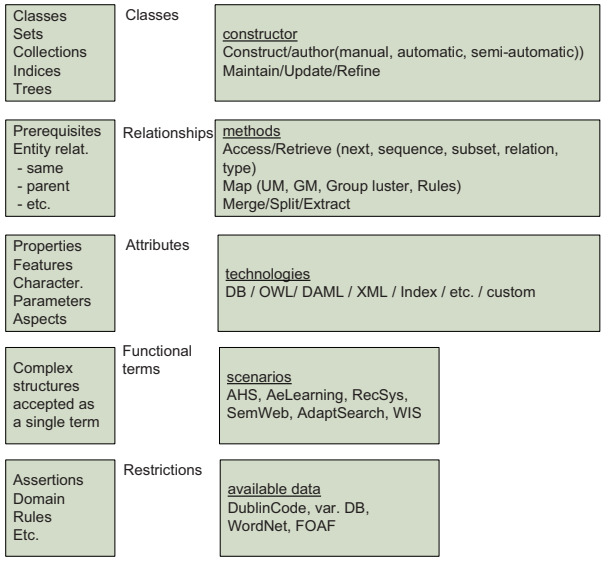


Figure 6.14: Domain Model abstraction class

concepts or concept maps, indices, trees, etc.), relationships (which are conventionally constituted by the ontology relationships), attributes of the concepts (e.g feature space, properties, characteristics, etc.), then functional terms which are denoted by complex structures usually treated as a single term, and restrictions defined by assertions or some specific domain rules.

Methods can be defined by constructors used to author DM as well as refine, maintain or update it. Major DM methods describe the access and retrieve procedures mainly called by User Model (UM), Resource model (RM) and Adaptation Engine (AE) to access the conceptual structure and query corresponding content. We also define mapping methods which are used to maintain structure sustainability especially in overlay type of models or ontology mapping for instance. These mappings (or alignments) can be done between DM and User, Goals, Groups models and Rules sets. Additionally we have methods to merge, split and extract sub-models of DM, which can be used in distributed domain modelling or open corpus adaptation.

DM scenarios describe the system behaviour in terms of functional flow and user interaction. We have described most prominent use-cases of such a framework compliance with different types of systems in [80]. Thus the DM usage in different cases could be analyzed against these reference scenarios.

Finally we have a number of particular technologies to work with DM and associated or cross-technology data available to start modelling (e.g Dublin Core to devise adaptive eLearning application or a dataset feature list to devise recommender system or adaptive

search portal). This may remind us of the UML notion used in [87] to formalize the AHS modelling, however we define more strict dependencies in the GAF formalization through defining interfaces, methods and scenarios, besides we use it to analyze system, identify alternatives and be able to compare and assess other systems in terms of the GAF framework. Table 6.1 presents high-level dependencies between DM properties and methods, scenarios and other AHS' models. This is just to give an idea of our approach, ideally these dependencies would be described in meticulous details, parametrizing abstract DM interfaces and to some extent show concrete technology or implementation approach for each of these models' interfaces. Eventually this approach with the combination of the the architecture description (chapter 4) and the system's compliance study give a full system evaluation and analysis picture.

Chapter 7

Technologies Leveraging Adaptive and Personalization Functionality

In this chapter we consider a number of interesting and yet important near-AH technologies, in other words the technologies that can tremendously leverage adaptive and personalization functionality in the field of AH. This chapter might be treated as the 'future work in progress' and not only envisions these future directions in AH but shows their importance and outlines the benefits.

First we are going to discuss the importance of provenance data in AH and essentially any type of personalization web-based system (section 7.1). We re-visit AH modelling questions and find out the way to align these questions with a widely recognized *W7* provenance model. Moreover we align not only the questions but also the answers thus enriching the conventional AH modelling approach (section 7.1.3). We conclude this part of the chapter by bringing up the examples of the provenance importance (section 7.1.4) and summarize with the provenance modelling problems and prospective solution (section 7.1.5).

In the second part of the chapter we describe a so-called 'versioning' framework which helps to store, manage and personalize various information aspects considering the ideas from the fields of source control and web-archiving. We show that versioning methodologies allow to tackle information capture and retrieval in a convenient and easy way by presenting basics of versioning. We consider some use cases of applying AH methods and techniques to overcome visualization overload issues and conclude by summarizing benefits of bridging AH and versioning technologies. The 'versioning' part is organized as follows: first, we revise versioning methodologies and operation (section 7.2.2) and information overload issues from AH perspective (section 7.2.4). Then we describe AH personalization aspects (section 7.2.5). Based on the real examples (surveyed in section 7.2.8) we show a number of 'representative' use-cases (section 7.2.9). We present our view on an adaptive proxy architecture (section 7.2.9) and finally outline the advantages and perspectives of combining these technologies (section 7.2.7).

Provenance related sections of this chapter have been previously published in [85], and our 'versioning' ideas and approaches have been elaborated and published in [77, 84].

7.1 GAF Provenance Modelling

A hypermedia application offers its users navigational freedom within a large hyperspace. AH offers personalized content, presentation, and navigation support. Most AHS do so by building UM and using that to guide AE. The subject of UM scrutability has been studied extensively [72] because users want to be able to review (scrutinize) what the system knows about them. Adaptation scrutability still remains largely uncharted territory: most systems are not set up to explain to users why the content, presentation and navigation are adapted the way they are. We take a new approach towards offering scrutability by studying the parallels with the area of data provenance. In general provenance data aims at providing users with an explanation of the origin, history and evolution of data and processes.
In this section we re-examine the adaptation questions stated in the very beginning of the AH era [23, 82] in the context of data provenance. In fact we match the major questions of adaptation (*What, To What, Why, Where, When, How*) with a question-driven provenance model (see table 7.1.2). Our major goal here is to show how complementary to each other adaptation and provenance are (in terms of question-based classification) (section 7.1.3). We also present a number of demonstrative examples of data lineage, harvesting and interpretation importance in AH (section 7.1.4). And finally we investigate what the problems of designing provenance support in AHS are (section 7.1.5).

7.1.1 Provenance

Provenance is information about the origin, ownership, source, history, lineage and/or derivation of an information object or data. Provenance is important as it is vital for providing the detailed explanation of user action, system usage and data origin and inference, ensuring analysis of dependencies in the system and repeatability of user actions.
Provenance Modelling: There are several provenance modelling approaches:

- **Data-centric**: refers to meta-data models such as Dublin Core[1], Premis[2], OAIS[3], etc., where a metadata schema stores the provenance data; this was for instance presented in [105].

- **Process-centric**: refers to the description of the process with particular change steps through which this metadata is obtained. It collects not only the data about a particular step, but about the application processes as well [70].

[1] http://dublincore.org/
[2] http://www.loc.gov/standards/premis/
[3] http://public.ccsds.org/publications/archive/650x0b1.pdf

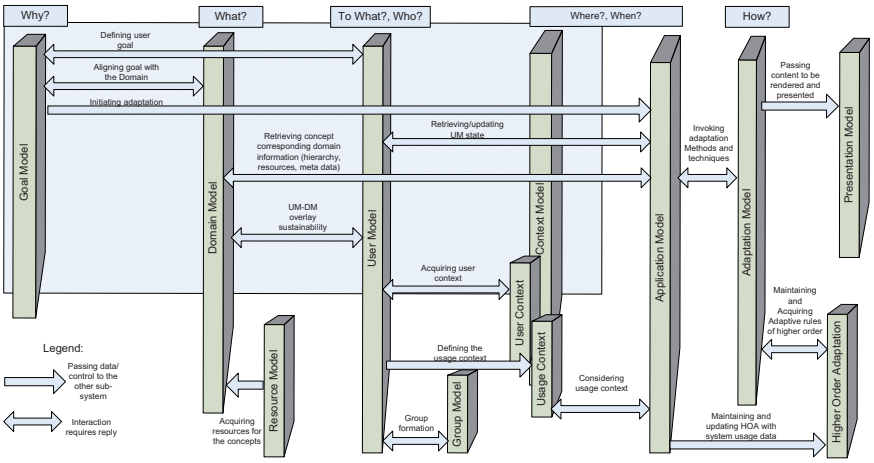


Figure 7.1: Conceptual Adaptation Process Sequence

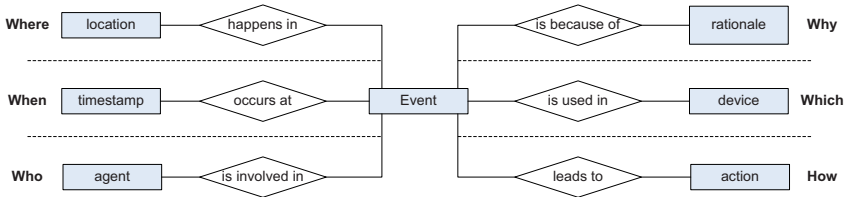


Figure 7.2: *W7* Provenance Model [112]

- **Pipeline-centric**: as investigated in [59] a new pipeline-centric approach to provenance data was introduced for the class of workflow-based applications, which helps to determine the provenance of the application output based on the provenance graph of the application.

We believe that process and pipeline type of obtaining provenance data [59] can be complementary with the adaptation process shown in figure 7.1. These two have very much in common with the process of collecting data that could be used for the provenance analysis. By *Adaptation Process* we mean the interaction in AHS which starts with the goal statement, exploits features of the user and domain models in different contexts and adapts various aspects of the system to the user. This sequence of user-adaptation actions could be aligned according to the classification of AH methods and techniques as well as Provenance questions and results in the adaptation sequence, coupling the 'layers' of AHS (figure 7.1).
Possible alternative and less generic classification of provenance models includes but is not limited to:

- In the *Data Specific Provenance Model*, each data type has its own provenance

model, carrying forward information covering the complete path of the data. This approach benefits in a way that all provenance metadata comes with each result, but lacks models interoperability.

- The *Generic Complete Provenance Model* retains provenance information in the form of prior data sets and transformations. This approach has the advantage of being very complete, however it requires storage of intermediate results and hardly be visualized and analyzed in a simple way.

- In the *Hierarchical Provenance Model* only the provenance information covering the previous transformation is retained, but at the same time all the provenance data can be recursively returned. This type of retaining hierarchical data may correspond to a hierarchical structure of DM. Thus it is complementary in such a way that it keeps track of the user following the conceptual structure in depth gaining more detailed knowledge on a particular subject represented with this hierarchy.

We leave out the question of a particular provenance modelling approach, but rather consider the generic model of provenance information. We will use *W7* which is a provenance model representing diverse information about the data produced in the system [112].

7.1.2 Adaptation and Provenance Questions

Adaptation Questions

There is a number of adaptation questions that have to be answered in order to build an AH application [82]. Moreover they define the adaptation process aligning the structure of system sub-components. These questions also denote the adaptation process flow.
The core of adaptation is defined by stating and answering six major questions:

- What can we adapt? (*What?*)
- What can we adapt to? (*To What?*)
- Why do we need adaptation? (*Why?*)
- Where can we apply adaptation? (*Where?*)
- When can we apply adaptation? (*When?*)
- How do we adapt? (*How?*)

We want not just to revisit these questions, but to address the issue of aligning them (and corresponding answers) in a common, modular structure of a generic AHS architecture.
By answering the major adaptation questions we elaborate the adaptation process description outlined in figure 7.1 and consider it in the context of the provenance model. This process is usually initiated by the user stating the adaptation goal and thus answering the "Why adaptation is needed?" question. Then in the process we consider the "What?"

and "To What?" questions, which emphasize DM and UM description. "When?" and "Where?" in this process go further providing context and application area definitions. Lastly, the, "How?" question describing methods and techniques on conceptual and implementation level and finally all together resulting in an AHS description.

Provenance Questions

Hereafter we will consider one of the most extensive definition of the provenance models and investigate the way it can be complementary to AHS, possibly after some extensions. According to [112] ***Provenance*** is defined as a *n-tuple P = (What, When, Where, How, Who, Which, Why, Occurs_at, Happens_in, Leads_to, Brings_about, Is_used_in, Is_because_of)*, which represents the *W7* model [112], where:

- *"What"* denotes the sequence of events that affect the data object;
- *"When"*, the set of event times;
- *"Where"*, the set of all locations;
- *"How"*, the set of all actions leading up to the events;
- *"Who"*, the set of all agents involved in the events;
- *"Which"*, the set of all devices;
- *"Why"*, the set of reasons for the events.

According to the model an action is taken by *agents* using *devices* for *reasons*, which is reflected by the various relationships existing between "what" and the elements "who", "which" and "why". The conceptual schema of the *W7* provenance model is presented in figure 7.2. In other words these questions may also describe such information as: event decision (what), duration (when), activity (how), method (which), person (who), arguments and justification (why).
This scheme shows a schematic representation of the *W7* model. In table 7.1 we will consider Provenance questions side by side with Adaptation questions and aim at aligning them hence extending AHS with the notion of provenance.

7.1.3 Aligning questions

Considering the question-centric, extensive definition of the *W7* Provenance Model [112] and the AH methods and techniques classification questions [82] we combine and align the questions and corresponding answers. Such an alignment will be able to provide complementary features description. Here we investigate commonalities and similarities in the semantics of the answers and meanings of these questions, emphasizing the idea that provenance information can be very useful in AHS and at the same time provenance information can help to reason in AH, for example providing more explanations to the

Table 7.1: Aligning Adaptation and Provenance questions

Questions	AHS	Provenance Model	Comments
What?	Domain Model	denotes the sequence of events that affect the data object	answers describe the sequence of events when the user gets access to the domain information and acquires domain knowledge
Who? (To Whom?) Which?	describes the user profile selection (or/and device usage) (e.g. can be used to select a group or target users)	the set of all agents and/or devices involved in the process	
To What?	UM attributes (selecting particular attributes that are accessed and updated within the concerned adaptation process)	no actual representation in terms of provenance question, however historical information on accessing and updating UM represents provenance information	
Why?	stating the adaptation goal(s) (might be a domain concept, representing either a new goal to follow or a sequence of concepts)	the set of reasons for triggering a particular event (evidence of what has happened)	reasons and goals are complementary, indicating the premises of the adaptation process
When? Where?	Application Model (which serves as the core of the system: coupling other layers and dispatching information in AHS) and Context information keeps track and interprets the context information	the set of event times and locations	contextual information in general
How?	describing AH methods and techniques on a conceptual and implementation level (Adaptive Engine (AE) functionality); explains the sequence of event-actions; ***describes the semantics*** of cause-effect relations	the set of all actions leading up to the events (keeping track of the events, and corresponding action in the system); ***describes the syntax*** of events and actions recorded	in pair provenance and AH describe the syntax and semantics of AE functionality (record events and actions and show cause-effect relationship)

end user or making the system more trustworthy. In table 7.1 we map questions and look for common understanding in-between Provenance and AH.
"What?" — answer to this question on the one hand describes the way domain information is represented in the system (hierarchy of concepts, ontology, etc.) and on the other hand shows what events in the system these data objects can affect.
"Who?" ("To Whom?"), "Which?" — answers to these questions give us an idea of the UM environment: *Which?* defines the device capabilities and in general *Who?* represents the user profile. They also describe the set of devices and agents involved in the process from the provenance point of view and can be used to select the target group of users, representing the high-level user division and defining the group adaptation parameters.
"To What?" — answer narrows down the user profile to a particular set of attributes involved in the adaptation process (accessed and updated by the system to retrieve or refresh the current state of the user knowledge, interest, competence, etc.). These are usually domain dependent attributes. There is no actual match on the provenance question here, however the history of UM attributes' access and updates directly refers to storing and harvesting provenance data from user logs.
"Why?" — answer determines the set (one-at-a-time or a sequence) of goals of adaptation and describes the set of reasons for initiating the concerned adaptation process. Thus, these two indicate the premises of the adaptation process in general, provide arguments and describe the way adaptation is initiated.
"When?" and ***"Where?"*** — answers are registered as a part of the provenance model events. The AHS keeps track of these changes and interprets this data to be used as the input for the reasoning component, which should take into account this time and place contextual information.
"How?" — answer provenance data records event-action sequences, describing mostly the syntax of these changes, on the other hand AHS describes the semantics (understanding of these cause-event relationships), contributing to the picture of the reasoning model. As a whole it describes AE functionality of the system.

7.1.4 Examples of Provenance Importance in AH

We have mentioned some of the motivating examples in the introduction, but we would like to extend this list and provide more insight on the importance of AH provenance. We consider the following examples to be significant:

- *Adaptation*: provenance data can be directly used in the adaptation process. Being interpreted by the AE to determine the result of the next adaptation step, it may extend the capabilities of the adaptive reasoning from a conventional pre-authored type to become more context and provenance/lineage dependent, taking into account not only UM values and updating them, but analyzing the origin of these UM updates and thus adjusting them accordingly (e.g. AE may interpret the knowledge source properties and assign different scores to the user depending on the source trustworthiness).

- *Explanation and Analysis*: explanations not only include information about system

usage (e.g. the prerequisite knowledge level is reached and the user can access new information) but also where this information comes from, how it was derived, etc. (e.g. the user is provided with an additional explanation about the origin of their 'knowledge' or 'interest' which may come from an update event issued by the system interpreted in a way the user can understand). This could be useful both in providing additional explanation and recommendations to the user and in the analysis of the system behaviour by the domain expert.

