Proceedings of the

First International Workshop on Dynamic and Adaptive Hypertext: *Generic Frameworks, Approaches and Techniques* (DAH'09)

Paul De Bra, Mykola Pechenizkiy

Eindhoven University of Technology (TU/e), Netherlands
debra@win.tue.nl; m.pechenizkiy@tue.nl

Preface

Dynamic generation of hypertext and its adaptation and personalization to particular users is a powerful and useful concept. It is particularly helpful for the reduction of the information overload such as is frequently experienced on the Internet. 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.

Reference models and generic architectures unify a community and provide a leading generic model and/or architecture that spawns research activities in many directions. Examples of such generic models are AHAM for adaptive hypermedia and FOHM for open hypermedia. A nice example of a resulting generic implementation is the AHA! system that was last described in ACM Hypertext'06.

The research fields of hypertext and adaptive hypermedia (or adaptive web-based information systems) however, have been growing rapidly during the past ten years and this has resulted in a plethora of new terms, concepts, models and prototype systems. As a result the established models no longer include many of the recent new concepts and phenomena. In particular, open corpus adaptation, ontologies, group adaptation, social network analysis and data mining tools for adaptation are not or at least insufficiently supported.

The DAH'09 workshop[1] organized in conjunction with the 20th ACM International Conference on Hypertext and Hypermedia and held on June 29, 2009, in Turin, Italy. The workshop provides a focused international forum for researchers to discuss new developments in generic methods and technologies for dynamic and adaptive hypertext. Topics discussed during the workshop include: adaptation and personalization including such issues as open-corpus adaptation, group adaptation, higher order adaptation, and sharing of user models; adaptive and dynamic hypertext authoring (e.g. authoring conceptual adaptation models); service-oriented approaches for adaptation; use of data mining for user and domain modeling, and automatic generation of adaptation rules.

These proceedings include six accepted contributions to the workshop. We would like to thank the authors for their interest in the workshop and for submitting their contributions. Special thanks to the PC Members for their help in reviewing the submitted papers.

The first paper by Levacher *et al*. "A Framework for Content Preparation to Support Open-Corpus Adaptive Hypermedia" proposes a novel framework for open-corpus content preparation that allows processing documents existing in open and closed corpora and producing coherent conceptual sections of text with associated descriptive metadata. This work is important for enabling the mainstream adoption of AHS in web applications, which is impossible without enabling the repurposing existing content. The authors adopt state-of-the-art information extraction and structural content analysis techniques for an on-demand provision of tailored, atomic information objects.

Next contribution by van der Slijs *et al.* "GAL: A Generic Adaptation Language for describing Adaptive Hypermedia" argues that despite a large variety of personalization and adaptation features that different emerging Web applications offer, it is possible to distinguish a central and unifying objective of adaptive engines that facilitates these various features, that is, to create an adaptive navigation structure. The authors present Generic Adaptation Language (GAL) that specifies the engine independent basic adaptive navigation structure that allows using any authoring environment in combination with any adaptive engine as long as there is a corresponding GAL compiler.

The following two papers present the service-based view on the development of adaptive hypermedia. Harrigan and Wade in "Towards a Conceptual and Service-Based Adaptation Model" consider the limitations of the current practice in adaptation modeling which assume that concepts and relationships between concepts are the fundamental building blocks of any adaptive content and introduce a representation of a conceptual and service-based adaptation model. With their approach

[1] The scientific programme overview, and other workshop-related information as well as the link to the online proceedings can be found at http://www.win.tue.nl/~mpechen/conf/dah09/.

having additional expressive power it would be possible to facilitate activity-based and process-oriented adaptive applications, including the adaptation of the process itself.

Koidl *et al*. in "Non-Invasive Adaptation Service for Web-based Content Management Systems" consider the architectural and technical issues related to provision of third party adaptive services pluggable into existing web-based content management systems (WCMS) like Wiki. The authors introduce a third party Adaptive Service that also contributes to mainstreaming the adoption of AHS. This work introduces a principled way of embedding adaptive technologies within existing WCMS allowing to reduce or avoid the expense of re-engineering such systems.

The paper by Hargood *et al.* "Investigating a thematic approach to narrative generation" considers the potential, the challenges and open issues in integrating themes in narrative generation systems that attempt to generate content within a narrative or story framework.

Knutov *et al.* in "Versioning in Adaptive Hypermedia" present an approach that reuses the key concepts and ideas behind versioning and applies them to the adaptive hypermedia field. The authors illustrate with a couple of intuitive examples how such an approach helps to facilitate authoring, managing, storing, maintenance, logging and analysis of behaviour because of providing more flexibility and maintainability of the systems. Particular, versioning helps to create, maintain and re-use concurrent versions of an application or a model or a particular property and value, saving authoring effort and that is not less important facilitating provenance analysis.

Eindhoven
June 2009

Paul De Bra
Mykola Pechenizkiy

Organization

Organizing Committee

- Paul De Bra, Eindhoven University of Technology, the Netherlands
- Mykola Pechenizkiy, Eindhoven University of Technology, the Netherlands

Programme Committee

- Peter Brusilovsky, University of Pittsburg, USA
- Dominik Heckmann, DFKI, Germany
- Nicola Henze, University of Hannover, Germany
- Geert-Jan Houben, Delft University of Technology, the Netherlands
- Riccardo Mazza, SUPSI, University of Lugano, Switzerland
- David E Millard, University of Southampton, UK
- Vincent Wade, Trinity College Dublin, Ireland
- Yang Wang, University of California, Irvine, USA

Table of Contents

Killian Levacher, Eamonn Hynes, Seamus Lawless, Alex O'Connor and Vincent Wade
A Framework for Content Preparation to Support Open Corpus Adaptive Hypermedia 1

Kees van der Sluijs, Jan Hidders, Erwin Leonardi and Geert-Jan Houben
GAL: A Generic Adaptation Language for describing Adaptive Hypermedia 13

Martin Harrigan and Vincent Wade
Towards a Conceptual and Service-Based Adaptation Model 25

Kevin Koidl, Owen Conlan and Vincent Wade
Non-Invasive Adaptation Service for Web-based Content Management Systems 37

Charlie Hargood, David E. Millard and Mark J. Weal
Investigating a thematic approach to narrative generation 49

Evgeny Knutov, Paul De Bra and Mykola Pechenizkiy
Versioning in Adaptive Hypermedia 61

A Framework for Content Preparation to Support Open-Corpus Adaptive Hypermedia

Killian Levacher, Éamonn Hynes, Séamus Lawless, Alexander O'Connor, Vincent Wade

Centre for Next Generation Localisation, Knowledge & Data Engineering Group, School of Computer Science and Statistics, Trinity College, Dublin, Ireland

{Killian.Levacher, hynese, Seamus.Lawless, Alex.OConnor, Vincent.Wade}@cs.tcd.ie

Abstract. A key impediment for enabling the mainstream adoption of Adaptive Hypermedia for web applications and corporate websites is the difficulty in repurposing existing content for such delivery systems. This paper proposes a novel framework for open-corpus content preparation, making it usable for adaptive hypermedia systems. The proposed framework processes documents drawn from both open (i.e. web) and closed corpora, producing coherent conceptual sections of text with associated descriptive metadata. The solution bridges the gap between information resources and information requirements of adaptive systems by adopting state-of-the-art information extraction and structural content analysis techniques. The result is an on-demand provision of tailored, atomic information objects called "slices". The challenges associated with open corpus content reusability are addressed with the aim of improving the scalability and interoperability of adaptive systems. This paper proposes an initial architecture for such a framework in addition to reviews of associated technologies.

Key words: Open Corpus Content, Adaptive Hypermedia, Statistical Content Analysis, Structural Content Analysis

1 Introduction

The increasing pervasiveness of the internet is fundamentally changing how people author, interact with and consume content. There are early signs of a shift in the way digital content is created, from the linear authoring and publication of material towards the aggregation and re-use of existing content from various disparate sources.

Adaptive Hypermedia Systems (AHS) have traditionally attempted to deliver dynamically adapted and personalised presentations to users through the sequencing of reconfigurable pieces of information.

While the effectiveness and benefits of such systems have been proven in numerous studies [1][2], a major obstacle to their widespread adoption relates to

the restriction of suitable content available to provide such adaptivity in terms of volume, granularity, style and meta-data.

One of the key impediments in offering a wide range of information objects is, for example, the considerable amount of manual effort involved in creating these resources. Information object content consists of either pre-existing documents [3], or of content created by small groups of individuals [4], intentionally authored for a particular system [5]. This generates both scalability and interoperability issues, which prevents the mainstream adoption of adaptive systems.

Metadata standards such as LOM [6] attempt to provide generic content packaging methods with the aim of enabling the interoperability of information objects within AHSs. Although they implement usage-agnostic principles, their development is very time consuming due to the complexity involved [7]. Furthermore, they also require AHSs to comply with a specific content structure in order to make full usage of these resources.

In parallel with these developments, a wealth of information is now accessible on the web, in digital repositories and as part of library initiatives. However, due to its heterogeneity, this information is not suited to direct usage by AHSs in its present form. It is available in several languages and within large documents with very limited meta-data. These objects are also generally very coarse-grained and correlated with unnecessary noise such as navigation bars, advertisements etc. Automated content preparation techniques are therefore necessary to provide scalable solutions to AHS specific content requirements.

Our goal is to provide AHSs with the ability to benefit from the wealth of knowledge already accessible on the web and in digital libraries by bridging the gap between these information resources and the information requirements of adaptive systems. We propose the creation of a service that can tailor open-corpus content for use in AHSs without requiring such systems to adopt a generic content structure. This system would hence resolve scalability issues in content authoring by providing AHSs with a large array of information objects. Moreover, the inclusion of open corpus information would provide up-to-date content in a wide variety of styles and structures.

We aim to combine several disciplines such as Information Retrieval, Adaptive Web and Information Extraction in order to provide an on-demand information object service. This service will harvest open corpus information, segment it structurally and automatically augment it with successive layers of internal metadata. Any AHS could then provide this service with a request that includes requirements for an information object which would in turn fetch and tailor this content to its specification.

This paper is structured as follows: Section 2 will present the key challenges addressed within this research in relation to AHS information requirements as well as structural and statistical text analysis. The framework we envisage developing is presented in detail in section 3, along with the overall workflow produced. Section 4 will present the application area of such a technology and how we intend to evaluate it. Finally, section 5 will conclude and present the road

map ahead. This work is applicable to any type of content, however some of the examples presented in this paper will be based on adaptive e-learning systems.

2 Key Challenges and Related Work

Throughout the evolution of adaptive systems, technologies introduced have moved from original "all in one" solutions, towards increasing modularity whereby AHSs become consumers of domain, pedagogical and user models [8]. This has the effect of leaving such systems with a core adaptive engine capable of dealing with a wider range of loosely coupled models that are integrated as desired. Content, however, is still very tightly coupled to these engines and as a result strongly impedes the general adoption of these systems.

Most AH systems, up until recently have operated in closed document spaces with content specifically authored for their usage [4], hence obstructing interoperability, both by accepting only a narrow field of content, and by motivating the generation of content in a highly-specific format. As a result adaptive systems also encounter scalability issues due to the limited amount of content available to adapt on, arising from the small amount of manual contributions available. Open corpus content is increasingly seen as providing a solution to these issues [9]. However, most systems incorporating this type of content have, for the moment, mainly focused on linking it with internal content as alternative exploration paths. Those incorporating it fully into their system [10][11] require manual mark-up of such content with specific meta-data. Schemas such as LOM (in the area of e-learning) and IMS packagings, attempt to provide usage-agnostic solutions, however they require a lot of development effort thus prohibiting scalability [7]. Moreover, AHSs must comply with specific content structures so as to avail of these resources. In order to leverage the full potential of open corpus resources, adaptive systems need to incorporate and correlate this type of content fully into existing systems using structure-agnostic and automated approaches.

Our approach, on the other hand, moves away from the packaging model altogether by considering content as a pluggable feature of adaptive systems to the same extent as user or domain models. A service providing open corpus content automatically in a form and format customised for each requesting adaptive system decouple this feature from current systems, disentangling it from content provision and hence providing a solution to both scalability and interoperability issues.

Fully integrating information harvested over the web within adaptive presentations, on the other hand, generates new issues. Web pages are usually written as stand-alone content without any re-composition purposes. In addition to the main content, they usually present navigation bars, advertisement banners and irrelevant features from various sources that must be removed prior to proper re-use.

Structural segmentation will therefore be an initial fundamental requirement in tailoring open corpus content for use within adaptive systems. A large propor-

tion of pages harvested will need to be stripped of any redundant information that might prevent its adequate re-use in adaptive systems.

The problem of structural segmentation has already been addressed from different perspectives mainly with the purpose of tailoring existing content for different displays [12]. Most of the techniques up until recently have focused on analyzing the DOM structure of a HTML page as a basis for segmentation. Several attempts, for instance, focus on removing templates from pages by identifying common DOM sub-trees [13] or using isotonic regression [14]. Machine Learning algorithms are also used to decide which pair of tags should coincide with suitable segmentation points. However, it has become increasingly popular to separate any formatting style from within HTML tags using Cascading Style Sheets (CSS) and Javascript, thus increasing the heterogeneity of rendered content from similar HTML trees [15]. For this reason, such techniques do not appear to be an adequate solution when dealing with a large set of documents as diverse as the World Wide Web.

Vision-based techniques, using entropy reduction [12], or techniques such as VIP algorithms [16] on the other hand, partition a page based on its rendering. These have the benefit of covering a wider range of pages regardless of their HTML trees. But their usage within systems dealing with large volume of data is questionable since rendering must be performed prior to any analysis, inevitably producing delays. Moreover, they completely ignore DOM features altogether which do provide structural clues where rendered surfaces appear similar [12].

This part of our research will therefore seek to provide a solution to this paradigm that scales both in terms of processing speed as well as structural segmentation precision within a wide heterogeneous range of pages. The procedure will additionally need to incorporate the notion of granularity in order to keep cohesive chunks of text together to provide meaningful input to subsequent semantic analysis.

Within the statistical analysis component of the framework, it is important to extract concepts with high precision/recall and efficiently represent these concepts in such a way as to make them as interoperable and re-usable as possible. Brusilovosky *et al.* [9] discuss the lack of re-usability and interoperability between adaptive systems and how current adaptive hypermedia systems are restricted to closed corpora by the nature of their design. This framework architecture is based on the idea that open corpus adaptive hypermedia is feasible: the effort will be in moving away from tables of document-term frequencies towards a richer semantic representation of data that is uncomplicated, easy-to-use and highly interoperable.

3 Framework for Content Preparation

The framework proposed in this research is divided into three separate components as shown in figure 1. Each part of the framework executes a specific task on the open corpus content. A specific layer of meta-data is subsequently appended, enriching it with structural and semantic clues for further analysis at

each step. The first two stages of the pipeline are executed pre-emptively prior to any client request while the intelligent slicing task is executed at run time.

The concept of a "slice" is an abstract notion representing a stand-alone piece of information, originally part of an existing document, extracted and segmented to fulfil a specific information request. A slice can be atomic or composed of other slices. It possesses its own set of properties (or meta-data) and can inherit those of others. A slice is virtual in the sense that it only exists temporarily to fulfil a client request. It is specific to each information request and represents a subjective perspective on a particular piece of a document and its description. The degree of complexity of a slice will match the requirements of the system which requests it.

The rest of this section will present in detail each stage of the framework. The structural analysis of open corpus content will initially be described followed by the statistical analysis of these resources. Finally, the overall work-flow between this framework and AHSs clients will be outlined using intelligent slicing technology.

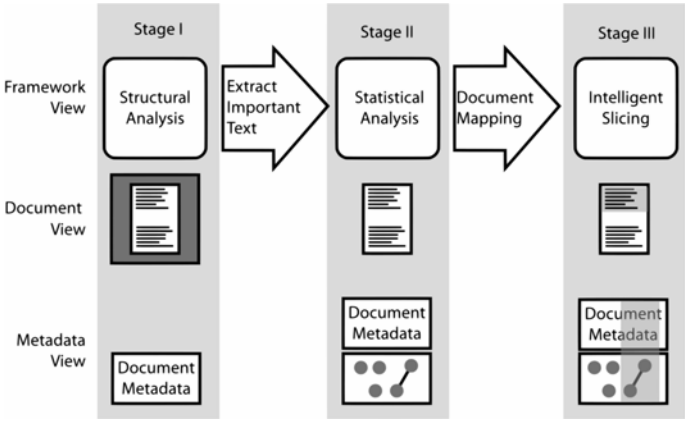

Fig. 1. Content analysis methodology

3.1 Structural Analysis

In order to provide the initial structural segmentation needed to create slices, this research will attempt to combine benefits of both DOM- and vision-based segmentation techniques. The process will initially divide pages in atomic blocks using relevant HTML tags similar to [15]. Following this extensive break down, the page will be reconstructed, aggregating relevant blocks together with similar inherent textual properties. Changes in text flow (short phrases in navigation bars vs. long sentences in text), for instance, will be considered as segment delineations.

The notion of text density, originating from the field of computer vision will also be applied to these pages using the ratio between the number of words and the space occupied on the page [15] as an additional hint for section divisions. Furthermore, the use of linguistic analysis techniques such as style boundary detection will be considered as additional boundary indication.

3.2 Statistical Analysis

In this section we discuss methods for statistically analysing text and extracting information such that it can be used as part of the wider framework; it serving as a substrate for the incorporation of several different statistical analysis techniques. In order for the proposed framework to be able to interact naturally with information seekers, it must cope with inherent ambivalence, impreciseness or even vagueness of requests and deal with them in a structured and systematic fashion. For these reasons, statistical methods offer a wealth of neat techniques for dealing with the underlying complex characteristics of language.

Work currently underway includes looking at methods for the automatic extraction of concepts from text. The vision is to extract and create a conceptual layer over the text which is amenable to efficient searching and logical reasoning. We move away from working only on closed-corpus texts and go some way to addressing the huge challenge that is the open domain problem [9] by making our algorithms as generalised as possible and not restricted to any particular domain. One method for identifying concepts from text is to use supervised learning methods. We use HMMs (Hidden Markov Models) to extract coherent, relevant passages from text [17]. This method is particularly suited to extraction from expositional texts as opposed to highly expressive literary texts (i.e. poetry, literary prose, etc.) which have high entropy vocabularies and diverse language use. Jiang *et al.* report very high precision/recall compared to other methods and their work fits in nicely with the idea of extracting concepts/coherent relevant passages of text.

The basic idea is as follows: given some *a priori* concept that is to be extracted from a text, several passages identified as being associated with that particular concept are used as training data for the HMM classifier. Once a reasonable amount of training data has been collected (enough data such that the classifier's train and test error converges towards the desired performance), the HMM can then make soft decisions on a text stream and identify coherent relevant passages of varying length with very high accuracy.

The topology of the HMM is shown in figure 2 below: the states B1, B2 and B3 are the "background" states, state E is an end state and state R is a "relevant" state. The 5 state HMM structure allows the system to emit a relevant symbol, move to a background state and then re-emit a relevant symbol at some later point before reaching the E state. The output observation probabilities can be extracted directly from the training corpora, and the objective is to learn the transition probabilities so as to maximise the likelihood of observing a relevant passage given a concept.

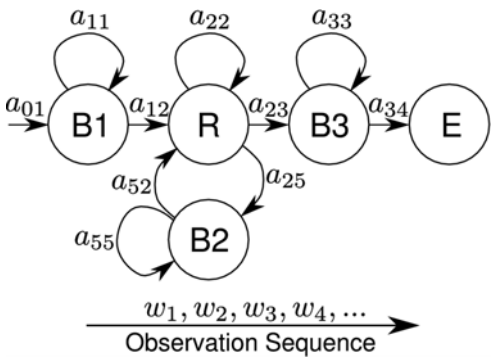

Fig. 2. HMM topology

On a synthesised corpus consisting of a number of article abstracts (coherent relevant passages) stitched together, with each abstract having an associated label, very high precision and recall values were reported [17]. It remains to be seen how well the HMM method performs on non-synthetic corpora and work is currently underway to explore this.

Several extensions to the work of Jiang *et al.* have also been explored, including the incorporation of anaphora to resolve pronouns and augment the presence of important entities such as people and places. In addition to the uni-gram lexical features, more elaborate query expansion methods and semantic analysis has been looked at to extract more elaborate features for the classifier to work from.

The incorporation of this additional functionality is expected to yield higher performance in terms of precision and recall on the synthetic document set and more reliable performance on real corpora. Instead of just a single-dimension vector with just one feature (i.e. a stemmed word), the word is represented as a slightly higher dimensional vector (i.e. the stemmed word with additional features) that helps improve the overall accuracy of the classifier.