- *Usage Patterns Analysis*: could be helpful to discover and analyze certain abnormal user behaviour patterns (in combination with information retrieval methods, e.g. provide the dependency of unusual data and the source of it using the provenance information). Essentially provenance data can be mined to discover these patterns and used to analyze the origin of such a behaviour.

- *Information reliability*: provenance regarding how the adapted information was delivered to a user helps to ensure that it can be trusted so that the user understands the way he received the information and is explained why he gets this and where it comes from.

- *Information currency (prevalence/efficacy)*: capturing provenance such as when the update to UM is done could be used to avoid being misled (e.g. by outdated information).

- *Semantics of provenance*: provenance data provide a semantic extension to the system, expanding the description of the data to what is answered by the provenance questions. For instance the data providing the information of the data origin will extend the semantics of each particular delivered information unit. Thus provenance information "naturally" extends the semantics of the existing data model [111].

- *Adaptation Process*: considering process and pipeline centric types of provenance information we anticipate mapping the notion of provenance on the adaptation process shown in figure 7.1. Each step of the adaptation sequence here represents a single data transformation from the data lineage point of view, answering aforementioned questions of Provenance and AH models.

7.1.5 Provenance Issues and Prospective Solutions

In this section we would also like to indicate some of the issues that one may face while investigating provenance of data in AHS. These are only some of the most common problems:

Harvesting of provenance data. Since AHS usually has a rather complex structure the problem of data harvesting arises. That's why we proposed a layered structure (figure 7.1), strictly distinguishing AH data (essentially adaptation questions) and process, using major AH classifications. This will provide transparency of the system functionality, reduce (by associating type of provenance data with the layer) places of provenance data origin and help in analyzing the data.

Understanding the semantics of provenance. Understanding provenance enables users to share, learn the meaning and take advantage of data, facilitating collaboration and learning, reducing the number of deadlocks and providing mechanisms to conceptual modelling [112]. We have partially covered this issue in this 'provenance'-related section trying to understand and match the semantics of provenance and AH data (in terms of questions).
Diversity in data types and many places of data origin. As mentioned above, considering a generic layered structure of AHS we try to put things in order and clearly distinguish between the major questions first of Adaptation and then Provenance, thus reducing and foremost matching overall places of origin diversity. Anticipating the layered structure of AHS we foresee the idea of clear separation in such a way that already mentioned harvesting of the provenance data becomes transparent and clear which simplifies data analysis and reduces the number of ambiguous places of data origin.
Moreover as investigated in [83] some of these provenance problems related to data dynamics, diversity and overload and in particular issues of storing, retrieving (harvesting) and analyzing (interpreting) data in AH can be facilitated by using versioning techniques. There are many more problems to be considered, such as reusability and alignment of provenance data, different reasons for needing provenance, its representation and propagation. Finally implementing and using provenance could become an issue. Without fulfilling most of these requirements it may cause misuse and misinterpretation of the AH related data (e.g. misaligning user and a domain knowledge) and lead to unpredictable adaptation results (e.g. termination and confluence problems).

7.2 Versioning in AH

7.2.1 Introduction to *Versioning* in AH

An ordinary user is now overwhelmed by different types of information flows (sending and receiving e-mails and news, editing documents, using social networks, and other daily routines). We receive a lot of information (and it is constantly growing) but can't handle it all. As a way to handle information overload issues we propose to leverage two different technologies such as Adaptive Hypermedia (AH) and version control to handle these issues in one go. We would like to investigate information overload issues from two perspectives: from personalized visualization mechanisms exploiting AH methods and techniques, to maintaining and to some extent personalizing the versioned structure by creating adaptive versions of accessed information. Systems like *DiffIE* [127] or *SIS* (Stuff I've Seen) [51] can be considered as good motivating examples of bringing personalization and storing the historical data together. In this and following sections we scrutinize basic concepts of versioning in source control and adjacent fields, as well as the research concerned version management in other fields like Open Hypermedia (OH) structures [58], Semantic Web ontologies versioning [74] and leverage *versioning* in AHS [83] which provide adaptation and personalization to a concerned user filtering out irrelevant or not important information.

7.2.2 Capturing, accessing and retrieving versioned information

Structured, reliable and transparent logging facilities may be crucial in authoring, maintenance, support, behaviour or process analysis and eventually system control and user support. Based on the differentiation used in the source control systems (e.g. ClearCase, SVN) we distinguish automatically and manually controlled information capturing (or gathering). Essentially most of the source control systems provide these opportunities either to perform automatic or interfere and supervise this process. They also provide simple commands for retrieving the information, and can be extended with the third-party tools to gather detailed reports (such as code or text changes analysis and metrics).

Automatic information capture

Automatic information capture refers to the system's capabilities to store the information about the change. This usually happens when the user accesses, modifies or deletes information elements from a data repository. Automatic information can be applied by means of scripts or system agents (e.g. applying labels or creating try-out branches on an element's access events, performing trivial comparison and merge operations), however these operations initially require manual control, but after some proper treatment and test procedures can automatically function on their own.

Manually-controlled information capture

The meta-data annotations to a concerned version can be appended to an object manually, or alternatively by a pre-defined script triggered on a check or an event within the system (e.g. a new version has been committed to a repository, user has retrieved content from a previous version and compared to a current instance, etc.). These meta-data annotations usually include:

- Version labels/tags/branches - Version attributes are usually represented with the labels, tags or can be denoted by a particular branch of the element which belongs to a user or a group of users (e.g. users with different access permissions). They keep track of the instance milestones, such as baselines, major and minor versions, etc. The default labelling applied in this context is always a timestamp and an owner of the change, which already answers the who and when questions of data lineage. Apart from the central role which is to keep track of changes of any particular element or an instance of a concerned application or a model, labelling facilitates process control in such a way that version inclusions, linking and merging can easily answer the questions such as: what is the difference from the previous search results, or how did this webpage change since the last visit.
- Attributes - Attributes (name/value pairs) can be applied to elements, branches, and different versions. Attributes can keep track of different metrics and attributes of the content (such as dimensions of a picture or lines of text) so that AE can reason about and adapt accordingly. Attributes can also be used to define groups of versions involved in the same versioning task.

- Hyperlinks - Versioned objects can be also connected with hyperlinks. These links may serve for a "conventional" cross-version navigation or tracing changes in the concerned hyperspace. The idea of handling everything by links comes from the 'Xanadu' system [99], where every document can essentially be linked to any other document and every new document version would be quoting (or as stated by T. Nelson "transcluding") the original version without copying it. As well as in a common source control systems these links may be of the following types (figure 7.3): - *"loose"* links connect two instances, irrespectively of their versions, these might be links between different types of objects or nodes. These links are not changed with the version history and are retained unless one of the elements is deleted or moved; - *"tigh"* links connect particular versions of two elements. These links are "inherited" from version to version as the elements evolve until the link itself or the element (and its subsequent versions) is moved or deleted.

We dont foresee any difference in adapting versioned or inter-versioned hyperlinks and conventional hypermedia links. Adaptive navigation support should be the same.

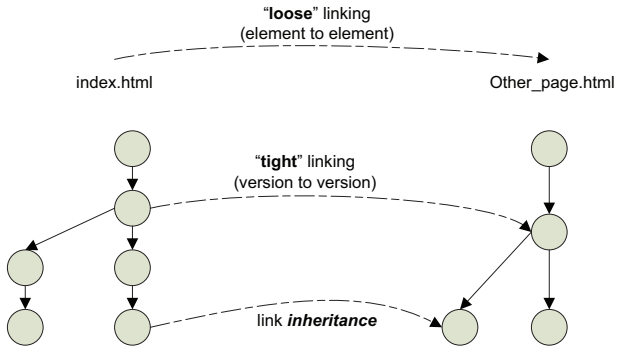


Figure 7.3: Hyperlinks in Versioned Environment

Accessing versioned data

In addition version control systems provide the following functionality to administrate access (and to some extent regulate information overload). These are access control operations that can be treated in the context of AH as follows:

- Individual and shared access to a certain information instances (e.g. certain concepts or web pages can be accessed only be a certain groups of users);

- Access should require authorization procedures and can have regulated (read/write/etc.) permission, by triggers (which can be both pre and post operational) (e.g. such as used in source control to check whether the naming convention for the label is consistent or to automatically run a script to apply a personalized view or a template to

accessed information based on the user profile) or locks (e.g. such as locking parts of the domain knowledge for students without sufficient knowledge in e-Learning applications);

- Local or shared data repositories facilitate collaboration efforts as well as access speed and privacy options.

Properties of versioned elements, retrieving and Analyzing versioned information

Aggregated information of a particular element in a version control system usually is

- label/baseline/tag these are the basic attributes that can be applied to a concerned version in order to keep track of, comprise complex structures (e.g. baselines consisting of multiple documents), annotate and retrieve data from the versioned repository according to particular label or tag;

- timestamps are the basic properties of the element instance and retains historical properties of the element. As described in [71] timestamps are the basis for history reconstruction and can help in temporal page analysis;

- user/owner the information about a version creator, owner and group. It helps to extract data according to UM and serves as a personalization property.

In general conventional version control systems may exploit the following metadata: time, date, owner, user, last change (by whom), label, tag, configuration of the version, path, personalized view(s), branches, version, annotation, directory, attributes, links, merge arrows (which contribute to information provenance), triggers.

Meta-Data Annotations

To support the information annotation can be explicitly done by the users. In general hyperlinks and merge arrows are also meta-data as well as attribute-value pairs and labeled data. Based on the meta-data used in the system, information instances, users and content fragments can be processed and analyzed using Data Mining techniques, such as aggregating or clustering similar types of changes (similarly annotated pages), classifying them (group for a particular user group or time). Apart from using conventional retrieval and querying approaches in Version control systems and multi version file systems as a high level approach we can propose OLAP (multi-dimensional analytical queries): first to construct an OLAP cube which will aggregate specific dimension (tag, label, other meta description) of the versioned base and second, to retrieve and analyze (since each element can be summarized using the hierarchy of versions) versioned data 'slicing' and 'dicing' data for a particular user or user group, set of documents or a concerned change in the system. This may considerably decrease information overload, both in terms of extraction and analysis done by the system and user handling and more intuitive presentation.

7.2.3 Versioning Changes to Support Personalization in a Dynamic Web Environment

Considering the basic concepts of the versioning we can come up with the following classes of operations which will reflect typological and structural types of changes and capture changes in the dynamic web environment. The following taxonomy of changes was mentioned in [74] in application to ontology evolution, and extended in the field of AH [83]. Here we extend and elaborate it in terms of web dynamics, describing properties and potentials of versioning operation which could be used mostly as a backend solution to store, keep track of changes and retrieve them for a further analysis:
Transformation is a set of actual changes that take place over the evolving structures, content, properties, and information instances in general. For instance these changes can be found at the DOM level of a web page (e.g. in "DiffIE" [127] changes are identified comparing hash trees of the pages). These changes include: addition, change, deletion and movement. *Conceptual changes* include concept and concept relationship changes. These modifications refer to a conceptual and more abstract (than real transformation) changes of a structure, relationships (including relations between constructs of two versions of a model or a system), or presentation aspects. This type of changes includes changes of types, relations, conceptual knowledge representation, system notions and definitions. *Descriptive changes* deal with the metadata describing the intentions, user or author credentials, and the reasoning behind the change or operation (information access, content retrieval, etc.). Descriptive changes can't contribute to the actual transformation of a data instance (e.g. change UM attributes, access to the web page), however they facilitate reasoning over multiple versions of the same instance (taking into account contextual information about the time or place of a change in order to refine a user query). Descriptive changes could be exploited to describe the information provenance, thus introducing even more elaborated changes, which can be used to overcome information reliability and currency issues, as well as to provide additional explanations to support complex reasoning. *Context changes* describe the environment in which the current update occurs and the environment where it is valid (changing a particular web page, concept in DM or a hyperlink). Context changes to some extent are the changes that drive versioning in most cases, both considering usage or user contexts. At the same time the system environment itself can be considered as a context. Context changes are usually the most space, time and effort consuming. They usually require domain experts to take part in the analysis process in order to capture a complete picture of a change.
In [115] we can see that to overcome the overload issue so-called "Milestones in Time" (aka. "Memory landmarks") are described. This is a set of public and personal time labels referring to a particular event, (in fact timestamp) according to which the information can be easily retrieved. It mostly relies on the visualization of the results of the personal content index while displaying the results of queries. The SIS ("Stuff Ive Seen") system [51] is a personal indexing and search system which provides personalized results for the aforementioned "Memory landmarks" visualization. SIS indexes full text and metadata which enables a fast way to search through the content. The result in SIS had a psychological aspect of an episodic memory and reduced the search time and in some

cases influenced whether the user would prefer to re-search information or go directly to an already defined public or personal “landmark”.

7.2.4 Overcoming Visualisation Overload Issues

Visualizing information

Adaptation techniques and methods are used to adapt presentation, content and navigation in AH systems and applications. Here we present that presentation techniques (incl. content and layout adaptation) can be used (probably with some modifications aligned with the versioning specifics) to enhance the ‘visual difference’ experience of the user. In this section we take a look at the versioning visualization concerns throughout AH research and explain how to use content and presentation adaptation techniques to support versioning visualization and tackle information overload issues. This set of adaptation techniques and methods makes up a toolkit of AH.

Overcoming Visualisation Overload: content and presentation adaptation techniques

Information presentation can be influenced in two ways: by showing and hiding or by emphasizing and deemphasizing certain information. The essential difference here is defined by the fact whether the concerned information is accessible or not. When inserting, removing, or altering fragments the presented information content is actually changed. Other techniques like dimming, sorting, zooming, and stretchtext suggest reading only the part of the available information. Mostly these are changes of the presentation. They can be also used for additional explanations. Change summarization when a textual content is analyzed statistically and linguistically and when a summary text is generated from these important sentences can be performed as well. In stretchtext only the title is shown whereas in zooming/scaling the entire information content is shown, but it may be scaled down (zoomed out) so much that it becomes unreadable. Accessing the information also causes UM updates so as to influence the adaptation in the future. Changing the presentation can be used to emphasize or deemphasize parts of the content or to suggest links to follow. There is also a presentation adaptation for different reasons though, such as device adaptation (to a device with limited capabilities, such as small screen, when the presentation can be scaled down or re-arranged to fit in the screen dimensions), accessibility adaptation or when a different preferred presentation layout or a theme is applied. Layout adaptation may also concern adaptation to a predefined presentation format, which may have a high value working with the open corpus domain where the presentation formats usually vary. One of the most popular methods in this respect is the usage of different CSS templates defining the web page layouts. A comprehensive survey of AH methods and techniques can be found in [82].

7.2.5 AH Personalization Aspects

As shown long ago in [125, 95, 51] the rate of re-visitation and re-searches on the web is very high. We prefer to re-discover and re-explore the web rather than come back us-

ing saved results. And this is where the idea of keeping, using and presenting historical changes comes into place (and this is not only saving bookmarks or using browser cache information). Views which are commonly used in source control systems can be to some extent considered as a personalization filter, delivering personalized content from the versioning system. Usually it is done manually or by means of a configuration specification set by the user to retrieve the required versions of the files (e.g. 'configspec' in ClearCase or 'client spec' in Perforce system). So in every particular case the view becomes a personalized representation of the file system (both in time (different versions) and file space (selecting only required files and leaving others out of the scope)). As a result we see that these so-called views can be enhanced with an AH approach: first, by providing means to select versions automatically based on the state of UM, and secondly, by presenting content and corresponding changes by means of AH techniques (see figure 7.4).

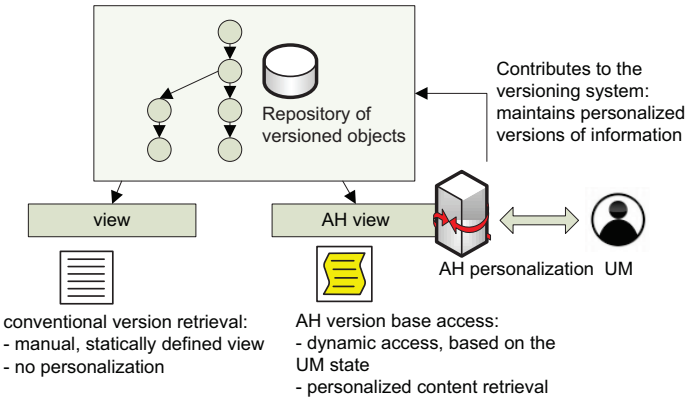


Figure 7.4: Conventional and AH personalized information access in Version Control

In the web environment this can be compared to a proxy server which processes, compares, retrieves, adapts and then presents versioned information to the user, providing an opportunity to see what other related web search queries and results were acquired, how these results relate to those of other users (e.g. as concurrent versions) and what pieces of the information can be re-used. This discussion relates to the (re-)visitation and (re-)search problems described in [1, 127, 126]. An AH component (by presenting a personalized view to the user) contributes to creating a new personalized version of a document or a content presentation of a particular concept in case of a fine grained domain structured system. Thus AH creates (authors) and supports versioned structure. The comparison of personalized visitation paths (or in general patterns) can be used for instance in recommender systems to analyze and relate user visitations and searches and for example do collaborative recommendations by matching individuals to a group, hence reducing information overload by filtering out only relevant users and corresponding results.
Time distribution of a web page visitations and search results is the most logical. Every browser has an option to provide historical information (so called 'History') of visitations

and very often recent searches to the user on request. Today people get the same results, independent of a current session, search history, etc. Here are these aspects (figure 7.5): type of the information (content and behaviour), timeframe, who and where (local vs. server), using and building *user profile*.

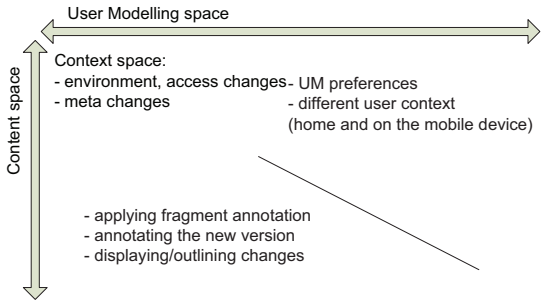


Figure 7.5: Versioning in AH chart

7.2.6 A View on an adaptive proxy architecture

In this section we would like to summarize the ideas about the proxy architecture which has been schematically presented in figure 7.4.
The architecture takes advantage of the versioning and source control benefits. The major components of the architecture are:

- View replaces Presentation model of a conventional AHS. View is a similar to a source code personal configuration (the number and the order of files we retrieve from the system) thus it is a personalized presentation;
- Versioning module keeps track, retrievs, annotates and in general is capable of performing all those versioning operations described in section 7.2.2. Due to this component UM, DM, AE information is efficiently stored and handled, providing structured changes logs;
- DM, UM are the basic AH elements. Here we assume that both UM and DM are better represented as versioned trees (the same as in source control systems) retaining all the structural changes of the respective model, besides we dont exclude historical databases implementation here;
- AE apart from the conventional reasoner should include or be capable of reasoning over the versioned domains. This could be a simple access operation or retrieving a historical information according to a designated version of the user, engine, etc. (using the operations discussed in section 7.2.9) or different type of extracting information (e.g. querying historical data bases in case we use them to support historical data). As described later in the use-cases section 7.2.9 AE rules can

have several concurrent instances. Refining and storing separate versions of these rules allows us to have concurrent versions of the engine, update these versions separately, or be deployed by a number of a different domain experts. Other than that AE should be able to handle not only domain concepts and reason upon then, but versions of these concepts as well.

7.2.7 Outlining benefits of versioning and AH

Here we would like to outline the innovative aspects and benefits of AH and versioning solutions that help to overcome various information overload issues. In this section we have mainly described AH and personalization concerns, but would like to emphasize some other advantages of bridging these two technologies. They are:

Personalization as we have scrutinized versioning can facilitate personalization in two ways. On the one hand storing an retrieving information from the version control system provides more flexibility and allows to perform more elaborate reasoning according to different versions of both content structure and according presentation and UM state. On the other hand, AH techniques contribute to the version control, creating new and new personalized versions that can be used for a further analysis.

Authoring and Design Here versioning facilitates creation, re-use, reconciliation of a concurrent versions (e.g. web site map and the corresponding content). In general authors can access and see a complete tree of versions of an application or its instances, compare, branch, merge changes from multiple instances in order to create new versions of this application, prepare try-out versions or perform test runs. This helps to overcome redundant authoring and design efforts, provides system authors as well as users with flexible options to manipulate application versions and reduces the information overload. Moreover AH approach (by deducing a newly adapted instance of the document presentation) allows to create/author the versioned structure that can be re-used later for other users or for the same user to perform backwards reasoning on his progress.

Separation of concerns - Apart from the design benefits we would like to outline that versioning can be a universal solution for a separation of concerns in many application areas (Personalization, Adaptation, User Modelling etc.).

Storing and Management - versioning provides an efficient way to store changes, including post-processing to create OLAP-cubes, to create labels or annotate versions in the versioning environment which facilitates management operations such as information retrieval from the versioned repository. This also saves space in large scale systems and keeps the changes sorted historically and by the user/owner/application which makes reasoning and analysis just easier by default, not mentioning any annotations or labels that can be used to perform more elaborate analysis or back-track/roll-back operations which are immediately available. On the other hand AH produces new versions of information

instances (e.g. news, blogs, web pages) thus contributing to a version control by creating new personalized versions that can be re-used by the users with similar interests or preferences. And as aforementioned separation of concerns only facilitated storing and authoring of AHS.

Support and Maintenance - versioned changes and corresponding operations (e.g. resolve, commit, merge, etc.) are enough to maintain and reconcile applications structures and support needs (e.g. the hierarchy used to log changes in two different applications can be easily reconciled for a single user and a common log can be produced for further analysis). This analysis contributes to a collaborative or group UM that can be later used in Recommender systems to filter out not interesting or irrelevant content.

System Logging and Analysis - Versioning logging facilities provide incremental logging according to the type of change happening in the system (transformations, descriptive change, conceptual or contextual change). This helps to facilitate provenance of every particular change, provides flexible playback/roll-back possibilities, and serves as a background structure for system analysis (e.g. logging user profile updates provides a ground for users' patterns comparison in order to provide suggestions or recommendations). Moreover user behaviour can be inferred analyzing personalized versions of the documents thus contributing to UM.

Transparency - Transparency is defined in such a way that at any given moment a user can see or retrieve (or depending on his privileges access and modify) any versioned change and at the same time retrieve the initial state of the document. Besides that user may retrieve personalized versions of the document based on their UM thus reducing information overload and being presented only relevant information.

Copyright issues - As mentioned by T. Nelson in [100] and some other of his works, the copyright issue can be solved by inducing transclusions.

7.2.8 Versioning Systems in Personalization: practical examples and AH use-cases

Versioning systems in the wild

Here we present a number of examples illustrating the idea of versioning and personalization that benefited from bringing these two technologies to an advantage in overcoming the information overload. These examples help to understand the introduced adaptive versioning proxy (section 7.2.9) better. Some of these systems implicitly implement versioning support, some will clearly benefit from having one. So let's have a look at these systems and applications.

- Xanadu [99] is a very good example of keeping and accessing multiple versions of the document. Essentially *Xanadu* doesn't change web pages, it creates a new

version of a page every time it changes (or is edited) and then links these pages. It allows easy management of many simultaneous versions (e.g. two versions can be compared side by side, versions support alternative views of the same document, etc.). Besides it implements *transclusions* providing 'recognisability' between contents of the documents and the versions showing origins of a particular information. It also touches upon copyright issues [100] and proposes the solution by making any content re-usable through versioning. The *Transclusion* concept was implemented in the *Pyxi* system and is shown in figure 7.6.

- In the e-Learning domain the course content, structure and navigation is adapted according to the student knowledge, competencies and preferences, so that an average student will not get too little or too much information designated for the course and is less likely to get lost or misunderstand the course information [20].

- Recommender systems may employ the same techniques for visual relevance ranking to relieve the user from manual filtering and organizing the results relevant to his or her profile (e.g. sorting search result snippets or annotate them with the additional data as a timestamp or collaborative results of a similar user interests) (partially done in [120]).

- An adaptive museum tour guide system based on the user preferences exploits UM and provides recommendations of the interesting museum items to follow. The system personalizes the structure of the tour as well as performs the device adaptation for those users who follow the tour on the handheld device [137].

- The SIS ("Stuff I've Seen") system [51] uses personal indexing and search which provides personalization. The 'SIS' system uses aforementioned 'memory landmarks' [115] to provide vizualisation of historical data and aligns it to certain time stamps that may correspond to memorable events, holidays or some personal time labels. Then indexes full text and metadata that facilitates the search through the user content.

- *Diff-IE* is a prototype Internet Explorer add-on. It implements changes highlight since the last page visitation and enables the comparison and view of the cached and the current page. As shown in [127] by comparing hashes of the evolving DOM structures of the web pages helps to identify and present the version changes to the user.

- OHP-Version [58] helps to reconcile versioning and context in Open Hypermedia (OH) structures. It helps to store structures according to the viewer's perspective.

- WebDAV (Web Distributed, Authoring and Versioning) [138] extends the HTTP protocol to provide version control operations over the documents on the web with the possibility to retrieve, delete, copy, move and lock respective versions.

- The "iProxy" system proposed a non-adaptive way to deal with the changing web and manually store personal web results [113]. Though this system emerged from

the field of web archiving [94] it has designed a sort of a personalization proxy that helped to overcome the issues of constantly changing web content by saving interesting (favourited) pages in a repository.

- Google *BigTable* [38, 39] is a distributed storage system for a structured data. On a very large scale it provides an effective and successful solution for many Google products (such as web indexing, Google Finance, Google Earth and others). Each cell in a 'BigTable' contains multiple versions of the same data indexed by the timestamp which provides a dynamic control over multiple versions and facilitates cache storing and access.

- "NewsJunkie" [56] represents news as a stream of information with evolving events. The main questions here are how to personalize news items using information novelty based on the time changes that these news is based upon. This involves identifying clusters of related news articles, characterizing what a user knows about the event and finally computing the novelty of new articles relative to this background.

7.2.9 Versioning Use-cases in AH

In this section we present a number of diverse use-cases showing the applicability of the proposed 'AH versioning' concept. We start with a generic example of web-pages versioning and retrieving personalized difference of the page, then continue with the recommender system use-case and in the end present an AE and rules refinement process in the 'versioned' environment.

Adaptive Hypermedia versioning proxy

In figure 7.7 we present a use-case bridging versioning approaches used to track user activities in the dynamic web environment with the AH presentation of the versioned content (e.g. user search results history, changes in a web page, etc.).
There is a user searching and browsing through the web. The initial state of the user profile (UM ver.1) starts accumulating search and visitation history, hierarchically structuring UM versioned instances according to the descriptive changes and the context changes happening with the user. After a few searches there is already enough information gathered to process. The user continues the interaction and posts a new query. The *proxy* retrieves the previous state of UM and information such as search queries and corresponding result lists with (descriptive, context, conceptual (if any available)) information about UM changes. These changes are processed and compared by the proxy and then presented to the user, providing an opportunity to see what other related queries and results he has already performed and received. As a result actual changes of a particular page (chosen by the user from his search results) can be retrieved and presented using AH presentation techniques (see section 2.3) (e.g. annotating new parts of the page with older descriptions or showing both versions on one page and dimming the old content). Search results could be also presented using AH techniques and re-arranged according to a new global

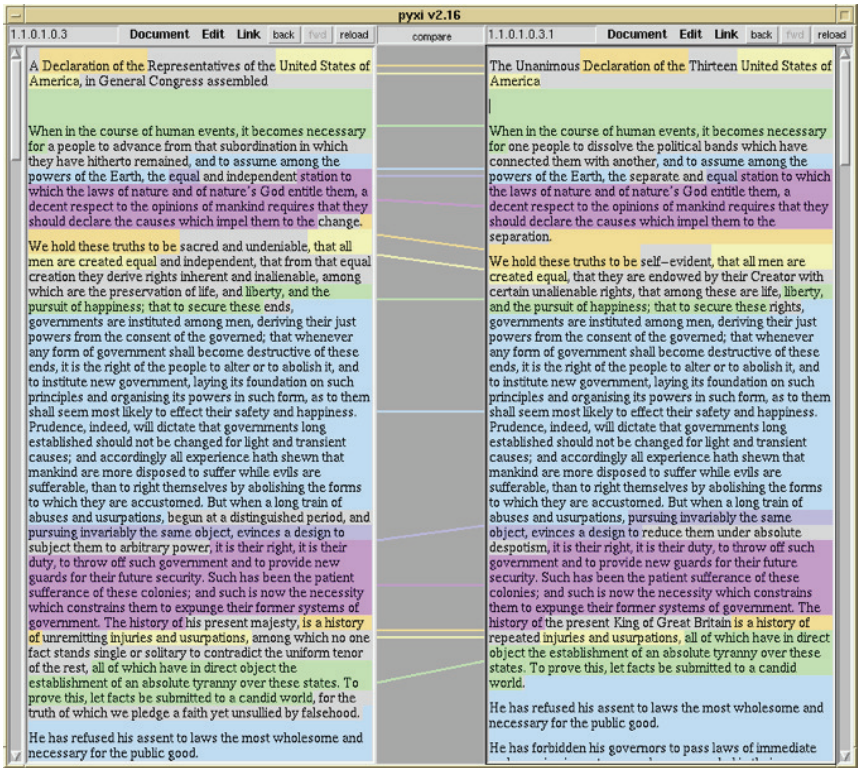


Figure 7.6: 'Transclusions' in Pyxi viewer (showing the principles of Xanadu) (this picture courtesy of Theodor Holm Nelson from [99])

or personalized ranking that may change over time for various reasons (e.g. in e-learning system a certain information object has changes a status from 'to be learned' to 'mastered' or in the news search a particular event is not a news item anymore or with respect to the user profile this news item is already read (known), etc.).
Here the difference between other users' results may be presented as well considering UM versions of multiple users from the same group or a random user with a similar query. Furthermore as mentioned in [77] this comparison can be used not only to present differences but also to make recommendations and suggestions.

Recommender System use-case

In our second use-case (figure 7.8) we would like to outline the advantages of applying revision (version) control in recommender systems. Typically, comparing a UM state to some reference characteristics, and predicting the user's choice, the system evolves from the initial user profile. Each recommendation step (viewing an item, ranking it and get-

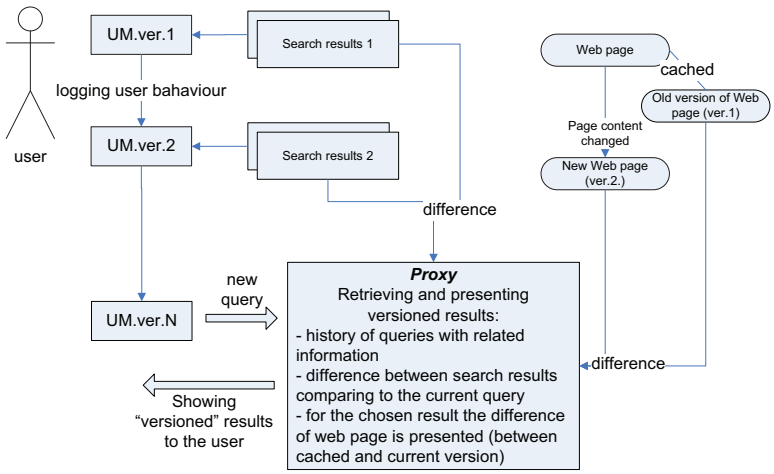


Figure 7.7: User web activities in versioned environment

ting recommendations) corresponds to a change in UM, which in a conventional system is committed to the latest state not taking into account user interactions and updates. Considering a versioning structure, we can store the difference in UM between two commits, thus logging user behaviour, and annotating it with the action (ranking) done by the user at each step. Later such a pattern of successful behaviour can serve as a recommendation to the other users providing the provenance of each recommendation step (how was this recommended and when, in what context, what was UM state at that point in time and what has influenced the recommendation). The same happens with the recommended item, which evolves and can be modified and in general have its own history of ratings. For instance a new book edition or an issue will become a separate instance in most of the RS or some custom approach will be used to relate these items to each other. Based on the assumption that a new book edition will become a branched product in the versioned system of the ratings associated with some key features (e.g. author of the book or the series) may be transferred to this new instance. Thus users will not have to rate it all over again and would be likely recommended this new book edition or a journal issue.

If the user moves from one system to another, he will face a problem of the user profile alignment, where the history of ratings that he made may help to resolve conflicts. On the other hand labelling and tagging of user and system behaviour may become a basis for the OLAP analysis. Building an OLAP cube where numeric facts (or measures), categorized by dimensions can have a set of labels or associated metadata (which is essentially corresponding to the labels used in the versioning system) will allow to organize everything in a form of a multidimensional conceptual view and apply 'slice and dice' operation as the most essential analysis tool in this respect, instead of a simple system or user logging and then applying complicated and time-consuming extraction from a plain log. This will facilitate quick extraction of rating information of a particular date or a timeframe, or a

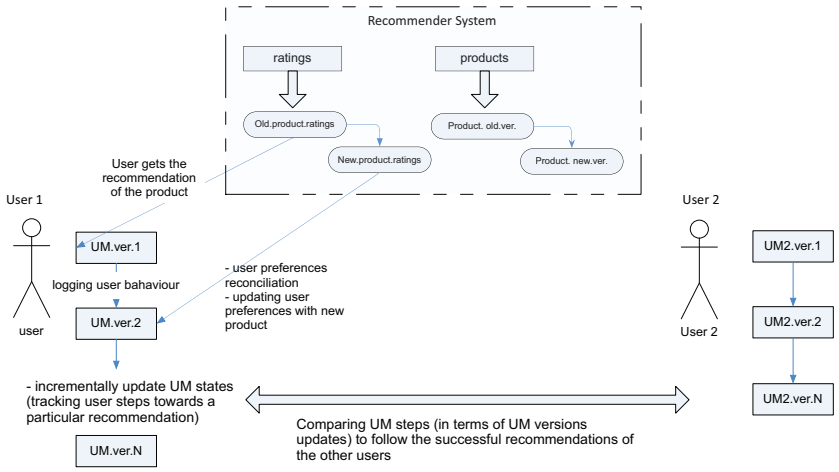


Figure 7.8: Versioning reconciliation in recommender system

set of users who rated the product a year ago. The same approach can be undertaken to version rated items in order to trace back the changes and the associated ratings (as well as the corresponding UM values) if the product or its description or properties change.