Word Sense Clustering Another method used to extract conceptual entities from text is word sense clustering. Lexical resources such as WordNet [18] provide fine-grained sense descriptions for a huge number of words in the English language. However, such fine-grained detail is not always required: it is possible to cluster together word senses and come up with a looser word sense that abstracts away a lot of the detail and is more compatible with the framework as described in this paper. Snow *et al.* [19] describe a supervised learning method for merging word senses using a SVM-based (Support Vector Machine) clustering algorithm. This algorithm yields senses at a slightly higher level of granularity and is more compatible with corporation into a conceptual layer representation. Use of latent semantic indexing techniques is another powerful and elegant method for dealing with synonymy and polysemy whilst at the same time, automatically generating concepts, or clusters of word senses [20]. In latent semantic indexing, each document is represented as a vector of terms, with

each element of the vector having an associated weight; some function of the term frequency. Dimensionality reduction techniques are then applied to each of these document vectors, resulting in a set of vectors that point in the direction of semantically similar words. Each term in a document can then be projected onto the reduced-dimensionality semantic space, where the dominant vector(s) correspond to the concept that the term is most associated with. Probabilistic latent semantic indexing techniques [21] provide an even more accurate means of mapping from terms to concepts and such techniques will be used extensively for the statistical analysis component of the framework.

Representing content in the form of a semantic map is thought to mimic the way the human mind classifies linguistic categories. While there is no direct link between semantic maps and the representation of concepts in the human mind, it is an objective starting point for investigation of the efficiency of the semantic map representation on human cognition.

3.3 Content Preparation Work-Flow

The framework architecture presented in figure 3 belongs to the core functional unit of a web service that delivers on-demand slices to adaptive systems.

As a content provider, the framework will first of all require domain specific data gathered previously by pre-emptive crawls in areas of interest to adaptive engine consumers. We envisage as an initial step to use the OCCS harvesting system [22], developed previously, as a content gathering tool in order to bootstrap this framework with a renewable wide range of content. The system will be content harvester agnostic, hence any other content gathering system could be plugged in if required, such as private repositories for instance.

Once suitable content is available, the framework described previously will firstly remove any redundant content, then hand over only relevant information to our semantic analysis unit, and then create a conceptual map of the data. These two steps will produce a large amount of internal meta-data, both from a semantic and structural point of view.

At this stage, the service will be ready to receive content requests from a diverse range of consumers through conceptual queries which the service will attempt to map as closely as possible to the internal meta-data gathered previously on its content. The intelligent slicing unit will be responsible for matching requests to available data within the system and slicing information objects to the finest semantic grain possible matching the query. Semantic maps constructed *a priori* will be used to produce a personalised slice for the adaptive systems. Semantic technologies (OWL and SPARQL) will form the underlying mechanism navigating through pre-emptive semantic mappings.

4 Evaluation and Roadmap

This section provides a brief summary of what are the main benefits of this framework in comparison to traditional systems. The limitations of this architecture are also discussed along with possible solutions. This will result in a

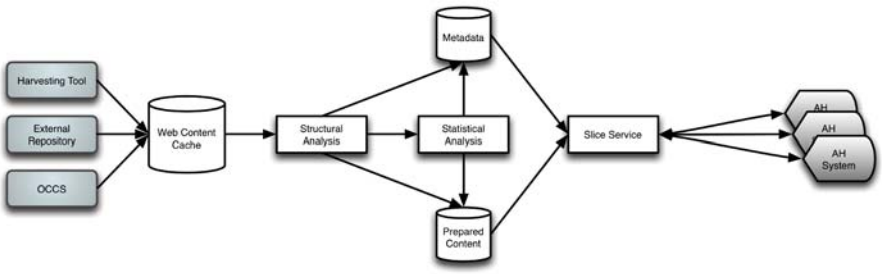

Fig. 3. Content analysis architecture

roadmap which focuses on an initial empirical evaluation of critical components, its implementation and the overall evaluation of the framework.

The content preparation framework discussed in this paper aims at bridging the gap between an increasingly large amount of open-corpus resources available on the web and AHS information requirements. The automated content slicing approach proposed offers a solution to the authorship scalability issues currently impeding the mainstream adoption of adaptive web technologies.

This method removes any content packaging interoperability issues by localising slices to the requirements of the AHS, instead of the AHS complying to a specific format. Moreover, the pipelined architecture of this framework enables the incorporation of additional plug-able meta-data generation services to the system, thus removing the need for unscalable hand-made meta-data.

However, this novel content preparation solution raises new challenges. At a component level, several technological challenges exist. To the best of our knowledge, no existing structural segmentation approach currently fulfills all of our requirements. For this reason, an initial evaluation of existing algorithms will be conducted, resulting in the combination of several techniques. Additionally, the semantic analysing step of this framework seen as the critical stage of this pipeline. An empirical evaluation of this step will therefore be needed so as to measure the quality and size of the meta-data created by this component. This process is critical since it will subsequently determine methods available to match slices with content requests. As a result, the content precision of slices generated will depend on the successive layers of the automated meta-data available. It is possible, in addition, that this precision could be improved by the use of an *a posteriori* manual crowd sourcing refinement of the slices.

As part of the Centre for Next Generation Localisation (CNGL), an initial implementation of this framework within a personalised multi-lingual customer care (PMCC) system is planned. A number of framework level evaluations will be undertaken in partnership with our industrial partners. An empirical evaluation of automatic meta-data generation performance will be conducted in order to estimate the degree of reliance towards automatic slice generation. These initial

steps will be necessary to ultimately provide a reliable foundation for the PMCC application envisaged.

5 Discussion

This paper described how restrictions in the provision of suitable content for Adaptive Hypermedia Systems currently impedes their full mainstream adoption due to interoperability and scalability authoring issues. The decoupling of content infrastructures from Adaptive Systems along with the use of automatically tailored open corpus resources was presented as a solution to this predicament.

Consequently, the architecture for a content provision service that fills the gap between open corpus information resources and Adaptive System information requirements, was outlined. To the best of our knowledge, this framework represents the first attempt to produce a pipelined, client agnostic, information preparation service that produces virtual content slices for adaptive systems. Other innovations in the framework architecture include the modular and pluggable open corpus analysers, which produce successive layers of internal metadata information. The combination of these components and how they integrate coherently to produce the framework is also an aspect under investigation.

An initial prototype of this framework is currently under development and aims towards integrating fully with a customer care information system that is currently being deployed as part of the Centre for Next Generation Localisation project.

Acknowledgements This research is supported by the Science Foundation Ireland (grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at Trinity College, Dublin. The authors wish to thank the anonymous reviewers for their comments.

References

1. Brusilovsky, P.: Adaptive Navigation Support in Educational Hypermedia: An Evaluation of the ISIS-Tutor. Journal of Computing and Information Technology (1998)
2. Hook, K.: Evaluating the utility and usability of an adaptive hypermedia system. International Conference on Intelligent User Interfaces (1997)
3. Henze, N., Nejdl, W.: Adaptivity in the KBS Hyperbook System. 2nd Workshop on Adaptive Systems and User Modeling on the WWW (1999)
4. Dieberger, A., Guzdial, M.: Coweb - Experiences with Collaborative Web Spaces. Interacting with Social Information Spaces (2002)
5. De Bra, P.: Teaching Hypertext and Hypermedia through the Web. Journal of Universal Computer Science **2** (1996)
6. IMS Global Learning Consortium: Learning Object Metadata (LOM) Information Model 1.2.2 (2009)

7. Farrell, R.G., Liburd, S.D., Thomas, J.C.: Dynamic Assembly of Learning Objects. In: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters. (2004) 162–169
8. Conlan, O., Wade, V., Bruen, C., Gargan, M.: Multi-model, Metadata Driven Approach to Adaptive Hypermedia Services for Personalized eLearning. In: Adaptive Hypermedia and Adaptive Web-Based Systems. Springer-Verlag (2002) 100–111
9. Brusilovsky, P., Henze, N.: The Adaptive Web. In: Open Corpus Adaptive Educational Hypermedia. Springer (2007) 671–696
10. Henze, N., Lenski, W., Wette-Roch, E.: Adaptation in Open Corpus Hypermedia. Journal of Artificial Intelligence in Education **12** (2001) 325–350
11. Carmona, C., Bueno, D., Guzman, E., Conejo, R.: SIGUE: Making Web Courses Adaptive. In: Adaptive Hypermedia and Adaptive Web-Based Systems. (2002)
12. Baluja, S.: Browsing on Small Screens: Recasting Web-Page Segmentation into an Efficient Machine Learning Framework. In: Proceedings of WWW-2006, ACM (2006) 33–42
13. Vieira, K., da Silva, A.S., Pinto, N., de Moura, E.S., Cavalcanti, J.M., Freire, J.: A Fast and Robust Method for Web Page Template Detection and Removal. In: Proceedings of the 15th ACM International Conference on Information and Knowledge Management, ACM (2006) 258–267
14. Chakrabarti, D., Kumar, R., Punera, K.: Page-Level Template Detection via Isotonic Smoothing. In: Proceedings of the 16th International Conference on World Wide Web, ACM (2007) 61–70
15. Kohlsch, C., Nejdl, W.: A Densitometric Approach to Web Page Segmentation. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management, ACM (2008) 1173–1182
16. Cai, D., Yu, S., Wen, J.R., Ma, W.Y.: Extracting Content Structure for Web Pages Based on Visual Representation. Springer (2003)
17. Jiang, J., Zhai, C.: Extraction of Coherent Relevant Passages using Hidden Markov Models. ACM Transactions on Information Systems (TOIS) **24** (2006) 295–319
18. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press (1998)
19. Snow, R., Prakash, S., Jurafsky, D., Ng, A.Y.: Learning to Merge Word Senses. In: EMNLP. (2007)
20. Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshaman, R.: Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science **41** (1990) 391–407
21. Hofmann, T.: Probabilistic Latent Semantic Indexing. In: 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. (1999)
22. Lawless, S., Hederman, L., Wade, V.: Occs: Enabling the Dynamic Discovery, Harvesting and Delivery of Educational Content from Open Corpus Sources. In: Eighth IEEE International Conference on Advanced Learning Technologies, IEEE Computer Society (2008) 676–678

GAL: A Generic Adaptation Language for describing Adaptive Hypermedia

Kees van der Sluijs[1], Jan Hidders[2],
Erwin Leonardi[2], Geert-Jan Houben[1,2]

[1] Eindhoven University of Technology
PO Box 513, NL-5600 MB Eindhoven, The Netherlands
k.a.m.sluijs@tue.nl
[2] Delft University of Technology
PO Box 5031, NL-2600 GA Delft, The Netherlands
{a.j.h.hidders, e.leonardi, g.j.p.m.houben}@tudelft.nl

Abstract. Nowadays, Web applications have become more sophisticated and interactive, and many of them also offer personalization and adaptation as their features. The mechanisms for these features are typically built on top of specific adaptive engine that has its own engine-specific adaptation language that can be very different from other engines. However, our observation leads to the fact that the main objective of these adaptive engines is the same, that is, to create adaptive navigation structure. In this paper, we present GAL (*Generic Adaptation Language*) that specifies the basic adaptive navigation structure in an engine independent way. Since GAL is *generic*, it can be compiled to difference specific adaptive engine and be used in combination with different authoring environments.

Keywords: GAL, generic adaptation language, navigation specification

1. Introduction

With the evolution of the Web, Web applications have grown more complex, interactive and more and more they try to adapt to the user. Personalization and adaptation are important features on the Web, as the Web allows a large and diverse public with diverse goals and demands to use the same Web applications. Personalization and adaptation are needed to help tailoring suboptimal one-size-fits-all solutions to an experience that fits a certain user in a certain context.

Adaptivity and personalization are complex mechanisms to build from scratch. Therefore, past research resulted in a number of adaptive engines and frameworks that simplify the creation of adaptive applications, like [1,2,3,4]. Also several authoring environments have been built that target the simplification of the creation of adaptive applications. If in the educational domain, for instance, a teacher should be able to create an adaptive course in such an adaptive engine, the authoring environment should allow such a lay user (in terms of adaptive systems) to easily create an adaptive application. Authoring environments also allow having different perspectives on adaptive applications, e.g. consider workflow or goal-oriented views.

A complicating factor however is that authoring environments are typically built on top of a specific adaptive engine, i.e. using a specific authoring environment restricts the specific adaptive engine that you may use and choosing a specific adaptive engine restricts the authoring environment that you can use. Moreover, we found that the different engines (even different versions of the same engines) typically have very different and sometimes complex engine specific languages that you need to use for specifying an adaptive application. However, even though the different adaptive engines have different functionalities and different strengths and weaknesses, in the end the core of these engines achieve the same thing: they create an adaptive navigation structure.

In this paper we present the Generic Adaptation Language (GAL). GAL is a language that allows expressing adaptive navigable Web applications over a given dataset. It is engine independent as it captures the *basic* adaptive navigation structure that is used by all (more specific) adaptive engines. This structure refers to the underlying dataset via queries. GAL is based on our previous experience with building Web applications in Hera [4]. However instead of specifying the whole applications design (domain, navigation, presentation), in GAL we focus on the adaptive navigation.

The primitives of the GAL language should are chosen such that they support the basic types of adaptation as specified in the field of adaptive hypermedia [5]. To show that GAL is generic we will build compilers that allow compiling GAL to several specific adaptive engines. This allows a new authoring environment to author applications for all those engines by just providing support for the GAL language as an output format. Similarly compilers will be built from several existing authoring environments to GAL, which allows a new adaptive engine to use existing authoring environments by adding support for the GAL language.

Our claim is that by using this GAL language we are able to use any authoring environment in combination with any adaptive engine (for which we write a compiler). GAL adds the additional benefit of having the code in one place in an orderly way with a simple easy to understand syntax in semantics. We will give well defined semantics, which adds additional benefits like adding model checking functionality at the GAL level, so that a GAL engine can abstract away some of the tasks of the adaptive engine.

The structure of the rest of the paper is as follows. In Section 2 we first discuss some related work. After that, we present the formal syntax and an informal semantics of GAL with a running example that presents an adaptive application about the Milky Way. In Section 3 we describe the basic concepts of GAL and how it models the navigational structure of a web site. In Section 4 we describe the global structure and semantics of a GAL program. In Section 5 we explain how in GAL the components of pages are defined. In Section 6 we describe the navigation semantics of GAL programs, i.e., what operations are executed in which order when a user navigates through the defined web application, and what are the pages presented to the user. Finally in Section 7 we present our conclusions and discuss future work.

2 Related Work

Since more than a decade ago, the Web Engineering community has proposed numerous web engineering methods (e.g. Hera [4], OOHDM [6], WebML [7], and UWE [8]) for covering complete web application development cycle. These methods try to separate Web application into three main design concerns: *data design*, *application design*, and *presentation design*. Some of these methods offer techniques for adaptation. In [4], an adaptive Web information system called Hera is presented. Hera focuses on adaptation in the navigational design. RDF is used for expressing the domain and context data, and the adaptive navigation. In addition, Hera also provides facility to use RDF query expressions in the definition of application model. This allows a more fine-grained specification of adaptation and context-dependency. OOHDM [6] supports adaptation by specifying conditions on navigation concepts and relationships in the navigation class model. These conditions determine the visibility of content and links. In WebML [7], navigational adaptation conditions are specified in a *profile* as queries. The Event-Condition-Action (ECA) rules are also applied in WebML in order to attain context-awareness. UWE [8] is an UML-based web engineering approach that uses OCL constraints on the conceptual model to specify adaptation conditions. The aspect-orientation is also applied to UWE in order to specify several types of adaptive navigation, namely, *adaptive link hiding*, *adaptive link annotation*, and *adaptive link generation*.

In the field of *adaptive hypermedia* adaptive engines are built that serve applications which are personalized for their users (which is a characteristic that they share with for example Hera). Some well known adaptive hypermedia systems include AHA! [2], InterBook [3], and APeLs [1]). AHA! is an Open Source adaptive hypermedia system mainly used in education domain. It supports adaptation in a number of different ways, namely: adaptive guiding, link annotation, link hiding, and adaptive presentation support, by using conditional fragments and objects. Interbook provides an environment for authoring and serving adaptive online textbooks. It supports adaptive navigation that guides the users in their hyperspace exploration. The guidance is visualized by using annotations (e.g. icons, fonts, and colors). Adaptive Personalized eLearning Service (APeLS) is a multi-model, metadata drive adaptive hypermedia system that separates the narrative, content, and learner into different models. The adaptive engine in APeLS is a rule-based engine that produces a model for personalized courses based on a narrative and the learner model.

The researchers in adaptive hypermedia also propose several formal models for adaptive hypermedia applications, e.g. AHAM [9], The Munich Reference Model [10], and XAHM [11]. AHAM and The Munich Reference Model extend the DEXTER model [12] by separating the storage layer further. In AHAM, this layer is separated into *domain model*, *user model*, and *adaptation model*. The Munich Reference Model divides this layer into *domain meta-model*, *user meta-model*, and *adaptation meta-model*. While AHAM follows a more database point of view, the Munich Reference Model uses an object-oriented specification written in the Unified Modeling Language (UML). The adaptation is performed using a set of adaptation rules. These models support the content-adaptation and the link-adaptation (or navigation adaptation). Cannataro et al. proposed an XML-based adaptive hypermedia model called XAHM [9]. The adaptation process in XAHM is based on three

adaptivity dimensions that form *adaptation space*, namely, user's behavior (e.g. preferences and browsing activity), technology (e.g. network bandwidth and user's terminal), and external environment (e.g. location, time, and language). The adaptation process is to find the position of the user in the adaptation space. The user's behavior and external environment dimensions determine the adaptation of page content and links. The technology dimension is used in adapting the presentation layout.

The above state-of-the-arts approaches are different from GAL in the following ways. Unlike these approaches, GAL is not intended to be another system, method, or model for designing adaptive applications. Instead, GAL focuses on abstracting from a specific adaptive engine configuration by *generically* specifying basic navigation adaptation in an engine independent way. That is, GAL can be compiled into specific adaptation engines and used by these engines. Note that GAL does not focus on typical presentation aspects like text color, font face, and how pages should be rendered to be presentable on a specific device. These aspects are typically very engine specific, and therefore hardly describable in a generic way. Moreover, GAL on purpose separates these concerns and thus limits the number of language constructs in order to get a clean and simple core language.

3 The Basic Concepts of GAL

The purpose of GAL is to describe the navigational structure of a web application and how this adapts itself to the actions of the users. The central concept is that of *unit* which is an abstract representation of a page as it is shown to a certain user after a certain request. Basically it is a hierarchical structure that contains the content and links that are to be shown, and also updates that are to be executed when the user performs certain actions such as requesting the page or leaving the page. All data from which these units are generated is assumed to be accessible via a single RDF query endpoint, and the updates are also applied via this endpoint. In the examples in this paper we will use the SPARQL[1] query language for referring to and updating data.

The set U of units contains *unordered units* and *ordered units.* These differ in that the first represents a unit with unordered content, i.e., a bag or multi-set, and the second has ordered content, i.e., a list. Unordered units are denoted formally as **uu**(C, h) where $C \in \mathbf{B}(E)$ describes the content as a bag of content elements (in the set E, to be defined later) and the relation $h \subseteq ET \times UQ$ defines the event handlers by associating event types in the set ET with update queries in the set UQ. Here we will assume that ET = {**gal:onAccess**, **gal:onExit**} and UQ describes some set of update queries over RDF stores. Ordered units are denoted as **ou**(C, h) where $C \in \mathbf{L}(E)$ describes the content as a list of content elements and the relation h is as for unordered units.

The set E of content elements contains (1) *attributes* representing presentable content such as text, (2) *unit container* representing a single nested unit, (3) *unit sets*

[1] http://www.w3.org/TR/rdf-sparql-query/

representing a nested set of units of the same type, (4) *unit lists* representing a nested list of units and (5) *links* representing navigation links. We denote attributes as **at**(*n, l*, *v*) with a name $n \in EN \cup \{\perp\}$ representing the name of the attribute, a label $l \in EL \cup \{\perp\}$ representing a printable label for the attribute and a value $v \in V$ representing the value contained in the attribute. Here *EN* denotes the set of content element names here assumed to be all URIs minus the special constants **gal:_top**, **gal:_self** and **gal:_parent**, $\perp$ a special constant not in $EN \cup EL$ denoting the absence of a name, *EL* is the set of content element labels here assumed to be the set of RDF literals and *V* the set of basic values, i.e., URIs and RDF literals. Unit containers are denoted as **uc**(*n*, *l*, *u*) with a name $n \in EN \cup \{\perp\}$, a label $l \in EL \cup \{\perp\}$ and a unit $u \in U$. The unit label *l* represents a presentable label and we assume *UL* is the set of RDF literals. The name *n* is used to refer in other parts of the unit to the position of this unit container. We denote unit sets and unit lists as **us**(*l*, *UB*) and **ul**(*l*, *UL*) with a label $l \in EL \cup \{\perp\}$ and as content a bag of units $UB \in \mathbf{B}(U)$ or a list of units $UL \in \mathbf{L}(U)$. Finally, links are denoted as **ln**(*ut*, *b*, *q*, *be*, *t*) with a unit type $ut \in UT$ that indicates the type of unit to which the link points, a binding $b : X \rightarrow V$ that maps variable names to a value and defines the static parameters of the link, a query $q \in LQ \cup \{\perp\}$ that computes dynamic parameters of the link, a binding extension function $be : X \rightarrow VE$ that maps variable names to value expressions to describe additional dynamic parameters and a target $t \in EN \cup$ {"**gal:_top**", "**gal:_self**", "**gal:_parent**"} that indicates which unit in the current page will be replaced with the retrieved page. Here *LQ* denotes the set of queries over RDF stores that return a list of bindings, *UT* denotes the set of unit type which we assume here to be URIs, *X* is the set of variable names which we assume here all start with "$" and includes the distinguished variable **$user** and *VE* denotes the set of value expressions which compute a single value given an RDF store and whose syntax will be defined in more detail later on. The set of all bindings, i.e., partial functions $b : X \rightarrow E$, is denoted *B*.

Note that the above definitions of the set of proper units *U* and the set of content elements *E* are mutually recursive, which can be resolved by stating that we define them as the smallest sets that satisfy the given definitions. Note that therefore a unit is essentially a finite tree of content elements.

4 The Global Structure of GAL Programs

We proceed with giving the syntax of GAL along with an informal description of its semantics. The purpose of a GAL program is to describe how, given a page request in the form of a unit type, a user identifier and a binding that describes the parameters, the resulting unit is computed from the current RDF store. The top level of the grammar looks as follows:

<gal-expr> ::= <unit-decl>* .
<unit-decl> ::= <unit-type> **"["** <unit-def> **"]"** .

We assume that <unit-type> refers to the set of unit type names *UT*, and <unit-def> describes the computation of a unit. For each page request the first <unit-decl> with

the requested unit type determines the result. The request is passed on to this <unit-def> while removing the unit type and user identifier and extending the binding such that **$user** is bound to the user identifier. The syntax of <unit-def>:

<unit-def> ::= **"a" ("gal:unit"** | **"gal:orderedUnit") ";"**
(<input-var>)* (<attr-def> | <subunit-expr> | <set-unit-expr> |
<list-unit-expr> | <link-expr>)* (<on-event-expr>)* **.**

First the type of the resulting unit is specified, i.e., whether it is ordered or unordered, then the list of required parameters (excluding the implicit parameter **$user**) is given, followed by a list of content element expressions, and finally a list of event handlers is given.

To illustrate the syntax we look at a use case that uses GAL to specify an adaptive application. We look at a learning application about the Milky Way. More specifically, we look at the page that specifies information about a specific planet. We then declare something like:

```
:Planet_Unit [ a gal:unit; .. {content specification} .. ]
```

The syntax of <input-var> and <on-event-expr> is defined as follows:

<input-var> ::= **"gal:hasInputVariable" "[" "gal:varName"** <var-name> **";"**
"gal:varType" <domain-concept> **"]" ";"** .
<on-event-expr> ::= **"gal:onEvent" "[" "gal:eventType" ("gal:onAccess"** | **"gal:onExit") ";"**
"gal:update" <update-query> **"]" ";"**.

Here we assume that <var-name> describes the set of variables X, and <domain-concept> describes the set of domain concepts DC and <update-query> describes the set of update queries UQ^X parameterized with variables from X. The semantics of the <unit-def> is that it first checks if the current binding indeed defines the required variables with values that belong to the specified domain concept. If this is not the case, then the result is undefined. Otherwise the binding is restricted to the specified variables plus **$user** and passed to the content element expressions. Depending on the specified unit type, we compute **uu**(C, h) or **ou**(C, h). Here C is either the concatenation of the lists of content elements computed by the specified content element expressions or the corresponding bag of content elements. For the content element expressions we assume that they produce either an empty or a singleton list. Finally, h associates event types with update queries that are obtained by taking the specified parameterized update query and replacing the variables with their value in the binding.

For example, in the running example an input parameter of the unit will indicate which planet we want to show to user. For this we assume that an incoming link to the planet unit carries this information in the form of a variable. Therefore the following will be specified in the content specification:

```
gal:hasInputVariable [
        gal:varName "Planet";
        gal:varType Planet;
];
```

5 Content Element Expressions

We proceed with the syntax of the different content element expressions, starting with attributes:

<attr-def> ::= **"gal:hasAttribute" "["** <name-spec>? <label-spec>? **"gal:value"** <val-expr> **"]"**.
<name-spec> ::= **"gal:name"** <elem-name> **";" .**
<label-spec> ::= **"gal:label"** <val-expr> **";" .**

Here <elem-name> describes the set *EN* of content element names defined here as the set of URIs not containing the special constants **gal:_self**, **gal:_parent** and **gal:top**. The semantics of the <attr-def> is the singleton attribute list [**at**(*n, l*, *v*)] with *n* the specified name (or ⊥ if it is not specified), *l* is the resulting literal of the <val-expr> (or ⊥ if it not specified or not a literal) and *v* the result of <val-expr>. If the result of <val-expr> is undefined, then the result is the empty list [].

The syntax and semantics of <val-expr> are as follows:

<val-expr> ::= <value> | "**(**" <val-expr> "**.**" <val-expr> "**)**" | "**[**" "**gal:query**" <val-query> "**]**" |
"**[**" "**gal:if**" <list-query> "**;**" ("**gal:then**" <val-expr> "**;**")? ("**gal:else**" <val-expr> "**;**")? "**]**" .

Here <value> describes the set of values *V* which we assume here to be the set of URIs and RDF literals, and <val-query> describes VQ^X, the set of parameterized queries over an RDF store that return a single value. Based on <val-expr> we define the earlier mentioned set of value expressions *VE* as the <val-expr> expressions without free variables. If the <val-expr> is <value> then this <value> is the result. If it is "**(**" <val-expr> "**.**" <val-expr> "**)**" then the result is the concatenation of the two literals that are the result of the two <val-expr>s, and if these do not both result in literals then the result is undefined. If it starts with **gal:query** then the result is that of <val-query> with the free variables substituted with their value in the current binding. If it starts with **gal:if** then the result is defined as follows. If the <list-query> with free variables replaced by their value in the current binding returns a non-empty list then the result of the **gal:then** <val-expr> is returned, otherwise the result of the **gal:else** <val-expr> is returned. If that particular <val-expr> is not present then the result is undefined.

To illustrate this we now provide some attributes in the running example to show information about this planet on the screen. For example, we want to print the name of the planet, and also provide an image of that planet:

```
gal:hasAttribute [
         gal:name planetName;
         gal:value [$Planet.name]
];
gal:hasAttribute [
         gal:name planetImage;
         gal:label ( "Image of: ".[$Planet.name] );
         gal:value [ gal:query // url of an appropriate picture
                   "SELECT ?image_url
                    WHERE { $Planet :hasAssociatedResource ?image
                                    :type Image;
                                    :url ?image_url.}"
         ]
];
```

Note the variable references for the variable named “Planet” by using the $-sign. The expression `[$Planet.name]` is a syntactic short-hand for the SPARQL query expression `[gal:query "SELECT ?name WHERE { $Planet name ?name }"]` which retrieves the name of the planet indicated by $Planet.

Now suppose we want to add an adaptive attribute that gives ‘beginner’ or ‘advanced’ information about the planet, but only if the user has advanced knowledge about the concept:

```
gal:hasAttribute [
   gal:name planetInformation;
   gal:label "Information: ";
   gal:value [
         gal:if [ gal:query
                  "SELECT  ?conceptPlanetInstance
                   WHERE { $User :hasConceptInstance ?conceptPlanetInstance
                                          :name $Planet.name;
                                          :visited ?Visited;
                                          :advancedUserValue ?Advanced.
                                    FILTER (?Visited >= ?Advanced)
                         }"];
         gal:then [$Planet.AdvancedInfo];
         gal:else [$Planet.BeginnerInfo];
   ]
]
```

The query checks if the user number of visits equals or is greater than the amount of visits to the concepts that are needed to make a user an advanced user. If this is true (and the query in the ‘if’ clause returns a result) then the advanced information is showed, otherwise the beginners information is showed.

The next type of content element we consider is the unit container:

<subunit-expr> ::= **“gal:hasSubUnit” “[”**<name-spec>? <label-spec>? <unit-constr> **“]” “;”** .

The semantics of <subunit-expr> is the singleton unit container list [**uc**(*n*, *l*, *u*)] where *l* is the unit label specified in <label-spec>, *n* is the unit name indicating the location specified in <name-spec> and *u* is the first unit in the unit list constructed by <unit-constr>. If the result of <unit-constr> is undefined then the result is the empty list []. The syntax and semantics <unit-constr> will be defined later on.

The following types of content element we describe are the list units and set units:

<list-unit-expr> ::= **“gal:hasListUnit” “[”** <label-spec>? <unit-constr> **“]” “;”** .
<set-unit-expr> ::= **“gal:hasSetUnit” “[”** <label-spec>? <unit-constr> **“]” “;”** .

The semantics of <list-unit-expr> is the list-unit list [**ul**(*l*, *LU*)] where *l* is the unit label specified in <label-spec> and *LU* is the unit list constructed by <unit-constr>. The semantics of <set-unit-expr> is the same except that *LU* is replaced with *BU*, the corresponding bag of units.

To illustrate the use of unit sets in the running example consider that we want to show a list of the moons of the planet. We can use the hasSetUnit construct for this.

```
gal:hasSetUnit[
          gal:label "The following Moon(s) rotate around ".[$Planet.name];
          gal:refersTo Moon_Unit_Short;
          gal:hasQuery "SELECT ?Moon
                        WHERE $Planet :hasMoon ?Moon";
];
```

The hasSetUnit construct executes a query and for every result of that query it will spawn a unit, in this case a Moon_Unit_Short. For every of those units the query result will be bound to the 'Moon' variable.

The final type of content element that can be specified is the link:

<link-expr> ::= **"gal:hasLink" "["** <link-constr> <target-spec>? **"]" ";"** .
<link-constr> ::= **"gal:refersTo"** <unit-type> **";"** <bind-list> .
<bind-list> ::= <query-bind>? <var-bind>* .
<query-bind> ::= **"gal:hasQuery" "["** <list-query> **"]" ";"**
<var-bind> ::= **"gal:assignVariable" "[" "gal:varName"** <var-name> **";"**
"gal:value" <val-expr> **"]" ";"** .
<target-spec> := **"gal:targetUnit"** (<unit-name> | **"gal:_top"** | **"gal:_self"** | **"gal:_parent"**) **";"** .

Here <list-query> describes the set LQ^X of parameterized queries over the RDF store that return a list of bindings and are parameterized with variables from X. The result of <link-expr> is the link list [**ln**(*ut*, *b*, *q*, *be*, *t*)] where *ut* is the unit type specified in <link-constr>, *b* is the current binding, *q* is the <list-query> but with the free variables replaced with their value in the current binding, *be* is the partial function that maps each <var-name> in the <var-bind> list to the first associated <val-expr> but with the free variables in this <val-expr> replaced as specified by the current binding, and finally *t* is the target in <target-spec> if this is specified and **gal:_top** if it is not.

For example, the Moon_Unit_Short that was mentioned in the preceding example could contain a link to the elaborate description as follows:

```
:Moon_Unit_Short [
    a gal:unit;
    gal:hasInputVariable [
              gal:varName "Moon";
              gal:varType Moon
    ];
    gal:hasAttribute [
              gal:name moonName;
              gal:label "Moon: ";
              gal:value [$Moon.name]
    ];
    gal:hasLink [
              gal:refersTo :Moon_Unit_Elaborate;
    ]
]
```

Note the hasLink construct that binds to the Moon_Unit_Short, meaning that every attribute in the unit becomes a link that points to the Moon_Unit_Elaborate unit. By default the current input variables, in this case $Moon, are passed as parameters of the link, so it indeed will lead to a page describing the same moon more elaborately.

The final non-terminal we consider is <unit-constr> which computes a list of units:

<unit-constr> ::= **"gal:refersTo" "["** <unit-def> **"]" ";"** <bind-list> .

The result is defined as follows. First, the query in <bind-list> is evaluated, while replacing its free variables with their value in the current binding, and the resulting list of bindings is extended with the binding specified by the list of <var-bind> expressions. Finally, for each of the resulting bindings we use them to extend the current binding and for this binding evaluate the <unit-def>, and finally collect all the resulting units in a list.

We extend the syntax with some syntactic sugar: inside a <unit-constr> the fragment "**[**" <unit-def> "**]**" can be replaced by a <unit-type> if in the total expression this is associated with this <unit-def>. In other words, if a <unit-type> occurs in the <unit-constr> after the **gal:refersTo** then it is interpreted as the associated "**[**" <unit-def> "**]**". This is illustrated in the preceding example for gal:hasSetUnit.

We conclude the presentation of the syntax an semantics of GAL with a possible output of the running example. Given the definitions for the Planet Unit it would yield for a specific instance, for example Jupiter, something along the lines as shown in the screenshot in Figure 1. This screenshot is based upon a GAL specification that is compiled to and executed by the AHA!3 engine [2].

Figure 1: Screenshot of an instance of the Planet Unit

6 The Navigation Semantics of GAL

In this section we describe the navigation process that is defined by a certain GAL program. At the core of the navigation semantics is the previously described computation of the unit that is the result of the request of a user. The resulting unit then is presented to the user and becomes the *current unit, i*.e., the unit that is currently presented to the user. In addition all the update queries associated with the **gal:onAcess** events in it at any nesting depth are applied to the RDF store in some arbitrary order.

If subsequently the user navigates to an external link then all the update queries associated in it with the **gal:onExit** event at any nesting depth are applied to the RDF store in some arbitrary order. If the user follows an internal link **ln**(*ut*, *b*, *q*, *be*, *t*) in the current unit, then the following happens. First the *target unit* in the current unit is determined as follows. If the target is a unit name then this is the unit in the first unit container with that name. If it is **gal:_self** then it is the containing unit, i.e., the unit in which the link is directly nested. If it is **gal:_parent** then it is the parent unit, i.e., the unit which has as a content element a unit container, unit set or unit list that refers to

the containing unit. If it is **gal:_top** then it is the root unit of the current unit. If in any of the preceding cases the result is not well defined then the target unit is the root unit of the current unit. When the target unit is determined then all **gal:onExit** events that are directly or indirectly nested in it are applied to the RDF store in some arbitrary order.

Then, the binding that describes the parameters is computed as follows: the binding *b* is combined with the first binding in the result of *q* and this in turn is combined with the binding that is obtained when evaluating *be*. The resulting binding is sent together with the specified unit type as a request. Then, if this request results in a unit, the **gal:onAccess** update queries in this unit are executed as described before. In the final step the new current unit is the old current unit but with the target unit replaced with the resulting unit of the request.

7 Conclusions and Future Work

Many Web application frameworks nowadays offer personalization and adaptation as their features. However, until now the adaptation mechanisms are typically engine specific and the adaptation languages used by the adaptive engines are different from one another. In this paper, we have presented Generic Adaptation Language (GAL) that aims to capture basic adaptive navigation functionality in an engine independent way. GAL allows us to use every authoring environment in combination with any adaptive engines. We gave a formal description of GAL, and demonstrated that GAL can be used to specify an adaptive application. Currently we work on working out examples that show that GAL allows every type of adaptivity as specified in [5]. In further future work we look at building compilers from GAL to several adaptive engines (e.g. for AHA! version 3, Hera and the GALE engine that is being built in the GRAPPLE project[2]). Similarly we will build compilers from several authoring environments to GAL (e.g. for the Graph Author and the authoring tools within the GRAPPLE project). We expect that our formal work as presented in this paper will help us to relatively simply build these compilers. In this way we aim to show that GAL is both applicable to most adaptive engines as well as that it is generic. This might pave the path in the future for general acceptance and standardization of a engine-independent generic adaptation language.

Acknowledgements: This work was supported by the 7th Framework Program European project GRAPPLE ('Generic Responsive Adaptive Personalized Learning Environment').

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

References

[1] Owen Conlan, Vincent P. Wade, Catherine Bruen, and Mark Gargan. **Multi-model, Metadata Driven Approach to Adaptive Hypermedia Services for Personalized eLearning**. In Proceedings of the Second Adaptive Hypermedia and Adaptive Web-Based Systems Conference (AH 2002), Malaga, Spain, May, 2002.

[2] Paul De Bra, David Smits, and Natalia Stash. **The Design of AHA!**. In Proceedings of the 17th ACM Conference on Hypertext and Hypermedia (Hypertext 2006), Odense, Denmark, August, 2006.

[3] Peter Brusilovsky, John Eklund, and Elmar W. Schwarz. **Web-based Education for All: A Tool for Development Adaptive Courseware**. In Computer Networks and ISDN Systems, 30(1-7), pp. 291-300, 1998.

[4] Geert-Jan Houben, Kees van der Sluijs, Peter Barna, Jeen Broekstra, Sven Casteleyn, Zoltán Fiala, and Flavius Frasincar. **Hera**. Book chapter: Web Engineering: Modelling and Implementing Web Applications, G. Rossi, O. Pastor, D. Schwabe, L. Olsina (Eds), Chapter 10, pp. 263-301, 2008, Human-Computer Interaction Series, Springer.

[5] Peter Brusilovsky. **Adaptive Hypermedia**. In User Modeling and User-Adapted Interaction, 11 (1-2), pp. 87-110, 2001.

[6] Gustavo Rossi and Daniel Schwabe. **Modeling and Implementing Web Applications with OOHDM**. Book chapter: Web Engineering: Modelling and Implementing Web Applications, G. Rossi, O. Pastor, D. Schwabe, L. Olsina (Eds), Chapter 6, pp. 109-155, 2008, Human-Computer Interaction Series, Springer.

[7] Marco Brambilla, Sara Comai, Piero Fraternali, and Maristella Matera. **Designing Web Applications with WebML and WebRatio**. Book chapter: Web Engineering: Modelling and Implementing Web Applications, G. Rossi, O. Pastor, D. Schwabe, L. Olsina (Eds), Chapter 9, pp. 221-261, 2008, Human-Computer Interaction Series, Springer.

[8] Nora Koch , Alexander Knapp, Gefei Zhang, and Hubert Baumeister. **Uml-Based Web Engineering: An Approach Based on Standard**. Book chapter: Web Engineering: Modelling and Implementing Web Applications, Gustavo Rossi, Oscar Pastor, Daniel Schwabe, and Luis Olsina (Eds), Chapter 7, pp. 157-191, 2008, Human-Computer Interaction Series, Springer.

[9] Paul De Bra, Geert-Jan Houben, and Hongjing Wu. **AHAM: A Dexter-based Reference Model for Adaptive Hypermedia**. In Proceedings of the 10th ACM Conference on Hypertext and Hypermedia (Hypertext'99), Darmstadt, Germany, February, 1999.

[10] Nora Koch and Martin Wirsing. **The Munich Reference Model for Adaptive Hypermedia Applications**. In Proceedings of the Second Adaptive Hypermedia and Adaptive Web-Based Systems Conference (AH 2002), Malaga, Spain, May, 2002.

[11] Mario Cannataro and Andrea Pugliese. **XAHM: An Adaptive Hypermedia Model Based on XML**. In Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE), Ischia, Italy, July, 2002.

[12] Frank G. Halasz and Mayer D. Schwartz. **The Dexter Hypertext Reference Model**. In Communications of the ACM (CACM), 37(2), pp. 30-39, Febuary, 1994.

Towards a Conceptual and Service-Based Adaptation Model

Martin Harrigan and Vincent Wade

Department of Computer Science, Trinity College Dublin, Ireland
{martin.harrigan, vincent.wade}@cs.tcd.ie

Abstract. Current practice in adaptation modeling assumes that concepts and relationships between concepts are the fundamental building blocks of any adaptive course or adaptive application. This assumption underlies many of the mismatches we find between the syntax of an adaptation model and the semantics of the 'real-world' entity it is trying to model, *e.g.* procedural knowledge modeled as a single concept and services or activities modeled as pockets of intelligent content. Furthermore, it results in adaptation models that are devoid of truly interactive services with workflow and dataflow between those services; it is impossible to capture the semantics of a process-oriented application, e.g. activity-based learning in education and Standard Operating Procedures (SOPs) in the workplace. To this end, we describe a representation of a conceptual and service-based adaptation model. The most significant departure from existing representations for adaptation models is the reification of services. The goal is to allow for the adaptation of the process itself and not just its constituent parts, *e.g.* an SOP can be adapted to the role or job function of a user. This expressive power will address the mismatches identified above and allow for activity-based and process-oriented adaptive applications.