Adaptation Engine: Rule System Versioning

Let's consider now the core AHS functionality - AE (Adaptation Engine) and the adaptation rules as the main mechanism and the rules refinement process. In this example we deal with a conventional *adaptation rule* (*ECA* Event-Condition-Action type of rules) that has been changed by two domain experts or maybe automatically refined (for example using some data mining tools). This was done to modify the rule's conditions in order to change the adaptation strategy. Now there are two different rules (however in both cases we represent them using the difference (delta) data). In figure 7.9 *Rule2* has a different event E2 which for example may reflect a different response of a system (reaction on a different triggers) and has been updated by an expert after analyzing the test results or as a result of changes in the system environment. *Rule3* is refined by changing its condition from C1 to C2 which is more strict (comparing to C1) in a certain application domain (which for instance in e-Learning domain results in a better pass/knowledge rate).
Considering these premises we need to combine both types of changes in a given application domain because both (as discovered) can provide a better adaptation outcome. We have a few possibilities of doing that. If we use the same application domain and one author or an expert wants to re-use common successful results or practices shared within this domain he or she can just merge the changes to a personal branch and create a new version of the rule (or thus create a new AE version to run the experiment on the refined engine). Besides this rule merge can be done automatically. System may also evolve and

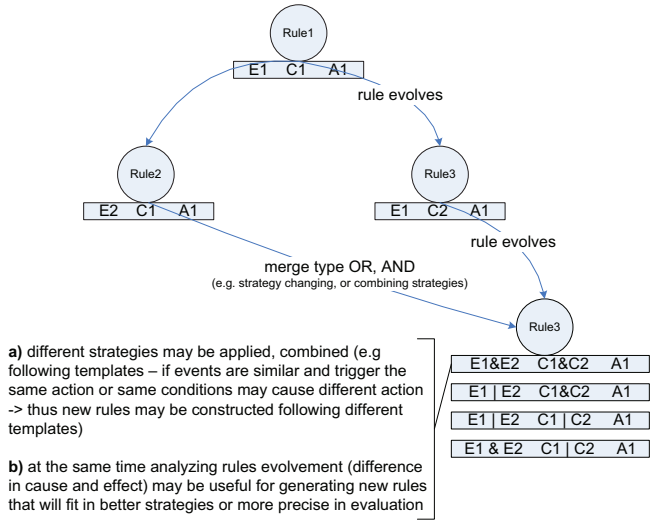


Figure 7.9: AE Rule refinement case in versioned environment

produce new rules which combine the parameters of each of the latest versions branched from the parent node. This gives a wider choice and possibilities to a domain expert to evaluate adaptive engine functionality and new (deduced) rules deployment.
In some cases refined or automatically generated rules may be applied in the system without a domain expert. For example if two different conditions and the same triggering event initiate a similar action, both rules can be automatically merged evolving in a single rule with composite condition (C1 OR C2).
Keeping each rule's tree (or in a general case - AE evolution tree) is a straightforward way to keep track of all the changes applied in the engine which as a result can be used to analyze changes, modify engine rules (both manually and automatically), help in keeping system log and AE updates deployment.
At the same time the aforementioned versioned structures help to re-calculate and propagate user's knowledge in a simple manner. A number of systems (e.g. AHA!) in case when a new child concept emerges (for instance when the domain author decides to broaden the knowledge representation on a particular topic) propagate these changes in UM (based on the average contribution of each child concept). In the versioned environment all the structural and rule changes (and moreover versioned UMs) are scrutinized which facilitates this type of the knowledge propagation.

Chapter 8

Concluding Remarks

The coming years will bring more and more use-cases of how AHS can provide adaptation, what techniques will be introduced, and what research areas will become adjacent to AH field and introduce new technologies. In this concluding chapter we summarize the results of the the *GAF* research and give some pointers for the future.

8.1 Conclusions: Revisiting the Research Question

In this dissertation we have studied the field of AHS modelling and devised a new up-to-date reference adaptation framework. We have started the research by stating seven research question in section 1.3. Hereafter we summarize our findings and give the answers.

- *Question 0: What is use of a generic framework and what is the specific use of GAF?*

  This question defined the motivation and the purpose of the *GAF* research. It has been answered in motivational section 1.1 and substantially has been proven throughout the dissertation particularly in chapters 5 and 6 where a number of system types and real systems have been studied.

- *Question 1: Is it possible to capture all up-to-date AHS functionality and developments? What would be the exemplary set of systems that can serve as a basis to capture the generalities of AH? What approaches, methods and techniques exist to support up-to-date adaptive hypermedia solutions?*

  In chapter 2 we came up with an up-to-date review of AH research for the past 15 years and the resulting requirements for a modular composition of a new AHS reference model that captured all new trends and adjoining technologies to support users within rich and diverse hyperspaces, bringing a new level of adaptation to the user experience. First and foremost we presented a survey of AH architectures, models and systems, and defined a new taxonomy of adaptation techniques. Secondly, we have obtained many requirements for a new reference model that we design and

that builds on the experience gained with existing models and that draws from the many new research ideas that show up in (prototype) adaptive models.

- *Question 2: Is it possible to bring conventional pre-authored Adaptive Hypermedia together with a data-driven approach to personalization?*

  To answer this question we touched upon the problems of data-driven approaches in adaptation and reasoning in chapter 3. Moreover in the compliance study chapter 5 we investigated a number of data-driven personalization aspects such as RecSys, SE, etc., and conducted a number of compliance studies in chapter 6 to bridge these two approaches together (conventional adaptation and data-driven approaches). We tried to breach this gap between conventional and data-driven systems and show that essentially many web-based personalization systems, application and approaches can be expressed in AH terms, which opens a whole new field to bring into AH research and vice versa.

- *Question 3: What are the framework requirements ("mandatory" and "optional" building blocks)? How the reference architecture of a generic adaptation framework would look like?*

  To answer these questions we first investigated the modelling aspects of *GAF* in chapter 3 and presented the reference architecture of *GAF* in chapter 4, where we elaborated on the structure, functionality, relationships and the necessary interfaces and corresponding layers' connections.

- *Question 4: Is it possible to devise a reference adaptation process out of the existing data models in order to clue together data representations the AH research field has been using for so long?*

  In chapter 3 we introduced the notion and the definition of the generic adaptation process *GAP* and aligned it with a layered *GAF* architectural building blocks. The Adaptation process describes interactions in AHS usually starting with the goal statement, exploiting features of the user and domain models in different contexts and adapting various aspects of the information and presentation to the user. GAP tightens the building blocks of the reference model and can be considered as the reference adaptation process or *GAP*.

- *Question 5: Does such a framework exist and how do we evaluate the framework?*

  In order to prove the framework existence and evaluate *GAF* we first introduced the definition of the framework *compliance* and presented a number of compliance use-cases in chapter 5 where the framework structure was evaluated against different types of systems (Adaptive e-Learning Course, Recommender System and Search Engine). Then in chapter 6 we described a number of a real system studies where these systems were described in terms of *GAF* identifying pros, cons and further system extensions. Evaluating the proposed general-purpose AHS architecture against RS and SE has shown that the GAF architecture is generic enough to

accommodate the description of different systems, as well as provide the flexibility in one go by building a custom system with the GAF building blocks. These studies gave a number of new challenges to investigate the applicability of different recommendation approaches, as well as new developments in adaptive information systems. And eventually we introduced the framework analysis approach in section 6.5 which in combination with the reference architecture description should be able to provide a detailed picture for each system or an application being evaluated.

- *Question 6: What are the technologies leveraging and extending the potentials of AH? How GAF can use these technologies?*

  In chapter 7 we discussed the importance of the provenance and versioning in AH and essentially any type of personalization web-based system. First, we aligned adaptation questions and also corresponding answers with the provenance model thus enriching the conventional AH modelling approach. Second, we introduced the versioning framework and investigate information overload issues from two perspectives: from personalized visualization mechanisms exploiting AH methods and techniques, to maintaining and to some extent personalizing the versioned structure by creating adaptive versions of accessed information.

8.2 Future Work

We have already started the future work discussion in chapter 7. There we have investigated a number of close to AH approaches and techniques, outlined their use and benefits, weighed pros and cons, thus giving a head start to a new system development. In future it would be interesting to study the aforementioned examples of provenance importance in depth, analyze the impact on the adaptation process and comparing results with the conventional AHS and to evaluate the utility of AH provenance in deployed systems. Considering future of the AH versioning we can think of describing layers of a generic adaptation framework in a versioning context (or essentially looking at the basics of adaptation through versioning), investigate technologies (e.g. source control, historical data bases, etc.) that meet the requirements of each and every model of the framework.
A complete implementation was not feasible within the boundaries of the *GAF* research and the research observations. In future additional effort might be put into the development of a general-purpose software system combining aforementioned developments and setting new standards in AH field.
In the future the search adaptation process sequence should be extended, and inter-layer communications in particular, emphasizing the interoperability of a new AH developments in the context of (web) search. This may require unifying search and linking methods (incl. adaptive navigation techniques) for AH field.
So far a study of existing approaches in RS was done to comply with the layered structure of adaptive information systems, which has resulted in an overlay models presented in the dissertation providing an overview of a RS functionality and a corresponding overlay of AHS layers and adaptation process. Evaluating the proposed general-purpose AHS

architecture against RS has shown that the GAF architecture is generic enough to accommodate the description of different recommenders, as well as provide the flexibility of both AH and RS in one go by building a custom system with the GAF building blocks. As a result we can foresee some further developments and research strategies in bringing recommendations to the field of AHS. Coming up with up-to-date requirements which would bootstrap AHS and RS development using heterogeneous information sources is important at this stage. The real case study of the 'HeyStaks' social RS has proven these points. It has given us new challenges to investigate the applicability of different recommendation approaches, as well as new developments in adaptive information systems which will allow to decide on the system composition on the implementation level. Thus studying in depth a particular use-cases of the 'HeyStaks' RS compliance and analyzing it in the context of adaptation process and vice versa, is helpful in further system development and adding new features to such systems.

Bibliography

[1] E. Adar, J. Teevan, S. T. Dumais, and J. L. Elsas. The web changes everything: understanding the dynamics of web content. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 282–291, New York, NY, USA, 2009. ACM.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 17(6):734–749, 2005.

[3] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender Systems Handbook*, pages 217–253. Springer, 2011.

[4] X. Amatriain, M. Torrens, P. Resnick, and M. Zanker, editors. *Proc. of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*. ACM, 2010.

[5] L. Ardissono, R. Furnari, A. Goy, G. Petrone, and M. Segnan. Context-aware workflow management. In L. Baresi, P. Fraternali, and G.-J. Houben, editors, *Web Engineering*, volume 4607 of *Lecture Notes in Computer Science*, pages 47–52. Springer Berlin, Heidelberg, 2007.

[6] L. Ardissono, A. Goy, and G. Petrone. A framework for the development of distributed, context-aware adaptive hypermedia applications. In *Proc. of the 5th international conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, AH '08, pages 259–262, Berlin, Heidelberg, 2008. Springer-Verlag.

[7] L. Aroyo, P. De Bra, and V. Chepegin. Semantic Web-based Adaptive Hypermedia. In *WWW Workshop on Application Design, Development and Implementation Issues in the Semantic Web*, volume 105 of *CEUR Workshop Proc.* CEUR-WS.org, 2004.

[8] L. Aroyo, P. De Bra, G.-J. Houben, and R. Vdovjak. Embedding information retrieval in adaptive hypermedia: Ir meets aha! *Hypermedia*, 10:53–76, June 2004.

[9] C. Bailey, W. Hall, D. E. Millard, and M. J. Weal. Adaptive hypermedia through contextualized open hypermedia structures. *ACM Trans. Inf. Syst.*, 25, October 2007.

[10] J. Bailey, A. Poulovassilis, and P. T. Wood. An event-condition-action language for xml. In *Proc. of the 11th international conference on World Wide Web*, WWW '02, pages 486–495, New York, NY, USA, 2002. ACM.

[11] M. Balik and I. Jelinek. General architecture of adaptive and adaptable information systems. In *Proc. of the Ninth International Conference on Enterprise Information Systems, ICEIS07 Doctoral Consortium*, pages 29–34. Springer, 2007.

[12] M. Balík and I. Jelínek. Towards semantic web-based adaptive hypermedia model. In *ESWC 2008 Ph.D. Symposium*, 2008.

[13] M. Belbin. *Belbin Team Roles*, pages 1–1. Butterworth Heinemann, 2004.

[14] P. Bellekens, K. Sluijs, L. Aroyo, and G.-J. Houben. Convergence of web and tv broadcast data for adaptive content access and navigation. In *Proc. of the 5th international conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, AH '08, pages 361–365, Berlin, Heidelberg, 2008. Springer-Verlag.

[15] S. Berkovsky, T. Kuflik, and F. Ricci. Mediation of user models for enhanced personalization in recommender systems. *User Modeling and User-Adapted Interaction*, 18(3):245–286, 2008.

[16] F. Bohnert, I. Zukerman, S. Berkovsky, T. Baldwin, and L. Sonenberg. Using collaborative models to adaptively predict visitor locations in museums. In *Proc. of Adaptive Hypermedia 2008*, 2008.

[17] M. Bordegoni, G. Faconti, S. Feiner, M. T. Maybury, T. Rist, S. Ruggieri, P. Trahanias, and M. Wilson. A standard reference model for intelligent multimedia presentation systems. *Comput. Stand. Interfaces*, 18(6-7):477–496, 1997.

[18] P. D. Bra, A. T. M. Aerts, G.-J. Houben, and H. Wu. Making general-purpose adaptive hypermedia work. In G. Davies and C. B. Owen, editors, *WebNet*, pages 117–123. AACE, 2000.

[19] P. D. Bra, M. Pechenizkiy, K. V. D. Sluijs, and D. Smits. Grapple: Integrating adaptive learning into learning management systems. *Engineering Education*, 2008.

[20] P. D. Bra, D. Smits, E. Knutov, E. Ploum, and K. van der Sluijs. Unifying adaptive learning environments: authoring styles in the grapple project. In *Adjunct Proceedings First International Conference on User Modeling, Adaptation, and Personalization (UMAP'09)*, pages 121–126, 2009.

[21] P. D. Bra, D. Smits, and N. Stash. Creating and delivering adaptive courses with aha! In *EC-TEL*, volume 4227 of *Lecture Notes in Computer Science*, pages 21–33. Springer, 2006.

[22] P. Brusilovsky. Methods and Techniques of Adaptive Hypermedia. *User Modeling and User Adapted Interaction*, 6(1-2):87–129, 1996.

[23] P. Brusilovsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11:87–110, March 2001.

[24] P. Brusilovsky. Adaptive navigation support. In Brusilovsky et al. [29], pages 263–290.

[25] P. Brusilovsky. Adaptive navigation support for open corpus hypermedia systems. In Nejdl et al. [98], pages 6–8.

[26] P. Brusilovsky, I. Cantador, Y. Koren, T. Kuflik, and M. Weimer. Workshop on information heterogeneity and fusion in recommender systems hetrec 2010. In Amatriain et al. [4], pages 375–376.

[27] P. Brusilovsky, J. Eklund, and E. Schwarz. Web-based education for all: a tool for development adaptive courseware. *Comput. Netw. ISDN Syst.*, 30:291–300, April 1998.

[28] P. Brusilovsky and N. Henze. Open corpus adaptive educational hypermedia. In Brusilovsky et al. [29], pages 671–696.

[29] P. Brusilovsky, A. Kobsa, and W. Nejdl, editors. *The Adaptive Web, Methods and Strategies of Web Personalization*. Springer, 2007.

[30] P. Brusilovsky and E. Millán. User Models for Adaptive Hypermedia and Adaptive Educational Systems. In A. K. P. Brusilovsky and W. Nejdl, editors, *The Adaptive Web*, Lecture Notes in Computer Science, pages 3–53. Springer Berlin / Heidelberg, 2007.

[31] P. Brusilovsky, E. Schwarz, and G. Weber. Elm-art: An intelligent tutoring system on world wide web. In C. Frasson, G. Gauthier, and A. Lesgold, editors, *Intelligent Tutoring Systems*, volume 1086 of *Lecture Notes in Computer Science*, pages 261–269. Springer Berlin, Heidelberg, 1996.

[32] I. Cantador, A. Bellogín, and P. Castells. A multilayer ontology-based hybrid recommendation model. *AI Commun.*, 21(2-3):203–210, 2008.

[33] F. Carmagnola, F. Cena, C. Gena, and I. Torre. A semantic framework for adaptive web-based systems, 2005.

[34] R. M. Carro, A. Ortigosa, and J. Schlichter. Adaptive collaborative web-based courses. In *Proc. of the 2003 international conference on Web engineering*, ICWE'03, pages 130–133, Berlin, Heidelberg, 2003. Springer-Verlag.

[35] R. M. Carro, E. Pulido, and P. Rodríguez. Tangow: Task-based adaptive learner guidance on the web. In *Proc. of the Second Workshop on Adaptive Systems and User Modeling on the Web*, pages 49–57, 1999.

[36] S. Casteleyn. *Designer specified self re-organizing websites*. PhD thesis, Vrije Universiteit Brusselsy, 2005.

[37] D. Challis. Committing to quality learning through adaptive online assessment. *Assessment & Evaluation in Higher Education*, 30(5):519–527, 2005.

[38] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: a distributed storage system for structured data. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7*, OSDI '06, pages 15–15, Berkeley, CA, USA, 2006. USENIX Association.

[39] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, 26:4:1–4:26, June 2008.

[40] Z. Chedrawy, S. Sibte, and R. Abidi. An adaptive personalized recommendation strategy featuring context sensitive content adaptation. In *AH 2006. LNCS*, pages 61–70. Springer, 2006.

[41] O. Conlan, V. Wade, C. Bruen, and M. Gargan. Multi-model, metadata driven approach to adaptive hypermedia services for personalized elearning. In *Proc. of the Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, AH '02, pages 100–111, London, UK, UK, 2002. Springer-Verlag.

[42] A. Cristea and M. Verschoor. The lag grammar for authoring the adaptive web. In *Proc. of the International Conference on Information Technology: Coding and Computing ITCC'04, vol. 2*, ITCC '04, pages 382–386, Washington, DC, USA, 2004. IEEE Computer Society.

[43] A. I. Cristea and L. Calvi. The three layers of adaptation granularity. In P. Brusilovsky, A. T. Corbett, and F. de Rosis, editors, *User Modeling*, volume 2702 of *Lecture Notes in Computer Science*, pages 4–14. Springer, 2003.

[44] A. I. Cristea and A. de Mooij. Laos: Layered www ahs authoring model with algebraic operators. In *WWW Alternate Paper Tracks*, 2003.

[45] P. De Bra and L. Calvi. AHA! an open adaptive hypermedia architecture. *The New Review of Hypermedia and Multimedia*, 4:115–140, 1998.

[46] P. De Bra, G.-J. Houben, and Y. Kornatzky. An Extensible Data Model for Hyperdocuments. In *ECHT 92, Proc. of the 4th ACM Conference on Hypertext*, pages 222–231. ACM, 1992.

[47] P. De Bra, G.-J. Houben, and H. Wu. AHAM: a Dexter-Based Reference Model for Adaptive Hypermedia. In *Hypertext*, pages 147–156. ACM, 1999.

[48] P. De Bra, D. Smits, and N. Stash. The design of AHA! In *Hypertext'06: Proc. of the 17th ACM Conference on Hypertext and Hypermedia,*, pages 133–134, New York, NY, USA, 2006. ACM.

[49] P. T. de Vrieze. *Fundaments of Adaptive Personalization*. PhD thesis, Radboud University Nijmegen, Nijmegen, The Netherlands, 2006.

[50] S. Dietze, A. Gugliotta, and J. Domingue. A semantic web service oriented framework for adaptive learning environments. In E. Franconi, M. Kifer, and W. May, editors, *The Semantic Web: Research and Applications*, volume 4519 of *Lecture Notes in Computer Science*, pages 701–715. Springer Berlin, Heidelberg, 2007.

[51] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff i've seen: A system for personal information retrieval and re-use. In *SIGIR '03*, pages 72–79. ACM Press, 2003.

[52] M. Eirinaki and M. Vazirgiannis. Web mining for web personalization. *ACM Trans. Internet Technol.*, 3:1–27, February 2003.

[53] S. G. Esparza, M. P. O'Mahony, and B. Smyth. On the real-time web as a source of recommendation knowledge. In Amatriain et al. [4], pages 305–308.

[54] R. Farzan and P. Brusilovsky. P.: Social navigation support in a course recommendation system. In *In proceedings of 4th International Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, pages 91–100. Springer Verlag, 2006.

[55] P. Funk and N. Xiong. Case based reasoning and knowledge discovery in medical applications with time series. *Computational Intelligence*, 22(3-4):238–253, November 2006.

[56] E. Gabrilovich, S. Dumais, and E. Horvitz. Newsjunkie: providing personalized newsfeeds via analysis of information novelty. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 482–490, New York, NY, USA, 2004. ACM.

[57] M. Gorle, J. Ohene-Djan, C. Bailey, G. Wills, and H. Davis. Is it possible to devise a unifying model of adaptive hypermedia and is one necessary? In *Proc. of the Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 167–183. New York, ACM, 2003.

[58] J. Griffiths, D. E. Millard, H. Davis, D. T. Michaelides, and M. J. Weal. Reconciling versioning and context in hypermedia structure servers. In *Proceedings of the 2002 international conference on Metainformatics*, MIS'02, pages 118–131, Berlin, Heidelberg, 2003. Springer-Verlag.

[59] P. Groth, E. Deelman, G. Juve, G. Mehta, and B. Berriman. Pipeline-centric Provenance Model. In *Proc. of WORKS '09: the 4th Workshop on Workflows in Support of Large-Scale Science*, pages 1–8, New York, NY, USA, 2009. ACM.

[60] F. Halasz and M. Schwartz. The Dexter Hypertext Reference Model. In *Proc. of the Hypertext Standardization Workshop by (NIST)*. NIST, 1990.

[61] F. Halasz and M. Schwartz. The dexter hypertext reference model. *Commun. ACM*, 37(2):30–39, 1994.

[62] F. Hanisch, M. Muckenhaupt, F. J. Kurfess, and W. Straer. The ahes taxonomy: Extending adaptive hypermedia to software components. In *AH'06*, pages 342–345, 2006.

[63] J. Hannon, M. Bennett, and B. Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. In Amatriain et al. [4], pages 199–206.

[64] J. Hannon, E. Knutov, P. De Bra, M. Pechenizkiy, B. Smyth, and K. McCarthy. Bridging recommendation and adaptation: Generic adaptation framework - Twittomender compliance study. In *Proc. of DAH'11: Int. Workshop Dynamic and adaptive hypertext: generic frameworks, approaches and techniques*, pages 1–9, 2011.

[65] M. Hendrix and A. I. Cristea. Reuse patterns in adaptation languages: Creating a meta-level for the lag adaptation language. In Nejdl et al. [98], pages 304–307.

[66] N. Henze. *Adaptive Hyperbooks: Adaptation for Project-Based Learning Resources*. PhD thesis, Univerität Hannover, Hannover, Germany, 2000.

[67] N. Henze and W. Nejdl. A logical characterization of adaptive educational hypermedia. *Hypermedia*, 10:77–113, June 2004.

[68] C. Hockemeyer. Rath a relational adaptive tutoring hypertext www environment based on knowledge space theory. In *Chalmers University of Technology*, pages 417–423, 1998.

[69] K. Hua, B. Fairings, and I. Smith. Cadre: case-based geometric design. *Artificial Intelligence in Engineering*, 10(2):171–183, 1996.

[70] F. James and B. Rajendra. Earth system science workbench: A data management infrastructure for earth science products. In *Proc. of SSDBM'01: the 13th Int. Conference on Scientific and Statistical Database Management*, pages 180–189, Washington, DC, USA, 2001. IEEE CS.

[71] A. Jatowt and K. Tanaka. Towards mining past content of web pages. *New Rev. Hypermedia Multimedia*, 13:77–86, January 2007.

[72] J. Kay. Scrutable Adaptation: Because We Can and Must. In *Proc. of AH 2006: 4th Int. Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 11–19, Berlin-Heidelberg, 2006. Springer.

[73] R. Kazman, M. Klein, and P. Clements. ATAM: Method for architecture evaluation. Technical Report ESC-TR-2000-004, Carnegie Mellon, Software Engineering Institute, 2000.

[74] M. Klein, D. Fensel, A. Kiryakov, and D. Ognyanov. Ontology versioning and change detection on the web. In A. Gmez-Prez and V. Benjamins, editors, *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, volume 2473 of *Lecture Notes in Computer Science*, pages 247–259. Springer, Heidelberg, 2002.

[75] E. Knutov. Gaf: Generic adaptation framework. In Nejdl et al. [98], pages 400–404.

[76] E. Knutov. GAF: Generic Adaptation Framework, 2009.

[77] E. Knutov, P. D. Bra, and M. Pechenizkiy. Versioning in Adaptive Hypermedia. volume 473, pages 61–71, Torino, Italy, June 2009. CEUR-WS.org.

[78] E. Knutov, P. D. Bra, and M. Pechenizkiy. Adaptation and search: from dexter and aham to gaf. In M. H. Chignell and E. Toms, editors, *HT*, pages 281–282. ACM, 2010.

[79] E. Knutov, P. D. Bra, and M. Pechenizkiy. Adaptive Hypermedia Systems Analysis Approach by Means of the GAF Framework. volume 823, pages 41–46, Eindhoven, The Netherlands, June 2011. CEUR-WS.org.

[80] E. Knutov, P. D. Bra, and M. Pechenizkiy. Generic adaptation framework: a process-oriented perspective. *J. Digit. Inf.*, 12(1), 2011.

[81] E. Knutov, P. D. Bra, D. Smits, and M. Pechenizkiy. Bridging navigation, search and adaptation - adaptive hypermedia models evolution. In J. Cordeiro and J. Filipe, editors, *WEBIST*, pages 314–321. SciTePress, 2011.

[82] E. Knutov, P. De Bra, and M. Pechenizkiy. AH 12 years later: a comprehensive survey of adaptive hypermedia methods and techniques. *New Rev. Hypermedia Multimedia*, 15(1):5–38, 2009.

[83] E. Knutov, P. De Bra, and M. Pechenizkiy. Versioning in Adaptive Hypermedia. In *Proc. of DAH'09: Int. Workshop Dynamic and adaptive hypertext: generic frameworks, approaches and techniques*, pages 61–71, Aachen, 2009. CEUR-WS.org.

[84] E. Knutov, P. De Bra, and M. Pechenizkiy. Bridging Versioning and Adaptive Hypermedia in the Dynamic Web. In *Adjunct Proc. of 18th International Conference on User Modeling, Adaptation, and Personalization*, pages 13–15, 2010.

[85] E. Knutov, P. De Bra, and M. Pechenizkiy. Provenance Meets Adaptive Hypermedia. In *Hypertext '10: Proc. of the 21th ACM Conference on Hypertext and Hypermedia*, pages 93–98, New York, NY, USA, 2010. ACM.

[86] A. Kobsa. Generic user modeling systems. *User Modeling and User-Adapted Interaction*, 11:49–63, March 2001.

[87] N. Koch. *Software engineering for adaptive hypermedia systems*. PhD thesis, Ludwig-Maximilians University of Munich, Munich, Germany, 2001.

[88] N. Koch and M. Wirsing. The Munich Reference Model for Adaptive Hypermedia Applications. In *Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, AH 2002, Proceeding*, volume 2347 of *Lecture Notes in Computer Science*, pages 213–222. Springer, 2002.

[89] Y. Koren and R. M. Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. Springer, 2011.

[90] O. Lassila and R. Swick. Resource description framework RDF model and syntax specification. `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/`, 1999.

[91] H. B. Leon, H. Rego, T. Moreira, and F. G. Penalvo. A model for online assessment in adaptive e-learning platform. In *Recent Research Developments In Learning Technologies*, pages 1–5, Badajoz, Spain, 2005. FORMATEX.

[92] T. Lutkenhouse, M. L. Nelson, and J. Bollen. Distributed, real-time computation of community preferences. In *Proc. of the sixteenth ACM conference on Hypertext and hypermedia*, HYPERTEXT '05, pages 88–97, New York, NY, USA, 2005. ACM.

[93] E. Martìn, R. Carro, and P. Rodrguez. Comole: A context-based adaptive hypermedia system for m-learnin. In *Procs. of l VIII Simposio Nacional de Tecnologas de la Informacin y las Comunicaciones en la Educaciòn SINTICE 07*, pages 79–86. Thomson, 2007.

[94] J. Masans and J. Masans. Web archiving, issues and methods. In *Web Archiving*, pages 1–53. Springer, Heidelberg, 2006.

[95] B. McKenzie and A. Cockburn. An empirical analysis of web page revisitation. In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 5 - Volume 5*, HICSS '01, page 5019, Washington, DC, USA, 2001. IEEE Computer Society.

[96] L. Mei and S. Easterbrook. Evaluating user-centric adaptation with goal models. In *Proc. of the 1st International Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments*, SEPCASE '07, page 6, Washington, DC, USA, 2007. IEEE Computer Society.

[97] T. Morita, N. Fukuta, N. Izumi, and T. Yamaguchi. Doddle-owl: A domain ontology construction tool with owl. In R. Mizoguchi, Z. Shi, and F. Giunchiglia, editors, *The Semantic Web ASWC 2006*, volume 4185 of *Lecture Notes in Computer Science*, pages 537–551. Springer Berlin, Heidelberg, 2006.

[98] W. Nejdl, J. Kay, P. Pu, and E. Herder, editors. *Adaptive Hypermedia and Adaptive Web-Based Systems, 5th International Conference, AH 2008*, volume 5149 of *Lecture Notes in Computer Science*. Springer, 2008.

[99] T. H. Nelson. The unfinished revolution and xanadu. *ACM Comput. Surv.*, 31, December 1999.

[100] T. H. Nelson. Zigzag (tech briefing). In *Proceedings of the 12th ACM conference on Hypertext and Hypermedia*, HYPERTEXT '01, pages 261–262, New York, NY, USA, 2001. ACM.

[101] none. web-frontend of the mars framework. `http://www.semwebtech.org/mars/frontend/`, 2001.

[102] A. O'Connor and V. Wade. Informing context to support adaptive services. In V. Wade, H. Ashman, and B. Smyth, editors, *Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 4018 of *Lecture Notes in Computer Science*, pages 366–369. Springer Berlin, Heidelberg, 2006.

[103] J. Ohene-Djan. *A formal approach to personalisable adaptive hyperlink-based interaction*. PhD thesis, University of London, London, UK, 2000.

[104] J. Ohene-Djan and A. Fernandes. Modelling personalisable hypermedia: The goldsmiths model. *New Review of Hypermedia and Multimedia*, 8(1):99–137, August 2002.

[105] C. Pancerella and et al. Metadata in the collaboratory for multi-scale chemical science. In *Proc. of DCMI'03: the 2003 Int. Conference on Dublin Core and Metadata Applications*, pages 1–9. Dublin Core Metadata Initiative, 2003.

[106] A. Paramythis, S. Weibelzahl, and J. Masthoff. Layered evaluation of interactive adaptive systems: framework and formative methods. *User Modeling and User-Adapted Interaction*, 20:383–453, December 2010.

[107] C. Paris, M. Wu, K. V. Linden, M. Post, and S. Lu. Myriad: An architecture for contextualized information retrieval and delivery. In *In AH2004: International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 205–214, 2004.

[108] O. Phelan, K. McCarthy, and B. Smyth. Buzzer - online real-time topical news article and source recommender. In L. Coyle and J. Freyne, editors, *AICS*, volume 6206 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2009.

[109] R. Priedhorsky, J. Chen, S. K. Lam, K. Panciera, L. Terveen, and J. Riedl. Creating, destroying, and restoring value in wikipedia, 2007.

[110] L. Purvis. *Intelligent Design Problem Solving Using Case-Based and Constrain-Based Techniques*. PhD thesis, University of Connecticut, Connecticut, USA, 1995.

[111] S. Ram. Data Provenance. Project accomplishments, The University of Arizona, Advanced Database Research Group, The University of Arizona, Mar 2006.

[112] S. Ram and J. Liu. Understanding the Semantics of Data Provenance to Support Active Conceptual Modeling. In *Proc. of ACM-L: 1st Int. Workshop on Active Conceptual Modeling of Learning*, pages 17–29, Berlin-Heidelberg, 2006. Springer.

[113] H. C.-H. Rao, Y.-F. Chen, and M.-F. Chen. A proxy-based personal web archiving service. *SIGOPS Oper. Syst. Rev.*, 35:61–72, January 2001.

[114] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[115] M. Ringel, E. Cutrell, S. Dumais, and E. Horvitz. Milestones in time: The value of landmarks in retrieving information from personal stores. In *In Proceedings of INTERACT 2003*, pages 184–191, 2003.

[116] C. Romero, S. Ventura, A. Zafra, and P. De Bra. Applying web usage mining for personalizing hyperlinks in web-based adaptive educational systems. *Computers & Education*, 53(3):828–840, 2009.

[117] L. Rutledge, N. Stash, Y. Wang, and L. Aroyo. Accuracy in rating and recommending item features. In *Proc. of the 5th international conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, AH '08, pages 163–172, Berlin, Heidelberg, 2008. Springer-Verlag.