1 Introduction

The cornerstone of an authoring tool-set for an adaptive course or adaptive application is the adaptation model. It is the specification of the adaptive storyline or narrative [1] that guides each user through a course or application. What makes the adaptation model so difficult to author, is the very fact that there are so many different paths each user can follow. A specification of an adaptation model serves two functions. Firstly, it determines (or is determined by) what is expressible within an authoring tool-set [1, 2, 3, 4, 5]. Secondly, it hides the intricacies of the rule-based language or low-level instruction-set of an adaptive engine, e.g. AHA! [2]. However, a problem with current adaptation models is that they are unduly focused on concepts and relationships between concepts [6, 7, 8]. Procedural knowledge, interactive services and activities are difficult, if not impossible, to model. Adaptation models require greater expressive power in order to adequately model these possibilities.

To better illustrate this short-coming, consider adaptation models within an educational or learning environment. Activity is an important part of learning. For example,

This work was performed within the GRAPPLE Project. GRAPPLE (Generic Responsive Adaptive Personalized Learning Environment) is an EU FP7 STREP project that aims to deliver a *Technology-Enhanced Learning* (TEL) environment that guides learners through a learning experience, automatically adapting to personal preferences, prior knowledge, skills and competences, learning goals, and the personal or social context in which the learning takes place. This functionality will be provided as a set of adaptive learning services that will be integrated with existing Open Source and commercial *Learning Management Systems* (LMSs).

Active Learning [9] is a model of instruction that places the responsibility of learning on learners through practical activity. In order to assimilate the learning material, the learners must actively engage with it. Examples of active learning include class discussion and 'think-pair-share'. Situated Learning [10] states that learning is a function of activity, context and the culture in which it takes place. It occurs as a result of social interaction, *e.g.* through workshops, role-playing, field trips, etc. Many other learning theories are bound to the notion of an activity. At the same time, learners learn best when the learning experience offered to them meets their needs and stimulates their preferred modes of learning [11]. When creating an adaptation model for such an environment, we should consider not only adaptive content and adaptive navigation, but also adaptive learning activities [12, 13]. We need to shift from learning objects and content that are 'retrieved' or 'accessed' to learning activities that are 'experienced'.

Our conceptual and service-based adaptation model is novel in the sense that it treats services as first-class citizens. Services and workflow between those services are explicitly modeled. The goal is to provide for the adaptation of both concepts and services, *i.e.* adaptive content, adaptive navigation, adaptive services, and adaptive workflow. This is a fusion of the predominantly concept-centric models of adaptive hypermedia and the service-centric models of workflow and process-oriented systems. In order to accomplish this, we generalize the notion of a *Conceptual Relationship Type* (CRT) [14] to an *Abstract Relationship Type* (ART) to cater for both.

This paper describes an adaptation model that explicitly provides for interactive services and activities. It does not consider the theoretical or pedagogical concerns when designing an actual adaptive course or adaptive application, nor does it detail a user interface of an actual authoring tool-set. At the same time, we note that this work is undertaken within a suitable context (see earlier footnote). The paper is organized as follows. In Sect. 2 we provide definitions for commonly used terms. In Sect. 3 we enumerate the broad requirements for a conceptual and service-based adaptation model. The syntax and semantics of the abstract relationship type and adaptation model are detailed in Sect. 4. This is followed by their usage in Sect. 5. We conclude and consider future work in Sect. 6.

2 Definitions

This section establishes definitions for commonly used terms in the paper.

A *Resource Model* (RM) [15] models all assets (concept instances and service instances) that are available for use. These assets comprise text, images, sound, video, chat services, email services, forum services, exercises, references, datasets, etc. The actual assets can be contained in a closed-corpus repository or retrieved from an open-corpus repository, such as the Web. However, every modeled asset must provide meta-data in the form of a required set of controlled attributes. These attributes can specify, for instance, the difficulty of the content (e.g. introductory, intermediate, expert), the language, the media format, etc. Service instances also need to expose their inputs and outputs.

A *Domain Model* (DM) [6, 15] comprises a *Concept Domain Model* (CDM) and a *Service Domain Model* (SDM). A CDM models abstract concepts and relationships between them. The actual concept instances are modeled in an RM. Concept spaces, taxonomies, ontologies, and topic maps are all considered to be DMs. An SDM is an analogous model for services. It models abstract services and relationships between them. The actual service instances are modeled in an RM. Services can be domain-specific in their very nature whereas other services can be domain-independent but can be parameterized so that they

appear domain-specific, *e.g.* 'classDiscussion' can apply to any domain, but when the discussion is parameterized with a particular topic, it becomes domain-specific.

An *Abstract Relationship Type* (ART) is a relationship that can be instantiated between concepts, between services, or between concepts and services and provides for adaptive behavior. It is a generalization of a *Conceptual Relationship Type* (CRT) [14], in that its parameters can be concepts or services. An authoring tool-set will provide a toolbox or palette of commonly used ARTs. ARTs are distinct from the relationships in a DM; ARTs provide for adaptive behavior, DM relationships do not. However, in some cases ARTs can be derived from relationships in a DM, allowing DM relationships to be mapped to particular ARTs. This is explored in Sect. 4.1 when discussing the `derivedFrom` element.

A *conceptual and service-based Adaptation Model* (AM) is an assembly of concepts and services retrieved from a DM, with relationships between them, a workflow specification guiding the execution of the services, and provisions for adaptive content, navigation, service selection, workflow, and presentation. The relationships are instantiated ARTs whose parameters are bound to the assembled concepts and services.

A *User Model* (UM) [6] is a store of *attributes* describing individual users and groups of users; it permits software to adapt to their characteristics. It is sometimes useful to distinguish between a *UM schema* – how the UM is logically structured, and the actual data stored in a particular UM. A simple *overlay UM* stores, for each concept (resp. service) in a DM, a user's knowledge or familiarity with that concept (resp. service).

A *strategy* is a combination of ARTs that can only be instantiated within some predefined pattern. It allows for the aggregation of ARTs in order to implement, for example, quality control procedures, WebQuests, the Jigsaw learning technique, etc. The predefined pattern can permit a certain degree of customization, *e.g.* not all ARTs are mandatory. This is explored in Sect. 4.1 when discussing the `preConditions` and `postConditions` elements.

This proliferation of models does not lead to undue complexity [16]; in fact, it promotes separation of concerns and allows, for example, DM experts to focus on DMs and educational technologists and instructional designers to focus on ARTs and CRTs of a pedagogical nature. In some cases, they may need to collaborate.

In the following sections, we refer to an example from the educational or learning domain based on the student peer review process. This example is presented in detail in Sect. 5

3 Requirements

In this section we enumerate some broad requirements for a conceptual and service-based AM. These are based on recent interviews with potential users of an Adaptive Learning System [17, 18] (an important application area for conceptual and service-based adaptation models) and on previous experience with such models [12, 19, 1, 20, 21].

1. Concepts and services should be seamlessly integrated. Early approaches [21] provide distributed, intelligent services that are embedded in concepts, that is, the concepts contain the services. However, this does not support any form of information flow between those services. The services amount to isolated, disconnected applets.
2. The representation should allow an author to create, manipulate, delete, and organize all aspects of an AM. A canvas or workspace, into which an author can 'drag-and-drop' graphical representations of concepts, services and their relationships, supports the inherently dynamic and creative process of designing an AM. The representation should also include the spatial layout of an AM. Although this information does not

concern the semantics of an AM, it can be suggestive to an author of the directionality and temporal sequence of the model.

3. The entities and relationships that an author has in mind when creating an AM (introductory explanations, exercises, online discussions, prerequisites, roles etc.) should be the same entities and relationships that they are controlling and manipulating in the representation [22]. The model needs to operate at an appropriate level of abstraction. In practical terms, we would like an author to create, for example, an AM that describes the student peer review process by assembling concepts and services like 'introductoryResource', 'reviewResources', 'discussion', and 'writeReport' and instantiating relationships like 'prerequisite' and 'followedBy'. The service 'discussion' may be realized in any number of ways depending on what service instances or resources are available. However, when creating an AM, it is more appropriate for an author to manipulate graphical artifacts entitled `discussion`. Similarly, an encapsulated relationship entitled 'prerequisite' is more intuitive to an author than the adaptive behavior (adaptive sequencing and presentation [23, 24]) it embodies.
4. Adaptive behavior should be abstracted and refactored [25] into reusable, flexible relationships. Concepts and services can be reused in many different AMs. The same should be true for relationships like 'prerequisite', 'analogousTo', 'conformsTo', 'isSometimesFollowedBy', etc. This is a noticeable departure from existing representations and authoring tool-sets [7] that mix classical semantic relationships like 'is-a' and 'has-a' with application relationships like 'prerequisite' and 'analogousTo' which also embody adaptive behavior.
5. Practitioners appreciate the complexity and difficulty in authoring an AM [17, 18]. This is the 'price of adaptivity' [7]. However, they also seek an authoring tool-set that can produce AMs without the intervention of specialists. To overcome this difficulty, the tool-set should provide as much advice and support as possible, e.g. during author-time, the act of instantiating a relationship in an AM should only be allowed under certain 'terms-and-conditions'. This intelligence on the part of the tool-set is only possible if the representation for each relationship includes such conditions. However, the benefit is that the author can be advised at an early stage when something is askew.

These requirements, namely the seamless integration of concepts and services (R1), the 'mix-and-match' paradigm (R2), an appropriate level of abstraction (R3), the abstraction of reusable adaptive behavior (R4), and as much advice and support at author-time as possible (R5) underlie many of the decisions in the following section.

4 Syntax and Semantics

In this section we present the syntax and semantics of the ART and the AM. ARTs are instantiated in an AM so it is impossible to present one without the other. Fig. 1 summarizes their content. The ART schema comprises six elements described below. The AM schema comprises five elements also described below. An AM and a DM are linked through the AM's `entities` element: it refers to concepts and services in the DM. An AM and an ART are linked through the AM's `relationships` element: it contains instantiations of ARTs.

4.1 The Abstract Relationship Type (ART)

ARTs are the glue that bind together the concepts and services in an AM. They are also wrappers around the specification of adaptive behavior; they guide the author to the

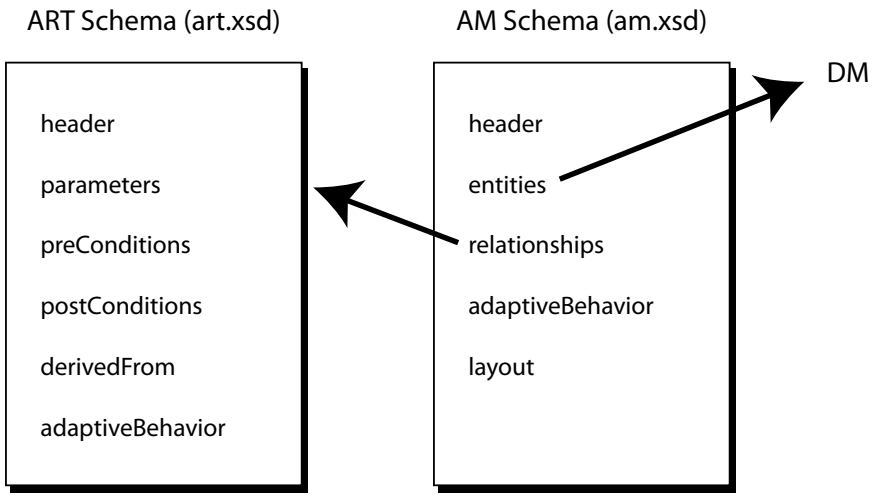

Fig. 1. The ART and AM representations are described by XML Schema.

appropriate places in which to include this behavior. If more than one end point of an ART is a service, then it also controls the workflow between those services, *i.e.* sequencing, branching, merging, etc. By refactoring [25] the adaptive behavior into reusable ARTs, we hope to improve the non-functional properties of an AM; namely its readability, maintainability, and extendability. An authoring tool-set should provide an author with a toolbox of commonly used ARTs. If an authoring tool-set is pedagogically-oriented, *i.e.* responsible for producing adaptive courses in a learning environment, then the ARTs should be pedagogical in nature, *e.g.* 'prerequisite', 'analogous-to', 'criticism-of', etc. However, these are by no means the only type of ART we envision. For example, if an authoring tool-set is responsible for producing *Standard Operating Procedures* (SOPs) for a workplace, then the ARTs should cater for quality-control, *e.g.* 'conforms-to', 'requires-accreditation', etc. The author may also need to tweak existing ARTs or create new ARTs from scratch. Each ART comprises `header`, `parameters`, `preConditions`, `postConditions`, `derivedFrom`, and `adaptiveBehavior` elements. These are specified using an XML Schema [26] and are explained in the following subsections.

The `header` Element The `header` identifies an ART (`identifier`, `author`, `name`, `description`, `keywords`) and provides rudimentary support for versioning (`majorVersion`, `minorVersion`) and permissions (`permissions`). `identifier` is a *Universally Unique IDentifier* (UUID) [27] allowing each authoring tool-set to uniquely identify information without significant central coordination. Authors are identified in the same way. `name`, `description` and `keywords` are intended to help an author decide when and where an ART is appropriate. The version number is separated into two fields, `majorVersion` and `minorVersion`. The former is initialized to one and is incremented when the 'interface' to an ART changes, *i.e.* the number or types of parameters (specified below) are changed. The latter is initialized to zero and is incremented for every other change in an ART. It is reset to zero when `majorVersion` is incremented. An instantiated ART can be updated to a new minor version automatically. However, an instantiated ART can be updated to a new major version only through the intervention of an author. `created` and `lastUpdated`

are stored in Coordinated Universal Time (UTC) but can be translated to and from local times by an authoring tool-set. The `permissions` element determines the access rights to an ART. It follows a Unix-like mechanism. ARTs are managed using three distinct classes: `author`, `group`, and `world`. Each can have `read`, `write`, and/or `publish` permissions. There are, of course, many alternatives, including role-based access control [28]. It is important to note that the model does not enforce these permissions but only allows for their representation. It is left to an authoring tool-set or a local or remote persistence service to enforce the rules.

The `parameters` Element An ART is instantiated between concepts and services in an AM. The `parameters` element specifies the number and types of concepts and services that it can be bound to. Each `parameter` is grouped under one of three distinct classes: `sources`, `targets`, and `undirected`. The first comprises parameters that are 'antecendents' of an ART; the second comprises parameters that are 'consequents' of an ART. For example, a 'prerequisite' ART might describe entities $X_1, X_2, \ldots, X_s$ as being prerequisites for entities $Y_1, Y_2, \ldots, Y_t$. $X_1, X_2, \ldots, X_s$ are classed as `sources` and $Y_1, Y_2, \ldots, Y_t$ are classed as `targets`. The third class, `undirected`, comprises parameters that take part in an undirected ART. For example, a 'similar-to' ART might describe an entity X as being similar to an entity Y. If Y must also be similar to X then both X and Y are classed as `undirected`.

Each `parameter` has its own `identifier` (UUID), `name`, `type`, `minOccurs`, and `maxOccurs` elements. `type` can have one of four values: `concept`, `service`, or `either`. They specify whether the parameter can be bound to a concept, service or either. The `minOccurs` and `maxOccurs` elements afford several possibilities including mandatory (`minOccurs` $= 1$), optional (`minOccurs` $= 0$), and unbounded (`minOccurs` $= 1$ and `maxOccurs` $= -1$) parameters. An ART must contain at least one `parameter`. An ART with exactly one `parameter` can be considered a tag or loop. It adds adaptive behavior to single a entity.

The `preConditions` and `postConditions` Elements The `preConditions` and `postConditions` elements are the 'terms-and-conditions' of instantiating an ART in an AM. The `preConditions` must hold before an ART is instantiated and the `postConditions` must hold after. All conditions are structural and can be checked at author-time. They provide an authoring tool-set with some intelligence to guide an author. We enumerate and describe the conditions which can appear within either the `preConditions` or `postConditions` tags:

1. `minDegree`: This is the minimum number of relationships that an entity bound to a parameter of an ART can take part in.
2. `maxDegree`: This is the maximum number of relationships that an entity bound to a parameter of an ART can take part in.
3. `directedCycle`: This determines whether instantiations of an ART can induce a directed cycle. For example, it should not be possible for instantiations of a 'prerequisite' ART to induce a directed cycle: X is a prerequisite for Y, Y is a prerequisite for Z and Z is a prerequisite for X.
4. `isParameterOfART`: This constrains an entity bound to a parameter of an ART into being bound to a parameter of some other ART. This forces certain ARTs to be instantiated in sequences or in combination with other ARTs, thus providing an implementation of our earlier definition of a strategy (see Sect. 2).

The `derivedFrom` Element An author creates an AM by assembling collections of concepts and services and instantiating ARTs between them. An authoring tool-set should assist the author in this endeavor. Consider the case where an author copies two entities, X and Y, from a DM into an AM and there already exists a DM relationship between them, *e.g.* X 'is-a' Y. It may sometimes be desirable and expedient for ARTs to be instantiated automatically or semi-automatically (through a dialogue with the author) in response to this, *e.g.* Y is a prerequisite of X can be derived from the fact that X 'is-a' Y. The derivations are specified within the `derivedFrom` element. This approach differentiates between user-agnostic DM relationships that are present in concept spaces, taxonomies, ontologies, and topic maps and ARTs that have adaptive behavior and references to UM attributes associated with them while allowing the latter to be automatically derived from the former. This allows, for example, the automatic derivation of adaptive navigation structures, e.g. concept-based hyperspaces [7].

The `adaptiveBehavior` Element The final element, `adaptiveBehavior`, specifies the adaptive behavior of an ART using an appropriate adaptation language [29, 30] within a single CDATA block. This language should be independent from the DM, adaptive engine, and target platform. It can reference the parameters from the `parameters` element above and can query and update the attributes of the UM. At publish-time (when the AM is translated to the low-level instruction-set of an adaptive engine), references to parameters are substituted with the identifiers of the entities they are bound to.

4.2 The Conceptual and Service-Based Adaptation Model (AM)

The AM is a model into which concepts and services are assembled and ARTs are instantiated. It is a mapping between the 'knowledge space' and what is eventually presented to a user [7]. An author creates an AM in a declarative fashion; he or she declares the high-level structure of an AM. The entities and relationships that the author has in mind when preparing an AM are the same artifacts that he or she is controlling and manipulating in the workspace, *e.g.* an introductory explanation, exercise, online discussion, etc. The adaptive behavior of each ART and an AM as a whole (also specified in an adaptation language) provide an imperative list of actions that realize the author's intent. Each AM comprises `header`, `entities`, `relationships`, `adaptiveBehavior`, and `layout` elements. These are specified using an XML Schema [26] and are explained in the following subsections.

The `header` Element The `header` element is identical to the `header` of an ART; it identifies an AM (`identifier`, `author`, `name`, `description`, `keywords`) and provides rudimentary support for versioning (`majorVersion`, `minorVersion`) and permissions (`permissions`).

The `entities` Element The building blocks of an AM are its `entities`. Each `entity` can be either a concept or a service from a DM and is identified through an `identifier`. It references the `identifier` and `version` of the concept or service (if they exist). Each `entity` also includes a list of `controlledAttributes` along with values. These place further restrictions on the actual concept or service instance that can be retrieved for each entity when an AM is published. For example, suppose an author includes a service from a DM entitled 'discussion'. However, suppose also that there are a number of different service instances or implementations that offer this functionality but vary in form. For example, in one instance, an instant-messaging portlet is provided, in another, a discussion

forum or bulletin board system is used. One standardized possibility for the controlled list of attributes the Dublin Core [31] elements. In the case of 'discussion', an attribute named 'duration' with a value of 'short' would restrict the actual service instances that can be retrieved for this entity to those that persist for a short period of time.

The `relationships` Element The `relationships` element is the container for instantiated ARTs. Each instantiated ART is identified through an `identifier`. It also references the `identifier` and `version` of an ART it is an instantiation of. It maps the `parameter`s of an ART to the `entity`s of an AM. This mapping determines the entities that the instantiated ART is gluing together. For example, suppose a 'conforms-to' ART has two parameters: a service X (source parameter) and a concept Y (target parameter). When an author instantiates this ART between, say, an interactive 'generalLedger' service and the 'doubleEntryBookkeeping' concept, X is bound to 'generalLedger' and Y is bound to 'doubleEntryBookkeeping'. This is identical to the binding of arguments to function parameters in programming languages. An authoring tool-set can ensure at author-time that the `type`, `minOccurs` and `maxOccurs` children of each `parameter` element of each instantiated ART are satisfied. Furthermore, it can also ensure at author-time that the `preConditions` and `postConditions` of each instantiated ART are satisfied. An instantiated ART must be bound to at least one `entity`. An instantiated ART that is bound to exactly one `entity` can be considered a tag or loop. It adds adaptive behavior to a single entity.

The `adaptiveBehavior` Element The `adaptiveBehavior` element specifies the adaptive behavior of an AM using an appropriate adaptation language [29, 30] within a single CDATA block. This behavior is independent of any of the individually instantiated ARTs. It can reference the individual `entity` and `relationship` elements above and can query and update the attributes of the UM. For example, it could save and retrieve the last known position of a user within the structure of an AM or it could change the role or qualifications of a user at the appropriate time.