[118] M. Schneider-Hufschmidt, T. Ktihme, and U. Malinowski, editors. *State of the Art in Adaptive User Interfaces*, pages 13–48. North Holland, Amsterdam, the Netherlands, 1993.

[119] P. Seefelder de Assis and D. Schwabe. A general Meta-Model for Adaptive Hypermedia Systems. In P. D. Bra and W. Nejdl, editors, *Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 433–436. Springer Berlin / Heidelberg, 2004.

[120] B. Smyth, P. Briggs, M. Coyle, and M. O'Mahony. Google shared. a case-study in social search. In G.-J. Houben, G. I. McCalla, F. Pianesi, and M. Zancanaro, editors, *UMAP*, pages 283–294. Springer, 2009.

[121] M. Specht, M. Kravcik, R. Klemke, L. Pesin, and R. Hottenhain. Adaptive learning environment for teaching and learning in winds. In P. De Bra, P. Brusilovsky, and R. Conejo, editors, *Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 2347 of *Lecture Notes in Computer Science*, pages 572–575. Springer Berlin, Heidelberg, 2006.

[122] N. Stash. *Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System*. PhD thesis, Eindhoven University of Technology, 2007.

[123] N. Stash, L. Aroyo, Y. Wang, L. Rutledge, and P. Gorgels. Semantics-Driven Recommendations In Cross-Media Museum Applications. In *Proc. of the Workshop on Personalized Access to Cultural Heritage PATCH, 2008*, 2008.

[124] S. Stober and A. Nrnberger. Context-based navigational support in hypermedia. In V. Wade, H. Ashman, and B. Smyth, editors, *Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 4018 of *Lecture Notes in Computer Science*, pages 328–332. Springer Berlin, Heidelberg, 2006.

[125] L. Tauscher and S. Greenberg. How people revisit web pages: empirical findings and implications for the design of history systems. *Int. J. Hum.-Comput. Stud.*, 47:97–137, July 1997.

[126] J. Teevan, S. T. Dumais, and D. J. Liebling. A longitudinal study of how highlighting web content change affects people's web interactions. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 1353–1356, New York, NY, USA, 2010. ACM.

[127] J. Teevan, S. T. Dumais, D. J. Liebling, and R. L. Hughes. Changing how people view changes on the web. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, pages 237–246, New York, NY, USA, 2009. ACM.

[128] J. Teevan, D. Ramage, and M. R. Morris. Twittersearch: a comparison of microblog search and web search. In I. King, W. Nejdl, and H. Li, editors, *WSDM*, pages 35–44. ACM, 2011.

[129] N. Tintarev and J. Masthoff. The effectiveness of personalized movie explanations: An experiment using commercial meta-data. In W. Nejdl, J. Kay, P. Pu, and E. Herder, editors, *Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 5149 of *Lecture Notes in Computer Science*, pages 204–213. Springer Berlin, Heidelberg, 2008.

[130] E. Triantafillou, A. Pomportsis, S. Demetriadis, and E. Georgiadou. The value of adaptivity based on cognitive style: an empirical study. *British Journal of Educational Technology*, 35(1):95–106, 2004.

[131] T. Tsandilas and M. Schraefel. Adaptive presentation supporting focus and context. In P. D. Bra, editor, *AH 2003: Workshop on Adaptive Hypermedia and Adaptive Web Based Systems*, pages 193–200, 2003.

[132] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen. Dynamic integration of classifiers for handling concept drift. *Inf. Fusion*, 9:56–68, January 2008.

[133] W. van der Aalst, P. De Bra, G.-J. Houben, and Y. Kornatzky. Browsing Semantics in the 'Tower' Model. Technical Report 47, Eindhoven University of Technology, Eindhoven, the Netherlands, 1993.

[134] K. van der Sluijs and K. M. Höver. Integrating Adaptive Functionality in a LMS. *International Journal Emerging Technologies in Learning*, 4(4):46–50, 2009.

[135] E. Vasilyeva, M. Pechenizkiy, A. Tesanovic, E. Knutov, S. Verwer, and P. D. Bra. Towards edm framework for personalization of information services in rpm systems. In R. S. J. de Baker, A. Merceron, and P. I. P. Jr., editors, *EDM*, pages 331–332. www.educationaldatamining.org, 2010.

[136] R. Vdovjak and G.-J. Houben. Providing the semantic layer for wis design. In *Proc. of the 14th International Conference on Advanced Information Systems Engineering*, CAiSE '02, pages 584–599, London, UK, UK, 2002. Springer-Verlag.

[137] Y. Wang, N. Stash, L. Aroyo, P. Gorgels, L. Rutledge, and G. Schreiber. Recommendations based on semantically enriched museum collections. *Web Semant.*, 6:283–290, November 2008.

[138] E. J. Whitehead, Jr. Webdav and deltav: collaborative authoring, versioning, and configuration management for the web. In *Proceedings of the 12th ACM conference on Hypertext and Hypermedia*, HYPERTEXT '01, pages 259–260, New York, NY, USA, 2001. ACM.

[139] H. Wu. *AHAM*. PhD thesis, Eindhoven University of Technology, Eindhoven, the Netherlands, 2002.

[140] T. W. Yan, M. Jacobsen, H. Garcia-Molina, and U. Dayal. From user access patterns to dynamic hypertext linking. *Comput. Netw. ISDN Syst.*, 28:1007–1014, May 1996.

List of Figures

List of Tables

Publication List

[1] E. Knutov, P. De Bra and M. Pechenizkiy, *AH 12 Years Later: a Comprehensive Survey of Adaptive Hypermedia Methods and Techniques*. New Review of Hypermedia and Multimedia 15(1), Taylor & Francis, UK, pp. 5-38, 2009.

[2] E. Knutov, P. De Bra and M. Pechenizkiy, *Generic Adaptation Framework: a Process-Oriented Perspective*. Journal of Digital Information 12(1), USA, 2011.

[3] E. Knutov, P. De Bra and M. Pechenizkiy, *Provenance Meets Adaptive Hypermedia*. HT'10: Proceedings of the 21st ACM Conference on Hypertext and Hypermedia, ACM, pp. 93-98, 2010.

[4] E. Knutov, P. De Bra, D. Smits and M. Pechenizkiy, *Bridging Navigation, Search and Adaptation. AH Models Evolution*. WEBIST'11: Proceedings of the 7th International Conference on Web Information Systems and Technologies, SciTePress, pp. 314-321, 2011.

[5] E. Knutov, P. De Bra and M. Pechenizkiy, *Versioning in Adaptive Hypermedia*. Proceedings of the 1st DAH'2009 Workshop on Dynamic and Adaptive Hypertext: Generic Frameworks, Approaches and Techniques, CEUR-WS, pp. 61-71, 2009.

[6] E. Knutov, P. De Bra and M. Pechenizkiy, *Generic Adaptation Process*. Proceedings of the WABBWUAS'2010 Workshop on Architectures and Building Blocks of Web-based User-Adaptive Systems, CEUR-WS, pp. 13-24, 2010.

[7] J. Hannon, E. Knutov, P. De Bra, M. Pechenizkiy, B. Smyth and K. McCarthy, *Bridging Recommendation and Adaptation: Generic Adaptation Framework - Twittomender compliance study*. Proceedings of 2nd DAH'2011 Workshop on Dynamic and Adaptive Hypertext, CEUR-WS, pp. 1-9, 2011.

[8] E. Knutov, P. De Bra and M. Pechenizkiy, *Adaptive Hypermedia Systems Analysis Approach by Means of the GAF Framework*. Proceedings of 2nd DAH'2011 Workshop on Dynamic and Adaptive Hypertext, CEUR-WS, pp. 41-46, 2011.

[9] E. Knutov, P. De Bra and M. Pechenizkiy, *Adaptation and Search: from Dexter and AHAM to GAF*. HT'10: Proceedings of the 21st ACM Conference on Hypertext and Hypermedia, ACM, pp. 281-282, 2010.

[10] E. Vasilyeva, M. Pechenizkiy, A. Tesanovic, E. Knutov and P. De Bra, *Towards EDM framework for Personalization of Information Services in RPM Systems*. Proceedings of the 3rd Conference on Educational Data Mining (EDM), pp. 331-332, 2010.

[11] P. De Bra, D. Smits, E. Knutov, E. Ploum, K. van der Sluijs, *Unifying Adaptive Learning Environments: authoring styles in the GRAPPLE project*. Adjunct Proceedings First International Conference on User Modeling, Adaptation, and Personalization (UMAP 2009), pp. 121-126, 2009.

[12] E. Knutov, *GAF: Generic Adaptation Framework*. AH'08 Proceedings of the Fifth International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Berlin/Heidelberg: Springer, pp. 400-404, 2008.

Summary

The research field of adaptive hypermedia (AH) and adaptive web-based information systems (AHS for short) has been growing rapidly during the past 15 years and this has resulted in new terms, models, methodologies, and a plethora of new systems. Adaptive systems are becoming more popular as tools for user-driven access to information. Adaptation of an information system or service to a user has been proven to be a powerful and useful concept. It is particularly helpful for the reduction of the information overload, which is frequently experienced on the Internet or any other information system of a large scale. But it is equally helpful for guiding users towards interesting topics, products, artifacts or descriptions thereof in electronic shops, libraries or museums, or for filtering appropriate items from a general or a domain-specific news feed.

The Generic Adaptation Framework (GAF) presented in this dissertation first and foremost creates a common framework for describing current and future AHS and adaptive web-based systems in general. It provides a commonly agreed upon taxonomy and a reference model that encompasses the most general architectures of the present and future, including conventional AHS, and different types of personalization-enabling systems and applications such as Recommender Systems (RS), personalized web search, semantic web enabled applications used in personalized information delivery, adaptive e-Learning applications and many more. At the same time GAF is trying to bring together two (seemingly disjoint) views on the adaptation: a classical pre-authored type, with conventional domain and overlay user models and data-driven adaptation which includes a set of data mining, machine learning and information retrieval tools. To bring these research fields together we conducted a number GAF compliance studies including RS (such as HeyStaks and Twittomender), AHS (such as KBS-Hyperbook, AHA!, APeLS and others), and other applications (e.g. CHIP (Cultural Heritage Information Personalization) and RPM (Remote Patient Management) systems) combining adaptation, recommendation and search. We also performed a number of real systems case-studies and a detailed analysis and evaluation of the framework. Secondly it introduces a number of new ideas in the field of AH, such as the Generic Adaptation Process (GAP), which aligns with a layered (data-oriented) architecture and serves as a reference adaptation process. This also helps to understand the compliance features mentioned earlier. Besides that GAF deals with important and novel aspects of adaptation enabling and leveraging technologies such as provenance and versioning. The existence of such a reference basis should stimulate AHS research and enable researchers to demonstrate ideas for new adaptation methods much more quickly than if they had to start from scratch. GAF thus helps bootstrap any adaptive web-based system research, design, analysis and evaluation.

The GAF reference model presented in the dissertation is important for a number of reasons. First and foremost it allows people to communicate about the architecture of AH systems using a common framework and vocabulary. It allows people to specify the functionality of a system and to compare the architecture and functionality of different systems using the same terms and definitions. And this becomes the key point because many researchers simply dont use the same terminology to describe similar or identical things thus producing systems with the same or slightly different functionality over again. Besides, they hardly look out for seemingly unrelated concepts from outer research fields and hardly accept new trends leaving AH research field relatively closed. Essentially in this dissertation we offer a common AH research language for the researchers to communicate, explain the terms and definitions they've been using, elaborate on the functionality, which has multiple interpretations. The reference model also stimulates the development of new systems and application that match more of the functionality captured by the reference model (than the systems existing before the creation of the reference model). Thus having an up-to-date common reference baseline new systems would be able to compare and evaluate the functionality (basically speaking the same language). Besides, previous reference models described the architecture of mainly closed and small to medium size adaptive hypermedia applications, besides many newer aspects that have entered the AH research field cannot be expressed and are not implemented. Thus there is also a need to breach this and bring adjacent research initiatives in the new generic reference AH model which the current dissertation covers.

Samenvatting

Het onderzoeksgebied van adaptieve hypermedia (AH) en adaptieve web-gebaseerde informatiesystemen heeft een snelle groei doorgemaakt gedurende de voorbije 15 jaar en deze groei heeft geresulteerd in nieuwe termen, modellen, methodologieën en een overvloed aan nieuwe systemen. Adaptieve systemen worden in toenemende mate populair als instrumenten voor user-driven toegang tot informatie. Adaptatie van een informatiesysteem of dienst aan een gebruiker is een krachtig en nuttig concept gebleken. Het helpt vooral de overvloed aan informatie terug te dringen die vaak ervaren wordt op het Internet of andere grote informatie systemen. Maar het helpt ook gebruikers te leiden naar naar "interessante" onderwerpen, producten, artefacten of beschrijvingen in electronica winkels, bibliotheken of musea, of om relevante artikelen te filteren uit een algemene of domein-specifieke nieuwsbron.
Het Generic Adaptation Framework (GAF) dat wordt gepresenteerd in deze dissertatie creëert ten eerste een algemeen raamwerk dat huidige en toekomstige AHS en web-gebaseerde adaptieve systemen in het algemeen beschrijft. Het introduceert een taxonomie en een referentiemodel dat de meeste algemene huidige en toekomstige architecturen omvat, inclusief conventionele AHS, en andere types van personaliseerbare systemen en toepassingen zoals Recommender Systems (RS), gepersonaliseerde web search, semantic web toepassingen die gebruikt worden in het aanbieden van gepersonaliseerde informatie, adaptieve e-learning toepassingen en vele anderen. Tegelijkertijd probeert GAF twee (ogenschijnlijk niet overlappende) inzichten in adaptatie te verenigen: een klassiek pre-authored type, met een conventioneel domein en overlay gebruikersmodel, en data-driven adaptatie wat een verzameling machine learing, datamining en information retrieval mechanismen omvat. Om deze onderzoeksgebieden samen te brengen hebben we een aantal compliance studies gedaan waaronder RS (zoals HeyStaks en Twittomender), AHS (zoals KBS-Hyperbook, AHA!, APeLS en anderen), en andere toepassingen (bijvoorbeeld CHIP (Cultural Heritage Information Personalization) en RPM (Remote Patient Management) systemen) waarin we adaptatie, aanbevelingen en zoekopdrachten combineren. We hebben ook een aantal casussen bestudeerd met echte systemen en een gedetailleerde analyse en evaluatie van het raamwerk uitgevoerd. Ten tweede introduceert het raamwerk een aantal nieuwe ideeën op het gebied van AH, zoals het Generic Adaptation Process (GAP) dat op een lijn ligt met een gelaagde (data-oriented) architectuur en dienst doet als een referentie adaptatieproces. Dit helpt ook met het begrijpen van de eerder beschreven compliance eigenschappen. Afgezien daarvan behandelt GAF ook belangrijke en nieuwe aspecten van het mogelijk en gebruik maken van adaptatie zoals provenance en versiebeheer. Het bestaan van een dergelijke referentiebasis zou het on-

derzoek naar AHS moeten stimuleren en onderzoekers in staat moeten stellen om nieuwe ideeën voor adaptatie methodes veel sneller te demonstreren dan wanneer zij vanaf nul hadden moeten beginnen. GAF helpt dus met het opstarten van een willekeurig adaptief web-gebaseerd systeemonderzoek, ontwerp, analyse en evaluatie.
Het GAF referentie model dat wordt gepresenteerd in deze dissertatie is belangrijk vanwege een aantal redenen. Ten eerste stelt het mensen in staat te communiceren over de architectuur van AH systemen in een gemeenschappelijk raamwerk en vocabulaire. Het stelt mensen in staat om de functionaliteit van een systeem te specificeren en om de architectuur en functionaliteit van verschillende systemen te vergelijken met dezelfde termen en definities. En dit is het centrale punt, omdat veel onderzoekers gewoonweg niet dezelfde terminologie gebruiken om gelijkaardige of identieke dingen te beschrijven en zo meerdere keren systemen met vergelijkbare functionaliteit produceren. Afgezien daarvan kijken de onderzoekers nauwelijks uit naar ogenschijnlijk ongerelateerde andere onderzoeksvelden en ze accepteren nauwelijks nieuwe trends waardoor het veld van AH relatief gesloten blijft. In deze dissertatie bieden we de onderzoekers een gemeenschappelijke AH onderzoekstaal om mee te communiceren, leggen we de termen en definities die ze gebruikt hebben uit en werken we de functionaliteit, die meerdere interpretaties heeft, verder uit. Het referentiemodel stimuleert ook de ontwikkeling van nieuwe systemen en toepassingen die meer overeenkomen met de functionaliteit zoals beschreven door het referentie model (dan de systemen die bestonden voor het creëren van het referentie model). En zo hebben ontwikkelaars van nieuwe systemen, gegeven een gemeenschappelijke up-to-date referentie baseline, de mogelijkheid om de funcionaliteit te vergelijken en te evalueren (omdat ze een gemeenschappeljke taal hebben). Behalve dat beschreven eerdere referentie modellen de architectuur van meestal gesloten en kleine tot middelgrote adaptieve AH toepassingen, zodat vele nieuwe aspecten die het AH onderzoeksveld zijn binnen gekomen niet konden uitgedrukt of geïmplementeerd worden. Dus is er een noodzaak om dit te doorbreken en om gerelateerde onderzoeksinitiatieven in het nieuwe generieke AH veld samen te brengen zoals wordt behandeld in deze dissertatie.