The `layout` Element Finally, the `layout` element provides a graphical representation of an AM. This representation is based on a *graph model*. A *graph* is a mathematical structure typically used to model the relationships between objects. It consists of *nodes* or *vertices* linked together by (possibly *directed*, *weighted*) *edges*. A graph can be defined by a list of vertices and edges, by using a matrix structure or simply by a *drawing* of the graph. A drawing of a graph is a visual representation of its vertices and edges. We describe one such representation. A polygonal or circular shape depicts a vertex with an optional label printed inside or close to the shape. An edge is depicted by a polyline or curve which connects the shapes and may also have an optional label. If an edge has an associated direction it is decorated with an arrow to indicate this direction. A 'good' drawing of a graph not only defines the graph but also effectively aids its comprehension by authors. Our graph drawings are also subject to *constraints* [32] that reflect the domain of AMs, *e.g.* their layout can be suggestive of the directionality and temporal sequence of the model. Graphs provide a uniform interface over heterogeneous models with standard graphical operations and easily-understood semantics. The same interface can be used for both creating and manipulating existing AMs. Indeed, we expect high-quality AMs to evolve rather than be completely designed from the ground up. Therefore, the interface must cater for both creation and maintenance.

A vertex can represent either a concept or a service. An edge represents an ART. Any subgraph can be contracted to a single vertex. It can be expanded again as necessary.

An entire graph represents an AM. An author creates an AM by dragging vertices and edges from a palette and adding them to a graph. For example, adding a vertex that represents some concept and linking it with an edge to some service indicates that the concept and the service are related. The service might be a realization of the concept or it might conform to the principles of the concept, etc. The type of relationship will be indicated by the type and labeling of the edge. Vertices and edges can also be removed. The actual drawing or layout of a graph does not concern the semantics of an AM. It is purely an aid for the author when comprehending the structure and contents of an AM. The interface enforces constraints during author-time. For example, when dragging an ART from a palette and adding it to a graph, the edge can only be added between vertices whose type matches those of the ART's parameters.

We use GraphML [33] to store our layout data. Our AM schema imports the GraphML schema and the contents of the `layout` element adhere to this schema. GraphML can also be extended using either attributes to attach scalar values or key/data pairs to attach structured data to a graph. This reduces the burden on us, allowing us to focus solely on the representation of an AM and not on the graphical and topological aspects.

5 Example Usage

The LADiE project [34] has developed a broad set of use case scenarios for activity-based e-Learning. One such scenario requires students to review a set of resources relating to a specific topic. This review is followed by a discussion on the topic, which is guided by the teacher. The students then write and submit a report based on their discussion. There is also the possibility that a student can re-examine the resources, and discuss the problem while writing the report at the same time. The list of steps is:

1. Teacher briefs students on the activity.
2. Students log into the system and access the resources.
3. Students discuss the problem.
4. Students write report.
5. Students submit report.
6. System notifies teacher that report has been submitted.

We wish to adapt both the content and the workflow of these activities to the needs of the students. The content can be adapted based on various adaptive axes such as the students' prior knowledge. In addition to this, the services can be tailored to the needs of the students as well as to their context. For example, discussion between groups of students can be supported by various different services. A bulletin board service might be appropriate if the activity is expected to run over a long period of time while a chat room service might be more appropriate for a relatively short activity where all of the students are online. Similarly, the writing and submission tasks can employ a choice of services.

For the purpose of this example, we assume that the topic is *Digital Rights Management* (DRM) and that the author has access to a DM containing concepts like 'DRM', 'DRMTechnologies', and 'DRMLegislation' and services like 'reviewResources', 'discussion', 'writeReport', and 'submitReport'. The author drags each of these entities into an AM. He instantiates a 'prerequisite' ART between 'DRM' and 'DRMTechnologies' and between 'DRM' and 'DRMLegislation'. He instantiates a 'isLegalIssue' ART whose only parameter is 'DRMLegislation'. This ensures that the 'DRMLegislation' concept is only presented to the class if they have an interest in law. The author instantiates an 'and-Split' ART between 'DRMTechnologies' and the services 'reviewResources', 'discussion'

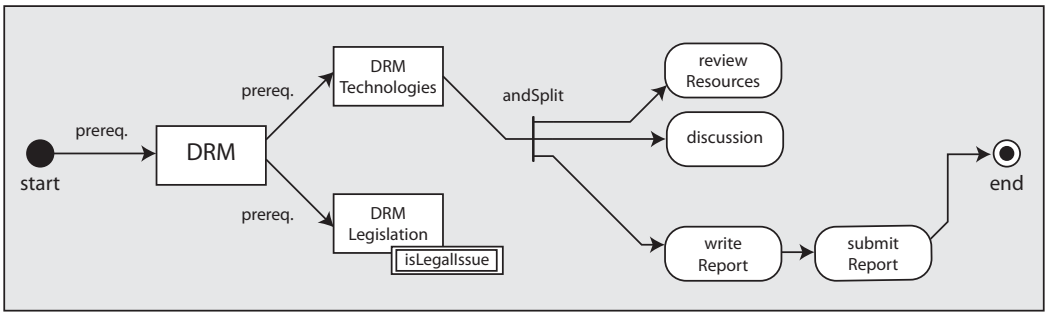

Fig. 2. The student peer review process.

and 'writeReport'. This indicates that once a student has studied the 'DRMTechnologies' concept he or she can perform the next three services in parallel, switching between them as needed. These may be presented to the user as three separate windows or portlets. Of course, it is altogether possible that an instantiation of an 'andSplit' ART is not only distinguished by its label but also by a unique graphical depiction –one that is similar to 'split' constructs in visual workflow languages. Finally, the author instantiates a 'sequence' ART between the 'writeReport' and the 'submitReport' services. The outcome is illustrated in Fig 2.

6 Conclusions and Future Work

We have presented the basic ingredients of an AM that supports the adaptive selection, sequencing and presentation of both concepts and services. By putting services on an equal footing with concepts, we allow for adaptive service selection and adaptive sequencing of the services themselves. We have generalized the notion of CRTs [14] to cater for these new possibilities. Within the context of the GRAPPLE Project (see earlier footnote), we are developing a graph-based authoring tool-set that will be based on this AM. The tool-set will comprise a *Rich Internet Application* (RIA) that will enable authors, in particular teachers, educational technologists and instructional designers, to compose AMs representing adaptive courses and adaptive procedural simulations. Server-side functionality will enable translation of this AM to the rule-based languages or low-level instruction-sets of an adaptive engine, e.g. AHA! [2].

There are many possible extensions to the AM. For example, some DMs use a frame-like knowledge representation, *i.e.* they model the internal structure of each concept or service. Instead of assembling entire concepts or services in an AM, an author may want to mix and match parts of different concepts and services. Also, we believe that the analogy between ARTs and functions in programming languages, mentioned in Sect. 4.2, is particularly apt. We are considering the benefits of inheritance and overloading when applied to ARTs.

References

[1] Dagger, D., Wade, V., Conlan, O.: Personalisation for All: Making Adaptive Course Composition Easy. Educational Technology and Society **8**(3) (2005) 9–25

[2] De Bra, P., Smits, D., Stash, N.: Creating and Delivering Adaptive Courses with AHA! In Nejdl, W., Tochtermann, K., eds.: Proceedings of the 1st European Conference on Technology Enhanced Learning (EC-TEL'06), Springer (2006) 21–33

[3] Specht, M., Kravčík, M., Klemke, R., Pesin, L., Hüttenhain, R.: Adaptive learning environment for teaching and learning in winds. In: Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'02), Springer (2002) 572–575

[4] Kravčík, M., Specht, M., Oppermann, R.: Evaluation of WINDS Authoring Environment. In: Proceedings of the 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'04), Springer (2004) 166–175

[5] Cristea, A., de Mooij, A.: Adaptive Course Authoring: My Online Teacher. In: Proceedings of the 10th International Conference on Telecommunications (ICT'03). (2003) 1762–1769

[6] De Bra, P., Houben, G., Wu, H.: AHAM: A Dexter-Based Reference Model for Adaptive Hypermedia. In: Proceedings of the 10th ACM Conference on Hypertext and Hypermedia (HYPERTEXT'99), ACM (1999) 147–156

[7] Brusilovsky, P.: Developing Adaptive Educational Hypermedia Systems: From Design Models to Authoring Tools. In Murray, T., Blessing, S., Ainsworth, S., eds.: Authoring Tools for Advanced Technology Learning Environments. Kluwer Academic Publishers (2003) 377–409

[8] Cristea, A., Smits, D., De Bra, P.: Towards a Generic Adaptive Hypermedia Platform: A Conversion Case Study. Journal of Digital Information **8**(3) (2007)

[9] Bonwell, C., Eison, J.: Active Learning: Creating Excitement in the Classroom. AEHE-ERIC Higher Education Report 1, Washington, D.C. (1991)

[10] Lave, J., Wenger, E.: Situated Learning: Legitimate Peripheral Participation. 1st edn. Cambridge University Press (1991)

[11] Conlan, O., Hockemeyer, C., Wade, V., Albert, D., Gargan, M.: An Architecture for Integrating Adaptive Hypermedia Services with Open Learning Environments. In Barker, P., Rebelsky, S., eds.: Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA'02), AACE (2002) 344–350

[12] Conlan, O., O'Keeffe, I., Brady, A., Wade, V.: Principles for Designing Activity-Based Personalized eLearning. In Spector, J., Sampson, D., Okamoto, T., Kinshuk, Cerri, S., Ueno, M., Kashihara, A., eds.: Proceedings of the 7th IEEE International Conference on Advanced Learning Technologies (ICALT'07), IEEE Computer Society (2007) 642–644

[13] O' Keeffe, I., Brady, A., Conlan, O., Wade, V.: Just-In-Time Generation of Pedagogically Sound, Context Sensitive Personalized Learning Experiences. International Journal on E-Learning **5**(1) (2006) 113–127

[14] De Bra, P., Aerts, A., Rousseau, B.: Concept Relationship Types for AHA! 2.0. In Richards, G., ed.: Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (E-Learn'02), AACE (2002) 1386–1389

[15] Conlan, O., Wade, V., Bruen, C., Gargan, M.: Multi-Model, Metadata Driven Approach to Adaptive Hypermedia Services for Personalized eLearning. In De Bra, P., Brusilovsky, P., Conejo, R., eds.: Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'02), Springer (2002) 100–111

[16] Conlan, O., Wade, V.: Evaluation of APeLS - An Adaptive eLearning Service Based on the Multi-Model, Metadata-Driven Approach. In De Bra, P., Nejdl, W., eds.: Pro-

ceedings of the 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'04), Springer (2004) 291–295
[17] Harrigan, M., Kravčík, M., Steiner, C., Wade, V.: What Do Academic Users Want from an Adaptive Learning System? Technical Report, Trinity College Dublin (2009) https://www.cs.tcd.ie/publications/tech-reports/reports.09/TCD-CS-2009-06.pdf.
[18] Harrigan, M., Kravčík, M., Steiner, C., Wade, V.: What Do Academic Users Really Want from an Adaptive Learning System? In: Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization (UMAP'09) (to appear), Springer (2009)
[19] O'Keeffe, I., Conlan, O., Wade, V.: A Unified Approach to Adaptive Hypermedia Personalisation and Adaptive Service Composition. In Wade, V., Ashman, H., Smyth, B., eds.: Proceedings of the 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'06), Springer (2006) 303–307
[20] Wilson, S., Blinco, K., Rehak, D.: An e-Learning Framework: A Summary. In: Proceedings of the Advancing Learning Technology Interoperability Lab Conference (ALT-I-LAB'04). (2004)
[21] Brusilovsky, P.: KnowledgeTree: A Distributed Architecture for Adaptive E-Learning. In Feldman, S., Uretsky, M., Najork, M., Wills, C., eds.: Proceedings of the 13th International Conference on World Wide Web – Alternate Track Papers and Posters (WWW'04), ACM (2004) 104–113
[22] Rosch, E.: Natural Categories. Cognitive Psychology **4**(3) (1973) 328–350
[23] Brusilovsky, P.: Methods and Techniques of Adaptive Hypermedia. User Modeling and User-Adapted Interaction **6**(2–3) (1996) 87–129
[24] Brusilovsky, P.: Adaptive Hypermedia. User Modeling and User Adapted Interaction **11**(1–2) (2001) 87–110
[25] Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D.: Refactoring: Improving the Design of Existing Code. Addison-Wesley (1999)
[26] W3C: XML Schema http://www.w3.org/XML/Schema.
[27] Leach, P., Mealling, M., Salz, R.: A Universally Unique IDentifier (UUID) URN Namespace. Technical Report RFC 4122, The Internet Engineering Task Force (IETF) (2005) http://tools.ietf.org/html/rfc4122.
[28] Sandhu, R., Coyne, E., Feinstein, H., Youman, C.: Role-Based Access Control Models. IEEE Computer **29**(2) (1996) 38–47
[29] Hendrix, M., De Bra, P., Pechenizkiy, M., Smits, D., Cristea, A.: Defining Adaptation in a Generic Multi Layer Model: CAM: The GRAPPLE Conceptual Adaptation Model. In Dillenbourg, P., Specht, M., eds.: Proceedings of the 3rd European Conference on Technology Enhanced Learning (EC-TEL'08), Springer (2008) 132–143
[30] Stash, N., Cristea, A., De Bra, P.: Adaptation Languages as Vehicles of Explicit Intelligence in Adaptive Hypermedia. International Journal of Continuing Engineering Education and Life-Long Learning **17**(4–5) (2007) 319–336
[31] The Dublin Core Metadata Initiative: http://www.dublincore.org.
[32] Tamassia, R.: Constraints in Graph Drawing Algorithms. Constraints **3**(1) (1998) 87–120
[33] Brandes, U., Eiglsperger, M., Herman, I., Himsolt, M., Marshall, S.: GraphML Progress Report. In Mutzel, P., Jünger, M., Leipert, S., eds.: Proceedings of the 9th International Symposium on Graph Drawing (GD'01), Springer (2001) 501–512
[34] JISC: Learning Activity Design in Education: LADiE. Technical report, The Joint Information Systems Committee (JISC) (2006) http://www.jisc.ac.uk/publications.

Non-Invasive Adaptation Service for Web-based Content Management Systems

Kevin Koidl, Owen Conlan, Vincent Wade

Centre for Next Generation Localisation and Knowledge and Data Engineering Group, Trinity College Dublin, Ireland

{Kevin.Koidl, Owen.Conlan, Vincent.Wade}@cs.tcd.ie

Abstract. Most Adaptive Hypermedia Systems today focus on providing an adaptive portal or presentation/application through which adaptive retrieved content is delivered. Moreover the content used by such adaptive systems tends to be handcrafted for adaptivity, i.e. from closed corpus repositories. If adaptive hypermedia is to become more mainstream we need the ability to embed adaptive information retrieval and composition techniques within existing Web-based Content Management Systems (WCMS) e.g. Wiki, Drupal. However the effort and expense in fundamentally re-engineering such WCMS or developing completely new adaptive WCMS, is very high. This research explores the architectural and technical issues involved in providing a third party adaptive service which is more easily plugged into existing WCMS. The advantage of this approach is that it doesn't break the current browsing paradigm of freely navigating across different independent web sites on the web. In order to address this challenge we introduce a third party Adaptive Service that enables a unified cross-website personalized experience by discretely interfacing with separate independent WCMS. Thus the Adaptive Service strives to personalize the users experience when accessing each WCMS in a non-invasive manner by not adversely interfering with the web sites overall look and feel, content and functionality without losing the web sites identity and fundamental experience. This paper describes the design and an initial use case supported implementation example of this Adaptive Service. It also provides a survey that assesses the feasibility of the services integration with leading WCMS.

Keywords: Open Corpus, Adaptive Hypermedia, Web-based Content Management Systems (WCMS), Adaptive Hypermedia Strategies

1 Introduction

Although Adaptive Hypermedia Systems (AHS) have matured, they may still be seen as a niche application mostly with a manually managed closed corpus that is adapted to a well known user group. Furthermore the adaptive logic driving the adaptive process needs to be closely related to the data models and the content. This close relationship mostly results in limitations in the re-usability and interoperability of the AHS, also known as the "Open Corpus Problem" in Adaptive Hypermedia [2].

Recently more flexible AHS architectures have been introduced addressing the problem of re-usability and interoperability. E.g. by providing a distributed server architecture [1] or flexible rule engines facilitating the adaptive logic [4]. Nevertheless most AHS still tend to provide a central AHS portal restricting the user's flexibility; thus isolating the adaptive experience. This paper introduces an approach using a flexible AHS architecture and providing flexibility to the user.

The goal of this approach is to provide open corpus adaptivity by personalizing the browsing experience over several individual content hosting systems known as Web-based Content Management Systems (WCMS). To achieve this we introduce a third-party Adaptive Service discretely interfacing with the individual WCMS. The main advantage of this approach is that the adaptivity is instrumented by the WCMS and not by a central AHS portal. By bringing the adaptivity to the WCMS, the user can maintain the current browsing paradigm by freely navigating on the web and, at the same time, benefiting of personalized content. For this the Adaptive Service provides the individual WCMS with non-intrusive adaptive recommendations reflecting the overall intent/interest of the user. The unified and personalized browsing experience across different WCMS is defined as *adaptive flow*.

In order to illustrate the functionality of the introduced third-party Adaptive Service a use case is provided in section 4. This use case describes the individual steps in which a user can be assisted on the web by the introduced third-party Adaptive Service. Based on the use case one particular implementation example is also given. This example includes a query interception and augmentation by the third-party Adaptive Service connected to a WCMS. The implementation example is followed by a description of the overall architecture.

Motivating the use case, a survey of two prominent WCMS Drupal and Wikipedia's implementation platform MediaWiki is provided. The survey focuses on the suitability of WCMS for adaptivity. Furthermore a state of the art in discussing recent developments in flexible service driven AHS is also provided.

2 Adaptive Hypermedia and third-party services integration

In order to develop a third-party Adaptive Service, which integrates with different WCMS, three main challenges need to be addressed: (1) the identification and implementation of necessary adaptive features in WCMS, (2) the development of adaptive logic driving the personalization process across several independent WCMS and (3) providing a user model unifying the browsing experience across the different WCMS. The second and third challenge is addressed in the following state of the art by discussing current AHS and user modelling developments. However the first and most difficult challenge is addressed in the two following separate sections three and four.

For an AHS to allow more flexible adaptation across different WCMS the possibility of separating the adaptive logic from the content is essential. Recently several

research groups are driving their AHS towards more flexible AHS architectures. A subset of the numerous examples discussed in the literature is: KnowledgeTree [1], APeLS [4], MEDEA [13] and AHA! [6]. E.g. KnowledgeTree implements an AHS by providing a community of distributed servers to clearly separate concerns. This is done by distributing different functions to designated servers compared with one AHS bundling all functionalities together. For this KnowledgeTree provides four types of servers: learning portals, activity servers, value-added services, and student model servers [1]. Although the communications protocols between the different distributed servers are based on a simple HTTP GET request and not on Web Service protocols such as SOAP, KnowledgeTree provides a good example of distributed AHS integration relevant to this research.

APeLS, as another example, also provides a distributed architecture, although by using separate adaptive services not servers. Furthermore APeLS enables Web Service based communication between the different AHS services. The individual AHS services are (a) the adaptive hypermedia service providing the content and (b) the learning environment. The later is used to track the learner, based on the tutor's guidance in the form of learner profiles, assessment information and pedagogical constraints. The result integrates both services and is displayed in the client's browser. Furthermore APeLS allows the flexible design of adaptive logic based on pluggable rule engines managing different narratives driving the adaptivity [5]. In addition to the usage of Web Service communication the multi model approach of APeLS is relevant to this research. Nevertheless both KnowledgeTree and APeLS focus on closed corpus and not apply adaptation across multiple independent services in order to include open corpus in the adaptation process.

In the following the integration of user models within a third-party Adaptive Service is discussed. The main challenge is to provide a modelling approach that reflects a unified browsing experience over several different WCMS without the need to model separate user models for each WCMS. For this a distributed user modelling service unifying the different modelling parameters of the WCMS is necessary. For example Personis [8] provides such a distributed user model approach designed to provide a unified user model of several different systems. An additional distinctive feature of Personis is user model scrutiny, ensuring the user is involved in all user model related decisions. The architecture is based on a XML-RPC interface allowing third party service integration. A similar approach, centralizing the user model, is taken by CUMULATE [3]. Nevertheless Personis and CUMULATE lack flexibility due to the unified storage of user models in a centralized repository. A more flexible approach is followed by FUMES [15] providing a decentralized mapping framework to support the exchange of heterogeneous user models without the need of a central repository. In relation to the introduced third-party Adaptive Service FUMES provides a possibility of retrieving a single user model based on user information collected in several different WCMS.

3 WCMS survey for compatibility to support modular based Adaptive Service access

This section is based on a survey investigating the integration of third party adaptive services in Web-based Content Management Systems (WCMS). The following is only a brief extract of the entire survey introducing the most important conclusions for this research.

Taking a broad view on WCMS shows that this type of system refers to a whole range of different applications and functionalities with the main commonality of providing effective and accessible tools for publishing and organising web-based content. Goodwin and Vidgen define a WCMS as:

" [...] an organizational process, aided by software tools, for the management of heterogeneous content on the web, encompassing a life cycle that runs from creation to destruction" [14]

The term WCMS mostly refers to open source based Content Management System rather than to commercial systems which are referred to as Enterprise Content Management Systems (ECM). Prominent examples for WMCS are MediaWiki, Drupal and WordPress. Beside these systems numerous WCMS have appeared and have become an enormously popular application domain in day to day Internet activities.

The integration of adaptive functionalities in WCMS pose a new set of challenges towards the adaptive process with the following as the most significant:

- Any adaptive intervention has to maintain the entire look and feel i.e. branding of the WCMS.
- All internal policies regarding user rights, especially content-related rights, need to be obeyed by any adaptive intervention.
- The WCMS has to provide API or Web Service interfaces to enable third party Adaptive Service intervention.
- The WCMS needs to be extendible/pluggable in order to handle adaptive interventions.
- Semantic web functionalities have to be available within the WCMS, e.g. via a module based extension to the core platform.

To assess and tackle these challenges, two specific WCMS were examined, MediaWiki [9] well known as the basic WCMS used for Wikipedia and Drupal [7] known as one of the most flexible WCMS based on its flexible pluggable module architecture. Both systems provide a wide range of possibilities and are supported by a large and active developer group.

The architecture of MediaWiki is simpler than Drupals, but it is not as flexible. This is due to the fact that Drupal is based purely on a module based pluggable architecture.

However MediaWikis core strength lies in the management of content and not in the extensibility of the platform. Nevertheless extensions are possible, e.g. the *extensions* feature is based on simple scripts for the adding of different plugin types like Adobe Flash, Video streaming, RSS feeds, ratings and API based third party accessibility. It also has to be noted that recently more enhanced extensions were introduced especially for the adding of semantic structures and relationships within the different MediaWiki implementation [10].

The WCMS Drupal on the other hand is based on a flexible module based pluggable framework also referred to as "Content Management Framework" [7]. Drupal does not specialise on one specific type of content, like MediaWiki which focuses on encyclopaedia based content or Wordpress concentrating on blog based content, but it provides an extendible core implementation that can plug different modules depending on the application area. This high level of flexibility and abstraction comes with the cost that developers need to have good knowledge of the core architecture in order to extend it. Like MediaWiki, Drupal also provides a module to enable semantic annotations [11].

However it is essential to explore the extensibility of WCMS for it to engage with a third-party Adaptive Service and to provide the ability to use the adaptive interventions send by a third-party Adaptive Service. Currently both discussed WCMS provide the possibility to communicate with a third-party Adaptive Service. Nevertheless both need to be extended to cater for effective adaptivity from within. Currently it is not possible to take any WCMS deployment and apply adaptivity simply by using API function calls.

Fortunately the engineering of a WCMS is not a difficult task and especially in the case of MediaWiki and Drupal extensibility is possible without major changes to the WCMS architecture. E.g. MediaWiki API provides a powerful interface for fast and high level access to all data within the database. Functions include user login, content uploads and updates. In addition to the API MediaWiki provides specific extension points for the extension of its deployment. These extensions can be plugged into the core implementation at any time and do not trigger any re-deployment. The following extensions can be seen as relevant for enabling adaptivity in MediaWiki: "Hooks" to react to user actions, displaying "special pages" and "skins" allowing changes in the look and feel of MediaWiki.

Extending the main Drupal core on the other hand implies the extension with specific Drupal modules. These modules are simple to design, plug and unplug. From within the modules several functionalities to control the information flow from the database to the Front End are provided. Following events are useful for the "hooking" of adaptive interventions into Drupal: Content/Node creation, deletion and viewing, as well as user login/logout and user account/profile updates. Both WCMS, MediaWiki and Drupal therefore provide a good base for further development towards more flexible and distributed adaptivity.

Besides applying extensions to the individual WCMS an alternative approach is possible. This approach is based on the fact that most interactions between user and WCMS are based on the usage of a browser. Furthermore current browser technologies allow the integration of extension/plug-in which can be used to manipulate, augment or redirect the data before being displayed to the user. In relation to this research an interesting example is the "Smarter Wikipedia" Firefox plug-in, which adds a "related articles" box to MediaWiki's Wikipedia implementation [12]. This kind of browser central development presents an alternative approach which avoids the current need to extend WCMS, but may lead to constant updates due to changes in the underlying WCMS. Nevertheless a purely browser focused implementation is currently not part of the approach introduced in this paper.

In the following a use case is illustrated discussing the usage of the third-party Adaptive Service on WCMS.

4 Use Case

This use case illustrates the usage of the third-party Adaptive service providing a unified adaptive experience over several independent WCMS. As mentioned above this unified experience is defined as *adaptive flow*. It's main purpose is to retain a unified personalized information space for the user. To achieve this each of the WCMS communicates with a third-party Adaptive Service. As the user navigates, the Adaptive Service gains knowledge about their browsing over time. Thus, the Adaptive Service can provide improved non-intrusive adaptive recommendations to the WCMS. Figure 1 illustrates a specific scenario.

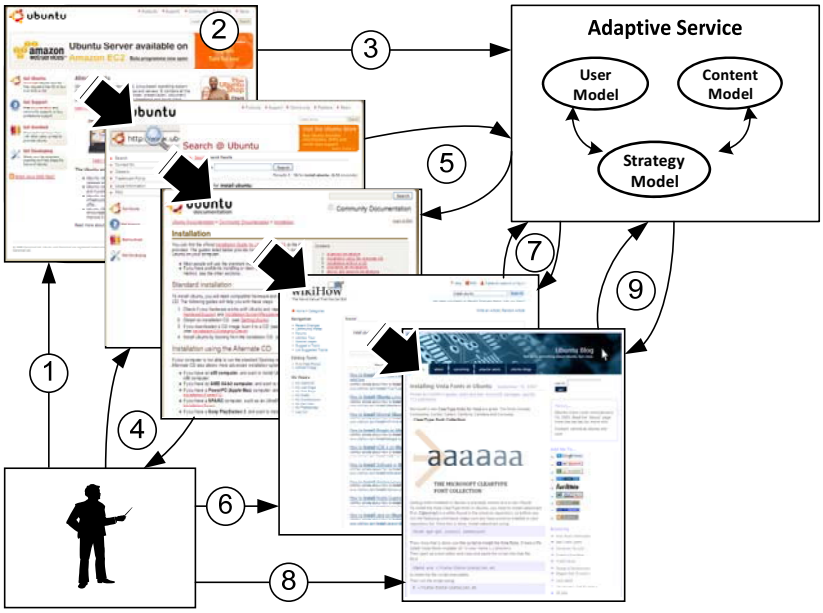

Figure 1 illustration of the overall adaptive approach indicating an *adaptive flow* over several different WCMS

1. User John is interested in installing the Debian GNU/Linux based Ubuntu operating system, but before he makes his final decision he wants to gather information about the installation process. For this he navigates to the Drupal based Ubuntu homepage. John's Adaptive Service is active and will provide a more personal browsing experience.
2. After navigating to the Ubuntu homepage John states the explicit part of his interest in the search field. He uses the term "install Ubuntu".
3. The search module of the Drupal based Ubuntu homepage informs the third-party Adaptive Service about John's query. The Adaptive Service registers the query and cannot find any previous interest related to this query. Therefore the Adaptive Service initialises a new "adaptation flow" session prompting the Adaptive Service to wait for more evidence coming from John's interaction with the Ubuntu homepage.
4. After stating the query John receives the result list from the Ubuntu homepage without any adaptive interventions. He starts clicking on different results from the original result list.
5. John's interaction with the result list is registered by the Adaptive Service. At this point the Adaptive Service remains in a non-adaptive state identifying John as being in an 'orientation' phase.
6. John believes he has enough high level information and leaves the Ubuntu homepage. Now he navigates to the MediaWiki implementation "wikiHow".

He wants to receive more in depth information about the installation process and hopes to find it at this point.

7. The Adaptive Service is informed by "wikiHow" and registers John's access to the MediaWiki based page. John uses this page frequently and knows that the Adaptive Service is interacting with the WikiMedia based WCMS. After the Adaptive Service receives the information that John is navigating to the new WCMS the Adaptive Service sends an extended query to the "wikiHow" page. John is now presented with a personalized result list based on his previous browsing experience on the Ubuntu Homepage, instead of seeing the main homepage in which John would have had to reissue his query.
8. After interacting with the personalized search result provided by the "wikiHow" webpage, John decides to navigate to the WordPress based blogging site "Ubuntu blog".
9. The Adaptive Service now maintains a well informed stream of experience from John's previous browsing pattern and is able to negotiate the most appropriate blog entries for John.
10. John believes he is well informed and decides to install Ubuntu.

The most essential part of the illustration in figure 1 is indicated with the diagonal arrows ranging over the different WCMS. It indicates the adaptive flow which can be seen as a development towards a more personalized browsing experience. Furthermore it is important to note that the illustrated open corpus adaptive scenario is controlled principally by the user. The Adaptive Service only provides adaptive recommendations which then can be used by the WCMS during their interaction. The use of the Adaptive Service allows different WCMS to share the latent aspects of the user's preferences and intent. These are typically lost as the user navigates between different WCMS on the web.

Compared with conventional distributed AHS the introduced third-party Adaptive Service follows a different approach in that it seeks to maintain attributes which apply across a variety of WCMS, i.e. by using content models representing semantic concepts.

To illustrate the functionality of the third-party Adaptive Service an implementation example is provided in the next section.

5 Implementation example

Taking the Drupal WCMS as an example this section defines the means by which the Adaptive Service can effect the recommendation of content to the user.

Figure 2 exemplifying a basic adaptive integration within the Drupal WCMS.

As indicated in figure 2 the basic Drupal search module was activated. Furthermore the Drupal *hook_search_preprocess* is used to intercept the user's query and send it to the Adaptive Service. In addition to the query interception, content related user activities are logged with Drupals *node_hook_api*. It registers content related activities and saves these in the underlying database. This information is then used to extend the user model of the Adaptive Service.

The Adaptive Service on the other hand has to handle the incoming information in order to send adaptive recommendations back to the WCMS. For this the Adaptive Service implements a JBoss Rules engine based on APeLS [4]. This engine allows the usage of flexible adaptive strategies manifested as individual rules. Based on the example illustrated in figure 2 the following steps are executed by the Adaptive Service:

- The Adaptive Service identifies and authenticates the user.
- The Adaptive Service waits for information to be sent from the Drupal based WCMS about the user's activities.
- After the user issues a query it is intercepted by Drupals hook_search_preprocess which sends it to the Adaptive Service.
- In addition the Adaptive Service can receive information about the user content related interactions from Drupals hook_node_api.
- Based on the available user and content model information a specific adaptive strategy is activated.
- The activated adaptive strategy orchestrates the information provided in the user and content model and uses it to augment the query.
- The Adaptive Engine sends the augmented query back to the WCMS.
- The WCMS executes the query and presents a personalized ranked list to the user.

This example illustrates a query interception and query augmentation scenario for Drupal. In this case the final result is a personalized ranked list. However this approach allows more complex adaptive scenarios like adaptive navigation support and adaptive presentation necessary for the overall research illustrated in figure 1.

6 Architecture

This section describes the overall architecture of the third-party Adaptive Service and its integration with different WCMS. The Adaptive Service is based on APeLS [4] and uses pluggable rule engines to facilitate the wide variety of WCMS architectures.

This architecture addresses three main challenges: (a) the sending of user and domain information as input for the overall personalization process from the WCMS to the Adaptive Service, (b) the processing of the send information by the Adaptive Service (c) the creation of appropriate adaptive recommendations by the Adaptive Service to personalize the output of the WCMS. Please note figure 3 illustrating the overall architecture.

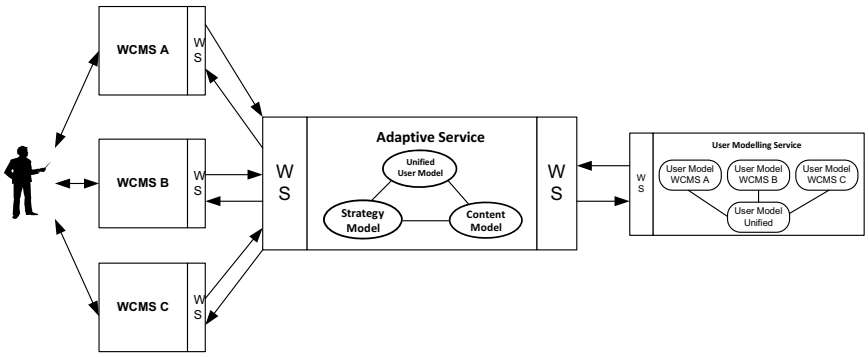

Figure 3 indicating the overall architecture of the third-party Adaptive Service

The input for adaptive personalization across independent WCMS is based on the user's browsing behaviour. This behaviour is collected by the WCMS and delivered to the Adaptive Service through a Web Service (WS) interface in the back-end of the WCMS. The user simply navigates within and between the different WCMS with a standard browser and is not explicitly aware of the back-end integration. This allows the user to remain in full control of the browsing experience.

The processing of all relevant information including the user and the domain model is handled by the Adaptive Service. For this the Adaptive Service communicates with an additional user modelling service e.g. FUMES [15] which provides a unified user model reflecting the current and previous browsing experience across separate WCMS. Together with a content model, which stores all information available about the structure and nature of the content within the WCMS, the Adaptive Service can compose an appropriate adaptive strategy reflecting the overall intent of the user.

Based on the appropriate adaptive strategy the Adaptive Service can send adaptive recommendations to the WCMS. The WCMS can use these recommendations to create a personalized output for the user.

The most significant feature of the introduced architecture is placing the Adaptive Service behind the WCMS allowing the user to retain the current browsing paradigm of free web navigation.

7 Conclusion and Future Work

This paper addresses the open corpus problem by introducing a third-party Adaptive Service providing adaptivity across different independent WCMS. The novelty of this approach is the complete separation of the Adaptive Service and the content host i.e. the WCMS. For this the Adaptive Service discretely connects with the WCMS providing non-intrusive adaptive recommendations. The user does not have to be aware of the adaptive interventions, thus this approach retains the current browsing paradigm of free navigation on the web. Furthermore the non-intrusive nature of the adaptive interventions maintains the branding, look and feel and operation of the individual websites modified.

To illustrate the usage of the third-party Adaptive Service a use case was provided and later specified by an implementation example proving third-party Adaptive Service integration with a Drupal based WCMS. However the example illustrates only one possible adaptive application within the overall architecture introduced. Further developments towards more sophisticated personalization include more flexible content and link adaptation. However the introduction of such advanced adaptive functionalities strongly depends on the architecture of the different WCMS. In order to gauge the feasibility of working with different WCMS a survey was discussed which indicates general key challenges for the integration of a third-party Adaptive Service in WCMS.

Future work will concentrate on three connected areas: (1) the integration of additional WCMS into the adaptive service framework, (2) the extension and empirical evaluation of resulting adaptive logic/strategies and (3) the further integration of a unified user modelling approach across different WCMS.

Acknowledgements. This research is supported by the Science Foundation Ireland (grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at Trinity College, Dublin.

8 References

[1] Brusilovsky, P.: KnowledgeTree: A distributed architecture for adaptive e-learning. In: Proceedings of The Thirteenth International World Wide Web Conference, WWW 2004 (Alternate track papers and posters), ACM Press 104–113. (2004)

[2] Brusilovsky, P. & Henze, N.: “Open Corpus Adaptive Educational Hypermedia”. In The Adaptive Web: Methods and Strategies of Web Personalisation, Lecture Notes in Computer Science, vol. 4321, Berlin: Springer Verlag, pp. 671-696. Berlin (2007).

[3] Brusilovsky, P., Sosnovsky, S. A., & Shcherbinina, O.: User Modeling in a Distributed E-Learning Architecture. Paper presented at the 10th International Conference on User Modeling (UM 2005), Edinburgh, Scotland, UK, July 24-29, 2005.

[4] Conlan, O., Hockemeyer, C., Wade, V., Albert, D.: Metadata driven approaches to facilitate adaptivity in personalized eLearning systems. Journal of the Japanese Society for Information and Systems in Education 1(1) 38–45 (2002).

[5] Dagger, D., Conlan, O., and Wade, V. P.: An architecture for candidacy in adaptive eLearning systems to facilitate the reuse of learning Resources. In: Rossett, A. (ed.) Proc. of World Conference on E-Learning, E-Learn 2003, Phoenix, AZ, USA, AACE 49-56 (2003).

[6] De Bra, P., Smits, D., Stash, N., The Design of AHA!, Proceedings of the ACM Hypertext Conference, Odense, Denmark, August 23-25, 2006 pp. 133, and http://aha.win.tue.nl/ahadesign/, (2006).

[7] Drupal Community Page, http://drupal.org/getting-started/before/overview. (Last access 25.02.2009)

[8] Kay, J., Kummerfeld, B., and Lauder, P. Personis: A server for user modeling. In: De Bra, P., Brusilovsky, P. And Conejo, R. (eds.) Proc. of Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems AH'2002, Málaga, Spain, pp 201-212 (2002).

[9] MediaWiki Foundation, http://www.mediawiki.org/wiki/MediaWiki (Last accessed 16/03/2009)

[10] Semantic MediaWiki, http://semantic-mediawiki.org/wiki/Semantic_MediaWiki (Last access 16/03/2009).

[11] Semantic Drupal, http://drupal.org/node/299584 (Last access 16/03/2009).

[12] Smarter Wikipedia, http://en.wikipedia.org/wiki/Smarter_Wikipedia (Last access 16/03/2009)

[13] Trella, M., C. Carmona, and R. Conejo: MEDEA: an Open Service-Based Learning Platform for Developing Intelligent Educational Systems for the Web, in Workshop on Adaptive Systems for Web-Based Education: tools and reusability (AIED'05). 2005: Amsterdam, The Netherlands. pp. 27-34 (2005)

[14] Vidgen, S. G: Content, content, everywhere... time to stop and think? The process of web content management. Computer and Control Engineering Journal. Vol. 13, No. 2, pp. 66-70 (2002).

[15] Walsh, E., Dagger, D. and Wade, V.P., Supporting “Personalisation for All” through Federated User Modelling Exchange Services (FUMES). in Towards User Modelling and Adaptive Systems for All Workshop at User Modelling 07, (Corfu, Greece, 2007).

Investigating a thematic approach to narrative generation

Charlie Hargood, David E Millard, Mark J Weal

LSL, Department of Electronics and Computer Science, University of Southampton, Southampton, England
cah07r@ecs.soton.ac.uk,dem@ecs.soton.ac.uk,mjw@ecs.soton.ac.uk
http://www.lsl.ecs.soton.ac.uk/

Abstract. Adaptive hypermedia aims to create a tailored experience for users and with narrative generation this dynamic content can be in the form of engaging stories. Narrative generation systems range from the automatic generation of bespoke stories to the representation of existing information as narrative. However narrative generation systems often produce bland and unvaried stories. In this paper we propose that the inclusion of themes will enrich the resulting narrative and explore how a prototype thematic system might be integrated with existing methods of narrative generation. This investigation reveals that integration at the generation of story elements is important to avoid constraints on desired themes, that the detailed characters fundamental to character centric narrative generation make integration difficult, and that integration must be done carefully to avoid damaging the resulting narrative.

Key words: Adaptive hypermedia, narrative, narrative generation, thematics

1 Introduction

The presentation of relevant media in an engaging fashion to users is a difficult challenge, a user constructing a presentation of information or media from a personal collection can often have difficulty finding items relevant to each other and to the task at hand as well as presenting them in a flowing an engaging manner. Adaptive Hypermedia systems create a dynamic experience for users by adapting content and navigational choices. Their goal is often to create a high quality experience that is tailored to the user's needs and requirements, and to avoid problems of information overload [5]. Narrative systems are a particular type of adaptive hypermedia application that attempt to generate content within a narrative or story framework. However, limitations with narrative generation methods can compromise the quality of the resulting experience, with stories that amount to a list of actions of characters who coldly state their motivations and pursue them directly in steps without any elaboration or subtlety.

Narratives, or stories, are a prevalent representation of human experience that can engage and entertain. Work in the field of narrative generation presents

a potentially powerful solution to the problem of generating engaging and relevant information dynamically. Narrative generation seeks to automatically create bespoke narratives for a variety of uses, either generating requested narratives from scratch or by representing existing information as a narrative.

Existing narrative generation systems while often successfully generating short narrative can find their results bland or unvaried depending on the limitations of their approach. We suggest a thematic approach to narrative generation where the addition of themes will enrich generated narratives making them closer to human authored narratives with a thematic objectivity beyond the base communication of the information present within the narrative.