Curriculum Vitae

Evgeny Knutov was born on June 25, 1983 in Leningrad (now Saint-Petersburg), Soviet Union (now Russian Federation). After finishing a state gymnasium 116 (with honors) in 2000 in Saint-Petersburg, Russia, he studied Computer Technologies and Informatics at Saint-Petersburg State Electrotechnical University (LETI) in Saint-Petersburg, Russia. In 2006 he obtained an Engineering degree (with honors) in Computer Science. His thesis is entitled ATM networks LAN emulation modeling, supervised by prof. dr. Oleg Kutuzov. After the graduation he worked as a software engineer and continued a research in networks emulation modeling. In February 2008 he moved to the Netherlands and started a PhD project in the field of Adaptive Hypermedia under the supervision of prof. dr. Paul De Bra and dr. M. Pechenizkiy in the Databases and Hypermedia (DH) group at the department of Mathematics and Computer Science, Eindhoven University of Technology (TU/e). The research results are presented in this dissertation. Currently Evgeny is employed at the TU/e as a researcher. His research interests include various aspects of adaptive hypermedia and adaptive web-based systems in particular. He can be reached at eknutov@gmail.com.

SIKS Dissertatiereeks

1998
====

1998-1 Johan van den Akker (CWI)
DEGAS - An Active, Temporal Database of Autonomous Objects

1998-2 Floris Wiesman (UM)
Information Retrieval by Graphically Browsing Meta-Information

1998-3 Ans Steuten (TUD)
A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective

1998-4 Dennis Breuker (UM)
Memory versus Search in Games

1998-5 E.W.Oskamp (RUL)
Computerondersteuning bij Straftoemeting

1999
====

1999-1 Mark Sloof (VU)
Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products

1999-2 Rob Potharst (EUR)
Classification using decision trees and neural nets

1999-3 Don Beal (UM)
The Nature of Minimax Search

1999-4 Jacques Penders (UM)
The practical Art of Moving Physical Objects

1999-5 Aldo de Moor (KUB)
Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems

1999-6 Niek J.E. Wijngaards (VU)
Re-design of compositional systems

1999-7 David Spelt (UT)
Verification support for object database design

1999-8 Jacques H.J. Lenting (UM)
Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation.

2000
====

2000-1 Frank Niessink (VU)
Perspectives on Improving Software Maintenance

2000-2 Koen Holtman (TUE)
Prototyping of CMS Storage Management

2000-3 Carolien M.T. Metselaar (UVA)
Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief.

2000-4 Geert de Haan (VU)
ETAG, A Formal Model of Competence Knowledge for User Interface Design

2000-5 Ruud van der Pol (UM)
Knowledge-based Query Formulation in Information Retrieval

2000-6 Rogier van Eijk (UU)
Programming Languages for Agent Communication

2000-7 Niels Peek (UU)
Decision-theoretic Planning of Clinical Patient Management

2000-8 Veerle Coup (EUR)
Sensitivity Analyis of Decision-Theoretic Networks

2000-9 Florian Waas (CWI)
Principles of Probabilistic Query Optimization

2000-10 Niels Nes (CWI)
Image Database Management System Design Considerations, Algorithms and Architecture

2000-11 Jonas Karlsson (CWI)
Scalable Distributed Data Structures for Database Management

2001
====

2001-1 Silja Renooij (UU)
Qualitative Approaches to Quantifying Probabilistic Networks

2001-2 Koen Hindriks (UU)
Agent Programming Languages: Programming with Mental Models

2001-3 Maarten van Someren (UvA)
Learning as problem solving

2001-4 Evgueni Smirnov (UM)
Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets

2001-5 Jacco van Ossenbruggen (VU)
Processing Structured Hypermedia: A Matter of Style

2001-6 Martijn van Welie (VU)
Task-based User Interface Design

2001-7 Bastiaan Schonhage (VU)
Diva: Architectural Perspectives on Information Visualization

2001-8 Pascal van Eck (VU)
A Compositional Semantic Structure for Multi-Agent Systems Dynamics.

2001-9 Pieter Jan 't Hoen (RUL)
Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes

2001-10 Maarten Sierhuis (UvA)
Modeling and Simulating Work Practice
BRAHMS: a multiagent modeling and simulation language for work practice analysis and design

2001-11 Tom M. van Engers (VUA)
Knowledge Management:
The Role of Mental Models in Business Systems Design

2002
====

2002-01 Nico Lassing (VU)
Architecture-Level Modifiability Analysis

2002-02 Roelof van Zwol (UT)
Modelling and searching web-based document collections

2002-03 Henk Ernst Blok (UT)
Database Optimization Aspects for Information Retrieval

2002-04 Juan Roberto Castelo Valdueza (UU)
The Discrete Acyclic Digraph Markov Model in Data Mining

2002-05 Radu Serban (VU)
The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents

2002-06 Laurens Mommers (UL)
Applied legal epistemology; Building a knowledge-based ontology of the legal domain

2002-07 Peter Boncz (CWI)
Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications

2002-08 Jaap Gordijn (VU)
Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas

2002-09 Willem-Jan van den Heuvel(KUB)
Integrating Modern Business Applications with Objectified Legacy Systems

2002-10 Brian Sheppard (UM)
Towards Perfect Play of Scrabble

2002-11 Wouter C.A. Wijngaards (VU)
Agent Based Modelling of Dynamics: Biological and Organisational Applications

2002-12 Albrecht Schmidt (Uva)
Processing XML in Database Systems

2002-13 Hongjing Wu (TUE)
A Reference Architecture for Adaptive Hypermedia Applications

2002-14 Wieke de Vries (UU)
Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems

2002-15 Rik Eshuis (UT)
Semantics and Verification of UML Activity Diagrams for Workflow Modelling

2002-16 Pieter van Langen (VU)
The Anatomy of Design: Foundations, Models and Applications

2002-17 Stefan Manegold (UVA)
Understanding, Modeling, and Improving Main-Memory Database Performance

2003
====

2003-01 Heiner Stuckenschmidt (VU)
Ontology-Based Information Sharing in Weakly Structured Environments

2003-02 Jan Broersen (VU)
Modal Action Logics for Reasoning About Reactive Systems

2003-03 Martijn Schuemie (TUD)
Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy

2003-04 Milan Petkovic (UT)

Content-Based Video Retrieval Supported by Database Technology

2003-05 Jos Lehmann (UVA)
Causation in Artificial Intelligence and Law - A modelling approach

2003-06 Boris van Schooten (UT)
Development and specification of virtual environments

2003-07 Machiel Jansen (UvA)
Formal Explorations of Knowledge Intensive Tasks

2003-08 Yongping Ran (UM)
Repair Based Scheduling

2003-09 Rens Kortmann (UM)
The resolution of visually guided behaviour

2003-10 Andreas Lincke (UvT)
Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture

2003-11 Simon Keizer (UT)
Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks

2003-12 Roeland Ordelman (UT)
Dutch speech recognition in multimedia information retrieval

2003-13 Jeroen Donkers (UM)
Nosce Hostem - Searching with Opponent Models

2003-14 Stijn Hoppenbrouwers (KUN)
Freezing Language: Conceptualisation Processes across ICT-Supported Organisations

2003-15 Mathijs de Weerdt (TUD)
Plan Merging in Multi-Agent Systems

2003-16 Menzo Windhouwer (CWI)
Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses

2003-17 David Jansen (UT)
Extensions of Statecharts with Probability, Time, and Stochastic Timing

2003-18 Levente Kocsis (UM)
Learning Search Decisions

2004
====

2004-01 Virginia Dignum (UU)
A Model for Organizational Interaction: Based on Agents, Founded in Logic

2004-02 Lai Xu (UvT)
Monitoring Multi-party Contracts for E-business

2004-03 Perry Groot (VU)
A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving

2004-04 Chris van Aart (UVA)
Organizational Principles for Multi-Agent Architectures

2004-05 Viara Popova (EUR)
Knowledge discovery and monotonicity

2004-06 Bart-Jan Hommes (TUD)
The Evaluation of Business Process Modeling Techniques

2004-07 Elise Boltjes (UM)
Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes

2004-08 Joop Verbeek(UM)
Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise

2004-09 Martin Caminada (VU)
For the Sake of the Argument; explorations into argument-based reasoning

2004-10 Suzanne Kabel (UVA)
Knowledge-rich indexing of learning-objects

2004-11 Michel Klein (VU)
Change Management for Distributed Ontologies

2004-12 The Duy Bui (UT)
Creating emotions and facial expressions for embodied agents

2004-13 Wojciech Jamroga (UT)
Using Multiple Models of Reality: On Agents who Know how to Play

2004-14 Paul Harrenstein (UU)
Logic in Conflict. Logical Explorations in Strategic Equilibrium

2004-15 Arno Knobbe (UU)
Multi-Relational Data Mining

2004-16 Federico Divina (VU)
Hybrid Genetic Relational Search for Inductive Learning

2004-17 Mark Winands (UM)
Informed Search in Complex Games

2004-18 Vania Bessa Machado (UvA)
Supporting the Construction of Qualitative Knowledge Models

2004-19 Thijs Westerveld (UT)
Using generative probabilistic models for multimedia retrieval

2004-20 Madelon Evers (Nyenrode)
Learning from Design: facilitating multidisciplinary design teams

2005
====

2005-01 Floor Verdenius (UVA)
Methodological Aspects of Designing Induction-Based Applications

2005-02 Erik van der Werf (UM))
AI techniques for the game of Go

2005-03 Franc Grootjen (RUN)
A Pragmatic Approach to the Conceptualisation of Language

2005-04 Nirvana Meratnia (UT)
Towards Database Support for Moving Object data

2005-05 Gabriel Infante-Lopez (UVA)
Two-Level Probabilistic Grammars for Natural Language Parsing

2005-06 Pieter Spronck (UM)
Adaptive Game AI

2005-07 Flavius Frasincar (TUE)
Hypermedia Presentation Generation for Semantic Web Information Systems

2005-08 Richard Vdovjak (TUE)
A Model-driven Approach for Building Distributed Ontology-based Web Applications

2005-09 Jeen Broekstra (VU)
Storage, Querying and Inferencing for Semantic Web Languages

2005-10 Anders Bouwer (UVA)
Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments

2005-11 Elth Ogston (VU)
Agent Based Matchmaking and Clustering - A Decentralized Approach to Search

2005-12 Csaba Boer (EUR)
Distributed Simulation in Industry

2005-13 Fred Hamburg (UL)
Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen

2005-14 Borys Omelayenko (VU)
Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics

2005-15 Tibor Bosse (VU)
Analysis of the Dynamics of Cognitive Processes

2005-16 Joris Graaumans (UU)
Usability of XML Query Languages

2005-17 Boris Shishkov (TUD)
Software Specification Based on Re-usable Business Components

2005-18 Danielle Sent (UU)
Test-selection strategies for probabilistic networks

2005-19 Michel van Dartel (UM)
Situated Representation

2005-20 Cristina Coteanu (UL)
Cyber Consumer Law, State of the Art and Perspectives

2005-21 Wijnand Derks (UT)
Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics

2006
====

2006-01 Samuil Angelov (TUE)
Foundations of B2B Electronic Contracting

2006-02 Cristina Chisalita (VU)
Contextual issues in the design and use of information technology in organizations

2006-03 Noor Christoph (UVA)
The role of metacognitive skills in learning to solve problems

2006-04 Marta Sabou (VU)
Building Web Service Ontologies

2006-05 Cees Pierik (UU)
Validation Techniques for Object-Oriented Proof Outlines

2006-06 Ziv Baida (VU)
Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling

2006-07 Marko Smiljanic (UT)
XML schema matching – balancing efficiency and effectiveness by means of clustering

2006-08 Eelco Herder (UT)
Forward, Back and Home Again - Analyzing User Behavior on the Web

2006-09 Mohamed Wahdan (UM)
Automatic Formulation of the Auditor's Opinion

2006-10 Ronny Siebes (VU)
Semantic Routing in Peer-to-Peer Systems

2006-11 Joeri van Ruth (UT)
Flattening Queries over Nested Data Types

2006-12 Bert Bongers (VU)
Interactivation - Towards an e-cology of people, our technological environment, and the arts

2006-13 Henk-Jan Lebbink (UU)
Dialogue and Decision Games for Information Exchanging Agents

2006-14 Johan Hoorn (VU)

Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change

2006-15 Rainer Malik (UU)
CONAN: Text Mining in the Biomedical Domain

2006-16 Carsten Riggelsen (UU)
Approximation Methods for Efficient Learning of Bayesian Networks

2006-17 Stacey Nagata (UU)
User Assistance for Multitasking with Interruptions on a Mobile Device

2006-18 Valentin Zhizhkun (UVA)
Graph transformation for Natural Language Processing

2006-19 Birna van Riemsdijk (UU)
Cognitive Agent Programming: A Semantic Approach

2006-20 Marina Velikova (UvT)
Monotone models for prediction in data mining

2006-21 Bas van Gils (RUN)
Aptness on the Web

2006-22 Paul de Vrieze (RUN)
Fundaments of Adaptive Personalisation

2006-23 Ion Juvina (UU)
Development of Cognitive Model for Navigating on the Web

2006-24 Laura Hollink (VU)
Semantic Annotation for Retrieval of Visual Resources

2006-25 Madalina Drugan (UU)
Conditional log-likelihood MDL and Evolutionary MCMC

2006-26 Vojkan Mihajlovic (UT)
Score Region Algebra: A Flexible Framework for Structured Information Retrieval

2006-27 Stefano Bocconi (CWI)
Vox Populi: generating video documentaries from semantically annotated media repositories

2006-28 Borkur Sigurbjornsson (UVA)
Focused Information Access using XML Element Retrieval

2007
====

2007-01 Kees Leune (UvT)
Access Control and Service-Oriented Architectures

2007-02 Wouter Teepe (RUG)
Reconciling Information Exchange and Confidentiality: A Formal Approach

2007-03 Peter Mika (VU)
Social Networks and the Semantic Web

2007-04 Jurriaan van Diggelen (UU)
Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach

2007-05 Bart Schermer (UL)
Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance

2007-06 Gilad Mishne (UVA)
Applied Text Analytics for Blogs

2007-07 Natasa Jovanovic' (UT)
To Whom It May Concern - Addressee Identification in Face-to-Face Meetings

2007-08 Mark Hoogendoorn (VU)
Modeling of Change in Multi-Agent Organizations

2007-09 David Mobach (VU)
Agent-Based Mediated Service Negotiation

2007-10 Huib Aldewereld (UU)
Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols

2007-11 Natalia Stash (TUE)
Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System

2007-12 Marcel van Gerven (RUN)
Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty

2007-13 Rutger Rienks (UT)
Meetings in Smart Environments; Implications of Progressing Technology

2007-14 Niek Bergboer (UM)
Context-Based Image Analysis

2007-15 Joyca Lacroix (UM)
NIM: a Situated Computational Memory Model

2007-16 Davide Grossi (UU)
Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems

2007-17 Theodore Charitos (UU)
Reasoning with Dynamic Networks in Practice

2007-18 Bart Orriens (UvT)
On the development an management of adaptive business collaborations

2007-19 David Levy (UM)
Intimate relationships with artificial partners

2007-20 Slinger Jansen (UU)
Customer Configuration Updating in a Software Supply Network

2007-21 Karianne Vermaas (UU)
Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005

2007-22 Zlatko Zlatev (UT)
Goal-oriented design of value and process models from patterns

2007-23 Peter Barna (TUE)
Specification of Application Logic in Web Information Systems

2007-24 Georgina Ramrez Camps (CWI)
Structural Features in XML Retrieval

2007-25 Joost Schalken (VU)
Empirical Investigations in Software Process Improvement

2008
====

2008-01 Katalin Boer-Sorbn (EUR)
Agent-Based Simulation of Financial Markets: A modular, continuous-time approach

2008-02 Alexei Sharpanskykh (VU)
On Computer-Aided Methods for Modeling and Analysis of Organizations

2008-03 Vera Hollink (UVA)
Optimizing hierarchical menus: a usage-based approach

2008-04 Ander de Keijzer (UT)
Management of Uncertain Data - towards unattended integration

2008-05 Bela Mutschler (UT)
Modeling and simulating causal dependencies on process-aware information systems from a cost perspective

2008-06 Arjen Hommersom (RUN)
On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective

2008-07 Peter van Rosmalen (OU)
Supporting the tutor in the design and support of adaptive e-learning

2008-08 Janneke Bolt (UU)
Bayesian Networks: Aspects of Approximate Inference

2008-09 Christof van Nimwegen (UU)
The paradox of the guided user: assistance can be counter-effective

2008-10 Wauter Bosma (UT)
Discourse oriented summarization

2008-11 Vera Kartseva (VU)
Designing Controls for Network Organizations: A Value-Based Approach

2008-12 Jozsef Farkas (RUN)
A Semiotically Oriented Cognitive Model of Knowledge Representation

2008-13 Caterina Carraciolo (UVA)
Topic Driven Access to Scientific Handbooks

2008-14 Arthur van Bunningen (UT)
Context-Aware Querying; Better Answers with Less Effort

2008-15 Martijn van Otterlo (UT)
The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains.

2008-16 Henriette van Vugt (VU)
Embodied agents from a user's perspective

2008-17 Martin Op 't Land (TUD)
Applying Architecture and Ontology to the Splitting and Allying of Enterprises

2008-18 Guido de Croon (UM)
Adaptive Active Vision

2008-19 Henning Rode (UT)
From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search

2008-20 Rex Arendsen (UVA)
Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven

2008-21 Krisztian Balog (UVA)
People Search in the Enterprise

2008-22 Henk Koning (UU)
Communication of IT-Architecture

2008-23 Stefan Visscher (UU)
Bayesian network models for the management of ventilator-associated pneumonia

2008-24 Zharko Aleksovski (VU)
Using background knowledge in ontology matching

2008-25 Geert Jonker (UU)
Efficient and Equitable Exchange in Air Traffic Management Plan Repair using Spender-signed Currency

2008-26 Marijn Huijbregts (UT)
Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled

2008-27 Hubert Vogten (OU)
Design and Implementation Strategies for IMS Learning Design

2008-28 Ildiko Flesch (RUN)
On the Use of Independence Relations in Bayesian Networks

2008-29 Dennis Reidsma (UT)
Annotations and Subjective Machines - Of Annotators, Embodied Agents, Users, and Other Humans

2008-30 Wouter van Atteveldt (VU)
Semantic Network Analysis: Techniques for Extracting, Representing and Querying Media Content

2008-31 Loes Braun (UM)
Pro-Active Medical Information Retrieval

2008-32 Trung H. Bui (UT)
Toward Affective Dialogue Management using Partially Observable Markov Decision Processes

2008-33 Frank Terpstra (UVA)
Scientific Workflow Design; theoretical and practical issues

2008-34 Jeroen de Knijf (UU)
Studies in Frequent Tree Mining

2008-35 Ben Torben Nielsen (UvT)
Dendritic morphologies: function shapes structure

2009
====

2009-01 Rasa Jurgelenaite (RUN)
Symmetric Causal Independence Models

2009-02 Willem Robert van Hage (VU)
Evaluating Ontology-Alignment Techniques

2009-03 Hans Stol (UvT)
A Framework for Evidence-based Policy Making Using IT

2009-04 Josephine Nabukenya (RUN)
Improving the Quality of Organisational Policy Making using Collaboration Engineering

2009-05 Sietse Overbeek (RUN)
Bridging Supply and Demand for Knowledge Intensive Tasks-Based on Knowledge, Cognition, and Quality

2009-06 Muhammad Subianto (UU)
Understanding Classification

2009-07 Ronald Poppe (UT)
Discriminative Vision-Based Recovery and Recognition of Human Motion

2009-08 Volker Nannen (VU)
Evolutionary Agent-Based Policy Analysis in Dynamic Environments

2009-09 Benjamin Kanagwa (RUN)
Design, Discovery and Construction of Service-oriented Systems

2009-10 Jan Wielemaker (UVA)
Logic programming for knowledge-intensive interactive applications

2009-11 Alexander Boer (UVA)
Legal Theory, Sources of Law and the Semantic Web

2009-12 Peter Massuthe (TUE, Humboldt-Universitaet zu Berlin)
perating Guidelines for Services

2009-13 Steven de Jong (UM)
Fairness in Multi-Agent Systems

2009-14 Maksym Korotkiy (VU)
From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)

2009-15 Rinke Hoekstra (UVA)
Ontology Representation - Design Patterns and Ontologies that Make Sense

2009-16 Fritz Reul (UvT)
New Architectures in Computer Chess

2009-17 Laurens van der Maaten (UvT)
Feature Extraction from Visual Data

2009-18 Fabian Groffen (CWI)
Armada, An Evolving Database System

2009-19 Valentin Robu (CWI)
Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets

2009-20 Bob van der Vecht (UU)
Adjustable Autonomy: Controling Influences on Decision Making

2009-21 Stijn Vanderlooy (UM)
Ranking and Reliable Classification

2009-22 Pavel Serdyukov (UT)
Search For Expertise: Going beyond direct evidence

2009-23 Peter Hofgesang (VU)
Modelling Web Usage in a Changing Environment

2009-24 Annerieke Heuvelink (VUA)
Cognitive Models for Training Simulations

2009-25 Alex van Ballegooij (CWI)
RAM: Array Database Management through Relational Mapping

2009-26 Fernando Koch (UU)
An Agent-Based Model for the Development of Intelligent Mobile Services

2009-27 Christian Glahn (OU)
Contextual Support of social Engagement and Reflection on the Web

2009-28 Sander Evers (UT)
Sensor Data Management with Probabilistic Models

2009-29 Stanislav Pokraev (UT)
Model-Driven Semantic Integration of Service-Oriented Applications

2009-30 Marcin Zukowski (CWI)
Balancing vectorized query execution with bandwidth-optimized storage

2009-31 Sofiya Katrenko (UVA)
A Closer Look at Learning Relations from Text

2009-32 Rik Farenhorst (VU) and Remco de Boer (VU)

Architectural Knowledge Management: Supporting Architects and Auditors

2009-33 Khiet Truong (UT)
How Does Real Affect Affect Affect Recognition In Speech?

2009-34 Inge van de Weerd (UU)
Advancing in Software Product Management: An Incremental Method Engineering Approach

2009-35 Wouter Koelewijn (UL)
Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling

2009-36 Marco Kalz (OUN)
Placement Support for Learners in Learning Networks

2009-37 Hendrik Drachsler (OUN)
Navigation Support for Learners in Informal Learning Networks

2009-38 Riina Vuorikari (OU)
Tags and self-organisation: a metadata ecology for learning resources in a multilingual context

2009-39 Christian Stahl (TUE, Humboldt-Universitaet zu Berlin)
Service Substitution – A Behavioral Approach Based on Petri Nets

2009-40 Stephan Raaijmakers (UvT)
Multinomial Language Learning: Investigations into the Geometry of Language

2009-41 Igor Berezhnyy (UvT)
Digital Analysis of Paintings

2009-42 Toine Bogers
Recommender Systems for Social Bookmarking

2009-43 Virginia Nunes Leal Franqueira (UT)
Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients

2009-44 Roberto Santana Tapia (UT)
Assessing Business-IT Alignment in Networked Organizations

2009-45 Jilles Vreeken (UU)
Making Pattern Mining Useful

2009-46 Loredana Afanasiev (UvA)
Querying XML: Benchmarks and Recursion

2010
====

2010-01 Matthijs van Leeuwen (UU)
Patterns that Matter

2010-02 Ingo Wassink (UT)
Work flows in Life Science

2010-03 Joost Geurts (CWI)
A Document Engineering Model and Processing Framework for Multimedia documents

2010-04 Olga Kulyk (UT)
Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments

2010-05 Claudia Hauff (UT)
Predicting the Effectiveness of Queries and Retrieval Systems

2010-06 Sander Bakkes (UvT)
Rapid Adaptation of Video Game AI

2010-07 Wim Fikkert (UT)
Gesture interaction at a Distance

2010-08 Krzysztof Siewicz (UL)
Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments

2010-09 Hugo Kielman (UL)
A Politiele gegevensverwerking en Privacy, Naar een effectieve waarborging

2010-10 Rebecca Ong (UL)
Mobile Communication and Protection of Children

2010-11 Adriaan Ter Mors (TUD)
The world according to MARP: Multi-Agent Route Planning

2010-12 Susan van den Braak (UU)
Sensemaking software for crime analysis

2010-13 Gianluigi Folino (RUN)
High Performance Data Mining using Bio-inspired techniques

2010-14 Sander van Splunter (VU)
Automated Web Service Reconfiguration

2010-15 Lianne Bodenstaff (UT)
Managing Dependency Relations in Inter-Organizational Models

2010-16 Sicco Verwer (TUD)
Efficient Identification of Timed Automata, theory and practice

2010-17 Spyros Kotoulas (VU)
Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications

2010-18 Charlotte Gerritsen (VU)
Caught in the Act: Investigating Crime by Agent-Based Simulation

2010-19 Henriette Cramer (UvA)
People's Responses to Autonomous and Adaptive Systems

2010-20 Ivo Swartjes (UT)
Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative

2010-21 Harold van Heerde (UT)
Privacy-aware data management by means of

data degradation

2010-22 Michiel Hildebrand (CWI)
End-user Support for Access to Heterogeneous Linked Data

2010-23 Bas Steunebrink (UU)
The Logical Structure of Emotions

2010-24 Dmytro Tykhonov
Designing Generic and Efficient Negotiation Strategies

2010-25 Zulfiqar Ali Memon (VU)
Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective

2010-26 Ying Zhang (CWI)
XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines

2010-27 Marten Voulon (UL)
Automatisch contracteren

2010-28 Arne Koopman (UU)
Characteristic Relational Patterns

2010-29 Stratos Idreos(CWI)
Database Cracking: Towards Auto-tuning Database Kernels

2010-30 Marieke van Erp (UvT)
Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval

2010-31 Victor de Boer (UVA)
Ontology Enrichment from Heterogeneous Sources on the Web

2010-32 Marcel Hiel (UvT)
An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems

2010-33 Robin Aly (UT)
Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval

2010-34 Teduh Dirgahayu (UT)
Interaction Design in Service Compositions

2010-35 Dolf Trieschnigg (UT)
Proof of Concept: Concept-based Biomedical Information Retrieval

2010-36 Jose Janssen (OU)
Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification

2010-37 Niels Lohmann (TUE)
Correctness of services and their composition

2010-38 Dirk Fahland (TUE)
From Scenarios to components

2010-39 Ghazanfar Farooq Siddiqui (VU)
Integrative modeling of emotions in virtual agents

2010-40 Mark van Assem (VU)
Converting and Integrating Vocabularies for the Semantic Web

2010-41 Guillaume Chaslot (UM)
Monte-Carlo Tree Search

2010-42 Sybren de Kinderen (VU)
Needs-driven service bundling in a multi-supplier setting the computational e3-service approach

2010-43 Peter van Kranenburg (UU)
A Computational Approach to Content-Based Retrieval of Folk Song Melodies

2010-44 Pieter Bellekens (TUE)
An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain

2010-45 Vasilios Andrikopoulos (UvT)
A theory and model for the evolution of software services

2010-46 Vincent Pijpers (VU)
e3alignment: Exploring Inter-Organizational Business-ICT Alignment

2010-47 Chen Li (UT)
Mining Process Model Variants: Challenges, Techniques, Examples

2010-48 Milan Lovric (EUR)
Behavioral Finance and Agent-Based Artificial Markets

2010-49 Jahn-Takeshi Saito (UM)
Solving difficult game positions

2010-50 Bouke Huurnink (UVA)
Search in Audiovisual Broadcast Archives

2010-51 Alia Khairia Amin (CWI)
Understanding and supporting information seeking tasks in multiple sources

2010-52 Peter-Paul van Maanen (VU)
Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention

2010-53 Edgar Meij (UVA)
Combining Concepts and Language Models for Information Access

2011
====

2011-01 Botond Cseke (RUN)
Variational Algorithms for Bayesian Inference in Latent Gaussian Models

2011-02 Nick Tinnemeier(UU)
Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language

2011-03 Jan Martijn van der Werf (TUE)
Compositional Design and Verification of Component-Based Information Systems

2011-04 Hado van Hasselt (UU)
Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference learning algorithms

2011-05 Base van der Raadt (VU)
Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.

2011-06 Yiwen Wang (TUE)
Semantically-Enhanced Recommendations in Cultural Heritage

2011-07 Yujia Cao (UT)
Multimodal Information Presentation for High Load Human Computer Interaction

2011-08 Nieske Vergunst (UU)
BDI-based Generation of Robust Task-Oriented Dialogues

2011-09 Tim de Jong (OU)
Contextualised Mobile Media for Learning

2011-10 Bart Bogaert (UvT)
Cloud Content Contention

2011-11 Dhaval Vyas (UT)
Designing for Awareness: An Experience-focused HCI Perspective

2011-12 Carmen Bratosin (TUE)
Grid Architecture for Distributed Process Mining

2011-13 Xiaoyu Mao (UvT)
Airport under Control. Multiagent Scheduling for Airport Ground Handling

2011-14 Milan Lovric (EUR)
Behavioral Finance and Agent-Based Artificial Markets

2011-15 Marijn Koolen (UvA)
The Meaning of Structure: the Value of Link Evidence for Information Retrieval

2011-16 Maarten Schadd (UM)
Selective Search in Games of Different Complexity

2011-17 Jiyin He (UVA)
Exploring Topic Structure: Coherence, Diversity and Relatedness

2011-18 Mark Ponsen (UM)
Strategic Decision-Making in complex games

2011-19 Ellen Rusman (OU)
The Mind ' s Eye on Personal Profiles

2011-20 Qing Gu (VU)
Guiding service-oriented software engineering - A view-based approach

2011-21 Linda Terlouw (TUD)
Modularization and Specification of Service-Oriented Systems

2011-22 Junte Zhang (UVA)
System Evaluation of Archival Description and Access

2011-23 Wouter Weerkamp (UVA)
Finding People and their Utterances in Social Media

2011-24 Herwin van Welbergen (UT)
Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior

2011-25 Syed Waqar ul Qounain Jaffry (VU)
Analysis and Validation of Models for Trust Dynamics

2011-26 Matthijs Aart Pontier (VU)
Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots

2011-27 Aniel Bhulai (VU)
Dynamic website optimization through autonomous management of design patterns

2011-28 Rianne Kaptein(UVA)
Effective Focused Retrieval by Exploiting Query Context and Document Structure

2011-29 Faisal Kamiran (TUE)
Discrimination-aware Classification

2011-30 Egon van den Broek (UT)
Affective Signal Processing (ASP): Unraveling the mystery of emotions

2011-31 Ludo Waltman (EUR)
Computational and Game-Theoretic Approaches for Modeling Bounded Rationality

2011-32 Nees-Jan van Eck (EUR)
Methodological Advances in Bibliometric Mapping of Science

2011-33 Tom van der Weide (UU)
Arguing to Motivate Decisions

2011-34 Paolo Turrini (UU)
Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations

2011-35 Maaike Harbers (UU)
Explaining Agent Behavior in Virtual Training

2011-36 Erik van der Spek (UU)
Experiments in serious game design: a cognitive approach

2011-37 Adriana Burlutiu (RUN)
Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference

2011-38 Nyree Lemmens (UM)
Bee-inspired Distributed Optimization

2011-39 Joost Westra (UU)
Organizing Adaptation using Agents in Serious Games

2011-40 Viktor Clerc (VU)
Architectural Knowledge Management in Global Software Development

2011-41 Luan Ibraimi (UT)
Cryptographically Enforced Distributed Data Access Control

2011-42 Michal Sindlar (UU)
Explaining Behavior through Mental State Attribution

2011-43 Henk van der Schuur (UU)
Process Improvement through Software Operation Knowledge

2011-44 Boris Reuderink (UT)
Robust Brain-Computer Interfaces

2011-45 Herman Stehouwer (UvT)
Statistical Language Models for Alternative Sequence Selection

2011-46 Beibei Hu (TUD)
Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work

2011-47 Azizi Bin Ab Aziz(VU)
Exploring Computational Models for Intelligent Support of Persons with Depression

2011-48 Mark Ter Maat (UT)
Response Selection and Turn-taking for a Sensitive Artificial Listening Agent

2011-49 Andreea Niculescu (UT)
Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality

2012
====

2012-01 Terry Kakeeto (UvT)
Relationship Marketing for SMEs in Uganda

2012-02 Muhammad Umair(VU)
Adaptivity, emotion, and Rationality in Human and Ambient Agent Models

2012-03 Adam Vanya (VU)
Supporting Architecture Evolution by Mining Software Repositories

2012-04 Jurriaan Souer (UU)
Development of Content Management System-based Web Applications

2012-05 Marijn Plomp (UU)
Maturing Interorganisational Information Systems

2012-06 Wolfgang Reinhardt (OU)
Awareness Support for Knowledge Workers in Research Networks

2012-07 Rianne van Lambalgen (VU)
When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions

2012-08 Gerben de Vries (UVA)
Kernel Methods for Vessel Traject

2012-09 Ricardo Neisse (UT)
Trust and Privacy Management Support for Context-Aware Service Platforms

2012-10 David Smits (TUE)
Towards a Generic Distributed Adaptive Hypermedia Environment

2012-11 J.C.B. Rantham Prabhakara (TUE)
Process Mining in the Large: Preprocessing, Discovery, and Diagnostics

2012-12 Kees van der Sluijs (TUE)
Model Driven Design and Data Integration in Semantic Web Information Systems

2012-13 Suleman Shahid (UvT)
Fun and Face: Exploring non-verbal expressions of emotion during playful interactions