A thematic model has been developed [9] based on the work on thematics in narratology [17]. The model describes themes within a narrative being built of themes and motifs and how they are connoted and denoted from elements of the narrative itself. This was further developed into a prototype, the Thematic Model Builder (TMB) that could score narrative segments on their relevance to desired narratives, and an evaluation is being performed into the effectiveness of the system and model at representing and connoting themes.

The focus of this paper is how such a thematic system could be integrated with existing narrative generation approaches in order to enrich the resulting narratives. We review existing approaches to narrative discourse generation and explore at what level a thematic system should be involved with narrative generation, what approaches it best compliments, and what benefits are likely to emerge from an integration.

2 Background

2.1 Narratology

Narratology is the study of narrative within literature and as such is primarily focused on the analysis of narrative. However it does provide a useful insight into how stories are constructed.

Structuralism is an area of narratology concerned with deconstructing narratives to identify the components from which a story is built and the structures that they build within a story. Because of the tangible nature of structuralism its ideas are very useful for narrative generation as its clear definition of structures and elements give an insight into what narrative generation systems should be generating. Most structuralists assert that a narrative is composed of an authored sequence of human experiences [13], and as such may be deconstructed into two important components; story, and discourse [4]. The story (or fabula) is a collection of all the information to be communicated and the discourse (or sjuzhet) represents the exposed parts of the story that are told and how they are presented (shown in Figure 1).

The story element is the sum of all experiences and elements that make up the narrative. The discourse represents what parts of the story are exposed in the narrative (the story selection) and how it is told (the story presentation).

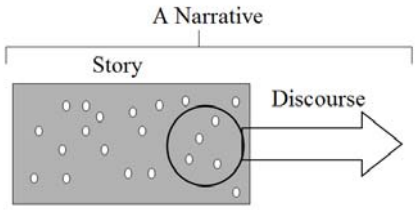

Fig. 1. A narrative can be deconstructed into story and discourse

Discourse is a complicated process concerning many different decisions including how the story is presented, what medium is used, the style, the genre, and the themes of the narrative. Thematics approaches themes with a structuralist method of deconstruction and identifies the narrative elements that communicate themes.

Tomashevsky identified the thematic elements of themes (broad ideas such as 'politics' or 'drama') and motifs (more atomic elements directly related to the narrative such as 'the helpful beast' or 'the thespian') [17]. He explains a structure of themes being built out of sub-themes and motifs. A motif is the smallest atomic thematic element and refers to an element or device within the narrative which connotes in some way the theme. Themes may always be deconstructed into other themes or motifs whereas a motif may not be deconstructed.

2.2 Narrative Generation

Narrative generation systems use a variety of different approaches and have a wide range of objectives. While many systems seek to generate full narratives for entertainment such as the virtual storyteller [16] and AConf [15] some systems use narrative generation to add additional meaning to information by representing it as a narrative using narratological devices like sequencing, emphasis and omission such as in Topia [3] and adaptive hypermedia systems like AHA! [8].

As a process narrative generation can be broken down into three stages; story, plot, and presentation generation. Depending on the project in question these stages can be consolidated together or separated, (for example, in the virtual storyteller, presentation generation is broken down in narration and presentation [16]). The majority of narrative generation projects deal with the creation of the narrative elements (story generation); resolution of the sequence of events that comprise the narrative and selection of narrative elements to be exposed and building of relationships between these elements (plot generation); and presentation of the narrative through a chosen medium (presentation generation). Figure 2 illustrates this process.

According to Riedl and Young [15] narrative systems take either a character or author centric approach depending on whether the system seeks to model the characters within the story, the authorial process itself, or whether the system

is a compromise of both approaches. [15] also identifies a third approach in the form of story centric approaches these are however less common and due to their more linguistic focus are less relevant to this research.

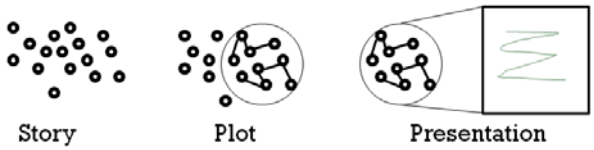

Fig. 2. Narrative generation can be broken down into three stages

Character Centric Character centric narrative generation revolves around the perspective of modeling the behavior and goals of the characters of a story. With the characters successfully simulated they are released to pursue their goals and their actions are exposed, the idea being that stories are everywhere and an engaging narrative will naturally emerge from the actions of a set of well-motivated characters.

Character centric narrative generation systems often use agent technology to suitably simulate the characters and their behaviors with a purpose built agent taking the part of each character such as in work by Cavazza [7] and in the Facade system [12] (Facade is not entirely character centric, but its approach is very similar). Sometimes the intelligence is much more simplistic and a reasoning system will handle the goals and behavior of all characters, such as in TaleSpin [14]. However, these systems lack the power to generate varied narratives and although short simple stories are generated the lack of in-depth modeling of individual characters behavior removes personalized variety from their actions.

Automatic generation of story elements is rare in character centric narrative generation. This is because elegantly written characters with sophisticated behavior are key to narratives being successfully emergent from the generated result and at present the only way to ensure this is to build the characters by hand. Some story elements are generated by using character archetypes with cliche behavior such as with the supporting characters in work by Cavazza [7] but it is rare to find this for key characters.

Plot generation in character centric generation is therefore a direct result of the characters behavior as dictated by the agents playing them or the intelligence modeling all of the characters. The actions they take to achieve their goals builds the relationships between story elements and the sequence of events that makes a plot. Presentation generation is not specifically tied to the character centric approach but the focus on entities and modeling their actions make character centric approaches ideal for presentation in game engines (for example AConf used the UT engine through the mimesis project [18]). Although the presentation of character centric systems still sometimes uses text as a medium of choice either

using sentence templates such as in talespin [14] or generated text using natural language processing.

The main weakness of character centric narrative generation is its reliance on an engaging narrative successfully emerging from the exposition of the characters actions. Often these systems generate bland stories that merely report on a series of uninteresting actions. These stories are thus often sensible and varied but lack narrative richness or interesting plot.

Author Centric Author Centric narrative generation seeks to model the authorial process itself rather then the content of the narrative. The systems seek to model the process by creating rule based systems or narrative grammars that use well defined structures that are typical of the desired genre of narrative in order to generate stories.

Author centric narrative generation also lends itself better to the representation of existing knowledge as narrative as its story elements are not necessarily the narrative devices such as characters and objects but the devices the author needs to construct a story. Systems such as ArtEquAKT [2] create narratives out of a variety of resources and media from the internet and for this project story generation is the compilation of these resources. The same could be said for narrative influenced hypertext systems such as Topia [3].

Some systems do model the contents of the narrative to be generated as part of story generation but remain author centric. Universe [11] builds stories around a set of author goals and constructs a structure for a story to satisfy these but does so using the actions of characters modeled from cliche archetypes and a finite set of actions. In other author centric systems the story structure is not explicitly generated, but emerges from the selection of a predefined set of story elements, such as in Card Shark [6].

Plot generation in these systems is a case of applying the rules of the system for the desired genre, utilizing the grammar with the available resources, or filling a story template with appropriate resources. Presentation for author centric systems is often text based, either using templates such as Universe[11] or Artequakt [2] or simply exposing the elements in sequence such as in Card Shark [6].

Author centric systems tend to be highly specialized for one particular type of narrative, making them inflexible and also often not with a view to generic narrative generation. The stories are seldom varied as they all follow a similar authoring process with the same rules and/or grammars and as such can generate engaging but not often varied narratives.

Compromise Approaches Many narrative generation systems often seek a compromise between these two approaches in order to counteract the weakness of using one approach or another. Some systems such as Facade[12] and Universe[11] will only make slight compromises, such as the ideal story drama curve approach in Facade or the choice to model characters in Universe, but others make much larger steps towards marrying the two approached.

The virtual storyteller [16] at first seems to be a character centric approach that uses agents to model the behavior of its characters, the difference arises with the addition of an extra director agent. The director agent has a set of rules about what makes an engaging story, much like an author centric approach, and uses these rules to influence the narrative by vetting character actions, influencing them by giving them new goals, and creating story events to channel the emergent narrative into being more engaging.

AConf[15] models each 'actor' as an expert system seeking to achieve its goals, giving it characteristics of a character centric approach, but it is fundamentally more author centric as its process of plot generation centers around the structure of the narrative building it as a network of events using story planners.

The presentation generation for these systems also vary. In the virtual storyteller [16] the director agent directly communicates with a narrator and presenter to generate text using sentence templates whereas AConf [15] uses its character's modeling and plot plans to interact with a system called mimesis [18] which uses the UT game engine to present the narratives.

These systems experienced mixed success with both reporting the generation of successful narratives. However both suffered from similar problems to character centric approaches, while the addition of measures to ensure the narratives structure is engaging does have a positive effect the engaging narrative can at times still fail to emerge from the result and the systems can be reliant of stories that are heavily predefined at the request stage rather then being entirely generated.

3 A Thematic Approach

3.1 The Thematic Model

Using a thematic model [9] based on Tomashevsky's work to describe how themes are constructed within a narrative we propose a thematic underpinning to narrative generation. The basis of the model (as illustrated in Figure 3) is built of *natoms* (narrative atoms) which contain *features* that denote *motifs* which in turn connote *themes*.

For example, we might view a digital photo as a natom, and the tags on that photo as the features that denote a particular motif. Thus a photo tagged with 'daffodil' could denote the motif of 'flower', which connotes the theme of 'Spring'. Themes can themselves build up into new themes, for example the theme of 'Christmas' can be used to connote the theme of 'Winter'.

3.2 The TMB

To facilitate an evaluation of the effectiveness of the model a prototype was built that could use an instance of the model to select images from a group of Flickr [1] images based on their ability to connote a desired theme. The prototype went under the working name of the Thematic Model Builder (TMB).

[1] http://www.flickr.com

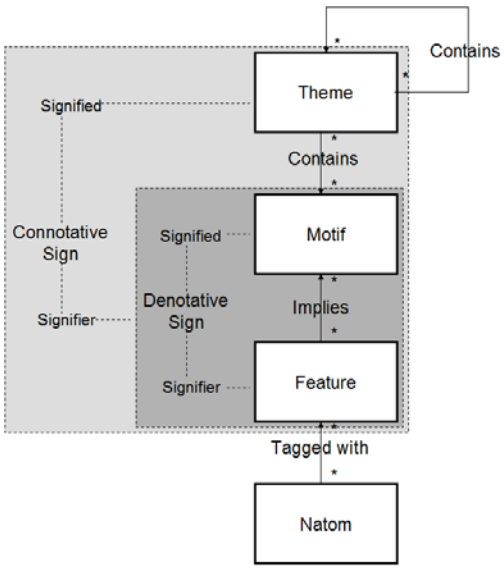

Fig. 3. The Thematic Model

As an original instance of the model four themes were modeled (Winter, Spring, celebration, family) along with all sub themes and motifs of these themes, XML was used to build this instance. Defining an instance of the model for particular themes is a complex and subjective process. We explored a systematic method for building themes based on semiotics [9]. Initially we identify what *connotes* the desired theme, these connotative signs make up the themes sub themes and motifs. However, these signs become sub-themes only if when expanded all of their connotative signs in turn connote the original theme, otherwise they are a seperate (if associated) theme. Connotative signs anchored to a specific element within the narrative become motifs which have their features defined by likely tags that *denote* the element.

The prototype was written in java with a simple JSP front end and Flickr was used as a source of natoms. As a folksonomy its items have rich semantic annotations in metadata [1] as opposed to the automatically generated metadata present in some other media collections. This makes the features in each image apparent and it also has a large freely available body of resources. The library of images (the fabula) was generated by making a keyword search of Flickr on the desired subject and storing the top n images (where n is the desired size).

The system then followed an algorithm of measuring the thematic quality of each natom in the fabula. It returns the natoms with the highest scores according to two metrics:

– *Component coverage*: the proportion of high-level sub-themes or motifs that a natom has features for - this is useful for measuring how strongly a natom

matches the desired theme. (for example, winter expands several high-level sub-theme and motifs including christmas, snow and cold. A natom matching just one of these has less coverage than one that matches many)
- *Thematic coverage*: the proportion of desired themes that a natom has features for - this is useful for searches with multiple themes

The TMB prototype allows us to compare the effectiveness of selecting photos according to their theme with the process of selecting photos based directly on their tags. A pilot evaluation of the prototype has been completed in [10], early results are promising and show the TMB successfully compiling collections that have been evaluated as better connoting the desired themes, and performing better in a narrative context then standard keyword selection.

4 Integration

Referring back to the division of narrative generation illustrated in figure 2, we can explore the possibility of a thematic systems involvement at different levels of narrative generation. Themes are intangible concepts, a subtext rather then a core focus of the narrative, and for this reason it seems at first that narrative generation would be benefit from thematic involvement at the presentation level. Here themes could be connoted by emphasis given through the presentation to the features within the narrative that denote motifs which in turn connote the desired themes. At this level a thematic subtext would become present through elaboration on the presentation of the plot. However this is a process that could potentially fail if there were no relevant features present in the narrative that could be elaborated upon to help connote the desired theme, the system might find that at the presentation level a thematic system might only be able to offer from a subset of themes.

At the story level of narrative generation a thematic systems involvement would be in some ways the opposite of its involvement at a presentation level. Instead of offering elaboration on existing narrative features at the story level a thematic system would generate additional narrative elements based on a shopping list of required motifs for the desired themes. This way themes would become apparent through the presence of certain story elements that connoted the desired themes. This could potentially fail however if the systems plot generation did not make use of the thematic story elements or they were not properly exposed possibly leading to absence of key motifs. Also, such an approach could damage the generated narrative more then help it potentially flooding the system with elements irrelevant to the plot. For some systems as well story generation integration is not always an option, at least on a fully autonomous level, with many systems generating plot out of pre written and defined story elements. These semi automatic approaches to story generation require a very different approach perhaps with thematic guidance on the creation of these elements as supposed to influencing the automatic generation process in others.

At the plot generation level thematics could play a role in the story selection of narrative elements as well as the way relationships build. The first part of this

would be similar to how the TMB prototype builds photo montages in that a list of desired motifs would be compiled and this would be used to thematicly score potential story elements and as such influence their selection and inclusion in the plot. Also, the relationships between story elements and actions of elements could in turn be factored in as features that denote motifs, as such potential actions at the plot generation of the story could be thematicly scored influencing what occurs. For example a story in which violence is a desired theme might see the protagonist kill the antagonist rather then banish or imprison them. However, like inclusion at the story level it is possible that heavy thematic involvement could damage the plot itself, making its involvement a dangerous balancing act, potentially forcing plot actions that damage the quality of the narrative. Furthermore like involvement at the presentation level, a lack of complete control over the story elements could potentially restrict available themes.

On the question of which approach to take character centric approaches are perhaps the most different from the current implementation using the thematic model. Rather then the natoms of narrative segments that our model describes, character centric approaches start by simulating the content of the narrative itself, modeling characters, locations, entities, and events. However, through the process of plot and presentation, character centric approaches still go on to generate natoms that contain features and by extension denote motifs. An integration would have to seek to ensure that certain features were planted in order for the themes to become apparent in the finished narrative. To do this involvement at the story level seems obvious as this is where the elements present within the narrative are generated. However, as a more semi automatic approach with predefined elements is more common then automatic generation at this level in character centric approaches it could be difficult to integrate a thematic approach with the prewritten characters. At the plot and presentation levels an integration seems more possible, potential character actions and story events can be thematicly scored to influence actions taken to be conducive with desired themes and then presented in a way that emphasises the relevant thematic content. Character centric generations frequent use of game engines means that integration at the presentation level may be easier where knowledge of the entities present in a particular scene is much more exact then in natural language. However as already discussed a reliance on integration at these levels potentially limits the available themes.

Author centric approaches are more similar in process to the current implementation of the thematic approach in that they're heavily based on structures and largely concerned with the authoring process rather then modeling the content of the narrative. The story generation process for some is about composing a pool from large collections of potential natoms, often from the Web, based on their relevance to required parts of the narrative structure. This is very similar to the way the TMB currently puts together selections for montage, and it is easy to see that with author centric projects that work this way thematic integration would be a relatively simple process of scoring potential segments to generate. At the plot level, integration could be a similar process to the inte-

gration that would be used with character centric approaches, in that elements selected for exposure could be chosen on their thematic qualities rather then only narrative ones. However, for rule based systems of plot generation thematic rules would need to be written for the system. The feasibility of this would need to be added on a system by system basis. At a presentation level natural language generation poses difficulties for thematic integration as a full lexicon for desired features would need to be developed and integrated with the system. Forcing it to uses a small subset of words might make the language clumsy and its important to remember that the thematic model was created with the theory of structuring a narrative in mind where as the structure of individual pieces of language is very different. However, for those systems that use templating or selected pre authored text presentation, using thematics becomes more feasible where narrative techniques such as emphasis (spatially or visually) can be used to highlight relevant segments to help connote a theme.

The possibilities apparent from this investigation are summarised in the table in figure 4. Decisions and selections made in generation may be influenced thematically by making the objective of the decision thematic as well as for plot objectives. Further thematic integration can be achieved through emphasis at the presentation or plot levels and other presentation choices such as style may have an influence that could be worked in favor of desired themes.

5 Future Work and Conclusion

In this paper we describe an investigation into the potential challenges of integrating a thematic system with narrative generation. It is possible that a thematic approach may improve the quality of narrative generation by enriching the resulting narratives which, could give adapted hypertexts tailored to the user and of a high quality. However, the question of integration between narrative systems and thematics is a complex one. At what point in the process of generation should there be thematic influence and what style of approach or compromise makes the prospect of a thematic narrative generation system most feasible?

Narrative generation is a rich and varied field with a wide variety of approaches. This variety is maintained in modern work and testament to the fact that no single perspective is a perfect solution to the problem, with this compromise approaches are becoming more popular as systems aim to find a successful middle ground. The complexity of the problem also means there are multiple stages of a system that it being integrated could exert an influence and effect the resulting narrative in different ways, all of which pose different constraints.

It seems that a combination of issues make integrating thematics with character centric narrative generation difficult. As the quality of such systems is reliant on rich and complex characters, they are often authored by hand which makes influencing the stories fundamental elements in an automatic way difficult, at the same time, abandoning thematic involvement at this level instigates constraints on the available themes to be used at later levels. These constraints mean that

		Narrative			
		Story		Discourse	
				Plot	Presentation
		Semi Auto	Auto		
Generation Approach	***Character Centric***	*Uses Prewritten Characters.* Could be supported by thematic writing tools.	*Goals and Rules are generated.* Generation process influenced thematically.	*Characters and Elements Selected. Rules applied. Actions chosen.* Thematically influenced selection of characters and elements as well as actions chosen.	*Generation of media. Revealed narrative chosen.* Emphasis of relevant elements and relationships. Choice of style.
	Compromise	*Uses Prewritten Characters and purpose built Director Agent or an author rule set.* Could be supported by thematic writing tools and have a thematically influenced author/director.	*Both characters and author goals and rules generated.* Goal and rule generation thematically influenced.	*Characters and Elements Selected. Rules applied. Actions and Events chosen. All Influenced and vetted by Director Agent or Author rule set.* Thematically influenced selections of elements and actions. Thematic vetting by director/author.	
	Author Centric	*Material generated or collected.* Thematically influence which resources collected/generated.		*Populating/Building Structure.* Influence selection and positioning of resources. Influence structure built.	

Fig. 4. Summary of potential integrations of a thematic system with narrative generation

thematic involvement at the story level (the initial generation of elements) is important to assure a wide variety of available themes. However, in order to assure they are exposed correctly involvement at either the plot or presentation level would be necessary as well. Similarities between parts of plot generation for both approaches and existing implementations of the thematic model make plot generation the easier option. The constraints surrounding thematic involvement at the initial story level however do not necessarily rule it out as a possibility for integration as compromise approaches may allow for thematic elements to be introduced to a story using a director agent along side complex characters.

With an initial exploration of the issues complete, the future of this work is experimentation with integrating thematics and narrative generation. There are still also many remaining questions surrounding this process such as how the thematic scoring would be balanced in an effective way so as to include themes without spoiling a narrative and also how the effectiveness of a resulting system could be evaluated.

References

1. H. Al-Khalifa and H. Davis. Folksonomies versus automatic keyword extraction: An empirical study. *IADIS International Journal On Computer Science And Information Systems (IJCSIS)*, 1:132–143, 2006.
2. H. Alani, S. Kim, D. Millard, M. Weal, W. Hall, P. Lewis, and N. Shadbolt. Automatic ontology-based knowledge extraction and tailored biography generation from the web. *IEEE Intelligent Systems*, 18:14–21, 2003.
3. M. Alberink, L. Rutledge, and M. Veenstra. Sequence and emphasis in automated domain-independent discourse generation. In *Information systems*, pages 1–10, 2003.
4. R. Barthes and L. Duisit. An introduction to the structural analysis of narrative. *New Literary History*, 6:237–272, 1975.
5. H. Berghel. Cyberspace 2000: dealing with information overload. *Commun. ACM*, 40(2):19–24, 1997.
6. M. Bernstein. Card shark and thespis: exotic tools for hypertext narrative. In *Proceedings of the twelfth ACM conference on Hypertext and Hypermedia*, 2001.
7. M. Cavazza, F. Charles, and S. J. Mead. Agents interaction in virtual storytelling. In *Intelligent Virtual Agents, Springer Lecture Notes in Artificial Intelligence 2190*, pages 156–170. Springer-Verlag, 2001.
8. P. DeBra, A. Aerts, B. Berden, B. de Lange, B. Rousseau, T. Santic, D. Smits, and N. Stash. Aha! the adaptive hypermedia architecture. In *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 81–84, 2003.
9. C. Hargood, D. Millard, and M. Weal. A thematic approach to emerging narrative structure. In *Web Science at Hypertext08*, 2008.
10. C. Hargood, D. Millard, and M. Weal. Using a thematic model to enrich photo montages. In *Proceedings of Hypertext 09*, 2009.
11. M. Lebowitz. Planning stories. In *Program of the Ninth Annual Conference of the Cognitive Science Society*, volume 9, pages 234–241, Seattle, Washington, 1987. Cognitive Science Scoiety Conference.
12. M. Mateas and A. Stern. Faade: An experiment in building a fully-realized interactive drama. In *Game Developers Conference*, 2003.
13. M. McQuillan. *The Narrative Reader*. Routledge, London, 2000.
14. J. R. Meehan. Tale-spin, an interactive program that writes stories. In *In Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 91–98, 1977.
15. M. O. Riedl and R. M. Young. Character-focused narrative generation for execution in virtual worlds. In *Proceedings of the International Conference on Virtual Storytelling*, pages 47–56, 2003.
16. M. Theune, S. Faas, A. Nijholt, and D. Heylen. The virtual storyteller: Story creation by intelligent agents. In *TIDSE 2003: Technologies for Interactive Digital Storytelling and Entertainment*, 2003.
17. B. Tomashevsky. *Russian Formalist Criticism: Four Essays*, chapter Thematics, pages 66–68. University of Nebraska Press, 1965.
18. R. Young. Story and discourse: A bipartite model of narrative generation in virtual worlds. *Virtual Reality*, 2003.

Versioning in Adaptive Hypermedia

Evgeny Knutov, Paul De Bra, Mykola Pechenizkiy
Eindhoven University of Technology, Department of Computer Science
PO Box 513, NL 5600 MB Eindhoven, The Netherlands
debra@win.tue.nl, {e.knutov, m.pechenizkiy}@tue.nl

Abstract. This paper presents an approach that uses the terms, operations and methods of versioning applied to the Adaptive Hypermedia (AH) field. We show a number of examples of such a use which helps to facilitate authoring, managing, storing, maintenance, logging and analysis of AHS behaviour, providing extensive flexibility and maintainability of the systems.

Keywords: Adaptive Hypermedia, Versioning Systems, Revision Control, Authoring, User Model Updates, Context Changes.

1 Introduction

Versioning (also known as revision control, source control or code management) is the management of multiple revisions of the same unit of information. It is commonly used in engineering and software development to manage the development of digital documentation/code that are usually produced and maintained by a team of software developers. Changes to these pieces of information are usually identified by incrementing an associated version and binding it historically with the person, group or system making the change. On a large scale it is becoming the only way to keep track of all the changes, perform roll back operations if needed, merge modifications between different systems, individual users and groups and facilitate provenance analysis.

It is becoming obvious that tracking all the changes and operations over evolving structures makes reasonable and feasible to apply versioning methodologies in adaptive hypermedia systems. In this paper we will consider basic principles and operations of revision control applications in Adaptive Hypermedia Systems (AHS). We will not only start investigating the use of versioning for designing the structure and content of AHS applications but also take a look at the user model evolution in a versioning context.

2 Background

Engineering version control systems emerged from a formalized processes of tracking document versions. The main idea was to give a possibility to roll back to an early state of the document. Moreover, in software engineering, version control was

introduced to track and provide access and control over changes to source code. Nowadays it is widely used to keep track of documentation, source files, configuration settings, etc.

As a system is designed, developed and deployed, it is quite common for multiple versions of the same system to be deployed with minor or major differences within different environments, furthermore system configuration/settings, state or/and context evolve which results in multiple co-existing system versions as well.

If we look at the revision control system as a system used to deliver a certain version of source code, documents, tables, or any file type or folder to a particular user at a given time or on request it may resemble delivering adapted content to a user in an Adaptive Hypermedia System. Of course version control systems are straightforward and don't take into account any user model or context changes, don't have any semantic or proof layers or any conceptual knowledge representation, but on the other hand provide very flexible techniques to deliver selected content in a certain context to a designated user, and facilitate methods to save and track system changes in a very efficient way.

At the same time AH systems store and process a lot of information, which is highly sensible to a context data and vice versa - context data itself, is highly dynamic and influences the adaptation process. Thus tracking AHS alterations and environment changes is a tightly connected task which requires storage and computational resources.

3 Revision Control in AHS

Hereafter we consider basic aspects of versioning in application to AHS, provide analysis of core terms and operations in parallel with AH. We take up major types of changes in a context of AH systems, models and process flow, thus coming up with a taxonomy of AH versioning techniques, types of changes and applications areas. As a result we come up with two examples typical for the AH field: one is an e-learning application providing an adaptive course and the other is a recommender system. We encapsulate these examples in a versioning context.

3.1 Types of Changes and Versioning Flow

In a layered structure of a generic AHS [7, 8] we presume that each layer is responsible for "answering" a single adaptation question [3], like Why?, What?, To What?, Where?, When and How?. Each layer may have its own data structure and thus also have its own way of dealing with evolutionary and versioning concepts of the system. This versioning structure in its turn can be devised and maintained using different technologies varying from a source control application to a historical Data Base. We don't investigate in the paper which technology is best suited for each of these layers (or questions), however we will speculate on this question, suggesting solutions. At the same time we take up basic terms and concepts of versioning and look at them through the eyes of Adaptive Hypermedia. In figure 1 we sketch the core components of a layered AH architecture and provide annotations with use-cases. We

will first discuss versioning terms and operations in section 3.2 and then consider most of these use-cases in detail in section 3.3 through examples.

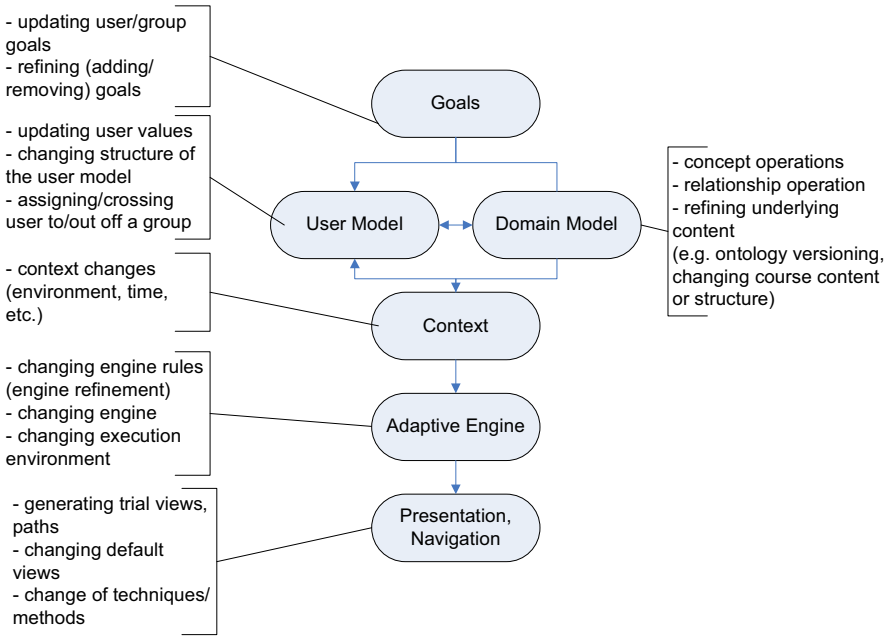

Fig. 1 Versioning process highlights

In figure 2 we outline a versioning taxonomy distinguishing types of changes, applications of versioning methodologies and we anticipate the set of potentially suitable versioning technologies. There are two major types of changes: structural and value changes. Both may take place in every layer/sub-system of AHS. For instance, refining the Domain Model structure and changing the underlying content or updating User Model values versus adding new properties to the model. Throughout the adaptation process, from authoring to presentation generation we anticipate the following versioning application areas:

Authoring – changes made by authors, to create, refine, update the structure and value parts of the adaptive application (usually these are the Domain Model and the Adaptation Model).

UM operations/updates – these refer to the user behaviour in the adaptive environment, tracking changes of the user knowledge, goals, preferences, achievements throughout system usage. Our UM versioning approach corresponds to ideas discussed in [4] and can serve as a back-end solution for ubiquitous user modeling.

Context changes – keeping track of context changes of the whole system as well as the context of a particular model, value or operation.

System deployment and analysis – changes done to an AHS to deploy the system in different environments, usually performed by the domain experts, whereas taking up the specifics of the domain to set up system logging, and analysis facilities.

We are not discussing technological aspects in depth, but can speculate about versioning (and related) applications to be used, and can foresee a number of well-known methodologies such as conventional source control systems and historical data bases for storing and querying for changes. OLAP technologies would suit to perform analysis. In addition a variety of custom-build systems (such as ontology versioning available in OntoView [2]) could also be considered.

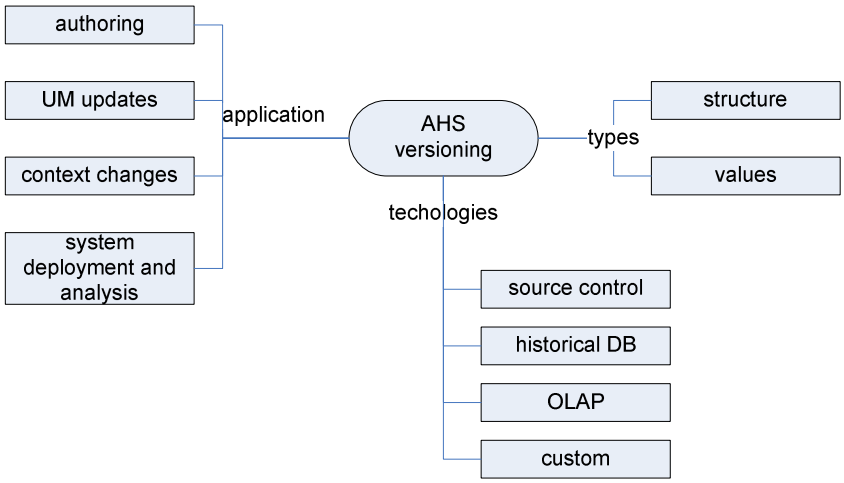

Fig. 2 AHS versioning taxonomy

3.2 Concepts and Operations of Revision Control

Here we would like to discuss the basic concepts of revision control and how they refer to the concept of evolution in AH systems. We will try to give a clear idea of how a particular concept or operation can be applied and will draw parallels with evolution and changes in AH through examples.

3.2.1 Basic versioning terms

Baseline – an approved version, usually the most stable, from which all the consequent versions are derived. In the AH field this is usually the core application, consequently used in different environment settings, within different user groups and thus being tuned up for those particular settings, which results in evolution and therefore new versions of the application. If we talk about **UM** – then the default version with which the user started can be considered as the baseline. In case of further analysis baseline is always the point of comparison of the application functionality and stability.

Branch – an instance is considered to be branched at a point in time, when from that time and on, two concurrent but independent versions of the same instance co-exist. As we have mentioned above, any AHS may result in a new set of settings or being used by a completely different group of people, thus a tune-up of the system will be required to meet the new settings or requirements. In these terms every AH system or model emerges in a branch of the initial baseline, representing a new authored version of the application (model/sub-system). Hence these systems may co-exist.

Branch can be also used as a prior concept of the User Model update, presentation generation or any other functionality which preview or try-out is desirable to analyze system behaviour. Even though the branch itself may not be used in the forthcoming version or update, a preview of a presentation or a try-out version of the application or engine will help to identify, observe and analyze the "***what-if***" case, and with the successful outcome commit branched changes, label as a new baseline or a head revision or merge them. This can be applied in terms of UM as well, considering the case when committing user properties to the latest application state is not desirable, however would be advantageous to see what can happen with the user in the new application environment. Though this UM case deviates from the original concept of the branch in a source control, it still uses the same principle of a concurrent instance, and in our case it is mapped on a context of UM in AH.

Head – the most recent commit, in other words the most latest version of the application/model existing on a single or multiple branches, with the latest functionality aspects prescribed to this branch(es). For UM the head revision is always the latest state of the model with all the corresponding values being updated to the latest state in the application.

Commit – applying changes to a central repository (branched or baselined) from one or multiple working sources. Usually the authoring and set-up processes require a number of experts, correspondingly a number of working instances which should emerge in a baseline version which in turn will require to commit all the authors' changes. In User Modeling we consider committing the changes to the user model values. At the same time not all the latest user changes should be committed to the user model in order to provide system stability (for example as it was done in AHA! system [5]).

Conflict – conflict occurs when the system can't reconcile changes made to the same instance of the structure. E.g. when the same concept properties are changed by two different authors, trying to commit them in the same course may result in a conflict which will require reconsideration of a concept structure and manual interference. Though conflict resolution may not guarantee such a property of the AH system as confluence, we can anticipate that versioned and annotated structures of the Domain or Adaptation model will be helpful in confluence analysis.

Change (difference, delta) – represents a specific modification under the version control. Though the granularity of change may vary from system to system, change always represents the basic structure of versioning concept. Changes evolve from the baseline, are branched, become new baselines, are committed, then resolved and merged. Depending on AH system, different instances can be changed (concepts, relationships, properties, content, context, rules or even a complete model or sub-system).

Moreover considering changes – is the best way to compare system functionality and settings, which is very important for analysis of inherited and evolved (branched – in terms of version systems) functionality of AHS.

Merge – in conventional source control systems – merge usually refers to a reconciliation of multiple changes made to different copies of the same file. In AH merge can be considered as part of an authoring or set-up process, where all the development and changes from different authors or domain experts come together. In general we can perform the merge operation over concepts, relationships, properties, individual user properties and user groups, ontologies, rule sets, etc. As a result merge provides a clear picture how (from which sources) the concerned concept or a property has emerged, facilitating application playback and analysis of changes. In applications where merge regulates UM values this operation can be granted to the user as well.

Resolve – is a user intervention in conflict resolution of 2 or more conflicting versions. We refer this to the authoring stage and let the experts decide on the outcome, or facilitate end-user resolution when dealing with UM.

Tag, Label – refer to an important snapshot of the system or its part in time implementing a certain functionality which has to be marked/tagged to settle a new state of the application.. This may be a stable Domain Model structure or a new user group with common interests (for instance).

At the same time an explicitly labeled structure of the model may be used in the context of OLAP (On-Line Analytical Processing). We will elaborate on this case below.

As we conclude, all aforementioned concepts of versioning and source control as they are applied purely in software engineering and document control can be represented in terms of Adaptive Hypermedia Systems, facilitating system flexibility and extensibility

3.2.2 Versioning operations

Having considered the basic concepts of versioning in the context of AHS, we can come up with the following classes of operations which will reflect typological and structural types of changes. The following taxonomy of operations was proposed in [2] in application to ontology evolution, however here we're trying to extend and elaborate it in terms of Adaptive Hypermedia. Hereafter we describe the properties of versioning operation.

Transformation – is usually a set of actual changes taking place over the evolving structure. These may be: changing properties of concept, classes (in case of ontologies), adding or removing concept structures.

Conceptual changes (may include concept and concept relationship changes) will refer to conceptual and abstract changes of the representation, structure and relationships within the AH system. This type of changes includes changes of types, relations, conceptual representation of a knowledge. It may also include relations between constructs of two versions of the model. Conceptual changes should be always supervised and carried out manually.

Descriptive changes – usually deal with metadata describing the reason, intentions, author's credentials, etc., regarding the new version of the model,

ontology, relation or a concept. Descriptive changes don't contribute to the actual change, however may help to reason over different versions of the same instance.

Context changes - will describe the environment in which the current update was made and in which it is going to be valid. At the same time all the environment settings for a particular change can be considered as a context change. E.g. changing a concept in the Domain Model we can consider the state of all the concept relationships as well as some other Domain specific information as a local context of this change. At the same the complete system environment is a context of embedding a new adaptive engine or re-organizing user groups. As we can see from these examples – context changes can be and usually are the most space and effort consuming, moreover domain experts usually analyze all the modifications in terms of contextual changes to capture the complete picture of a particular change.

4 Use-Cases

In order to demonstrate our conceptual view of versioning in AH, we present two examples that are typical for the AH field.

4.1 E-Learning application use-case

Let's consider the following: the same discipline is taught in two different universities, where adaptive courses Course 1 and Course 2 on this subject are provided to the students (see fig. 3). After one year, a curriculum plan in the first university (University 1) changes and one of the teachers is asked to change (refine) the course navigation structure and some of the content which results in a new version of the course (Course 1 ver.2). At the same time one of the students moves from the second university (University 2) to the first one (University 1), having received the credits for a first part of the course (Course 2). The aforementioned teacher is creating a new version of the course, reusing the complete structure and introducing only designated changes (in navigation structure, updating adaptive engine with the associated changes in rules and changing content). Thus the initial version of the navigational structure is preserved to trace back, compare and analyze how the user behaviour changes (paths, clickstreams). At the same time the student's User Model (which has been created in another university) (UM ver.1) emerges into a new version (UM ver.2), providing the student and his supervisor the possibility to compare basic concepts of Course 2, perform reconciliation and merge the corresponding UM values (of the results that the student has achieved taking the similar course in another university), and at the same time keep the history of his studies. On the other hand the automatic concordance is also possible if only the changes made in the course are corroborated with the additional descriptive information (is accompanied by the descriptive change (see section 3.2.2)) which contains extra explanations how to reason and update values (in this case UM values) due to the evolved course structure and/or content.

The complete picture of courses updates and user (student) development becomes transparent and the course instructor may trace back all the changes done to the

course, re-use, update, merge any particular part without creating anything from the scratch. He can analyze differences in and between courses.

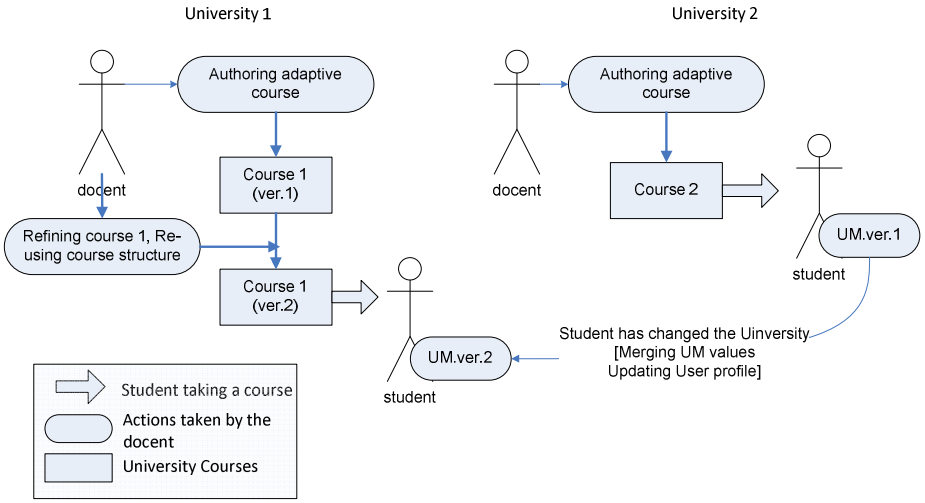

Fig. 3 Versioning adaptive courses

4.2 Recommender system use-case

In our second use-case we would like to outline the advantages of applying revision 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 getting recommendations) corresponds to a change in the User Model, 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 the user model state at that point in time and what has influenced the recommendation). The same happens with the recommended items, which evolve, can be modified and in general can have their own history of ratings.

If the user moves from one system to another, he will face a problem of the user profile alignment (like in the previous use-case sec. 4.1), where the history of ratings that he made may help to resolve conflicts. On the other hand labeling and tagging of user and system behaviour may become a basis for 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 multidimensional conceptual view and apply 'slice and dice' operation as the most

essential analysis tool in this respect, instead of simple system or user logging and then applying complicated and time-consuming extraction algorithms. This will facilitate quick extraction of rating information of a particular date or timeframe, or for example the set of users who rated a certain product a year ago. The same approach can be undertaken to version rated items in order to trace back the changes and associated ratings (as well as corresponding UM values) if the product or its description or properties change. In figure 4 we briefly summarize versioning aspects of the recommendation use-case.

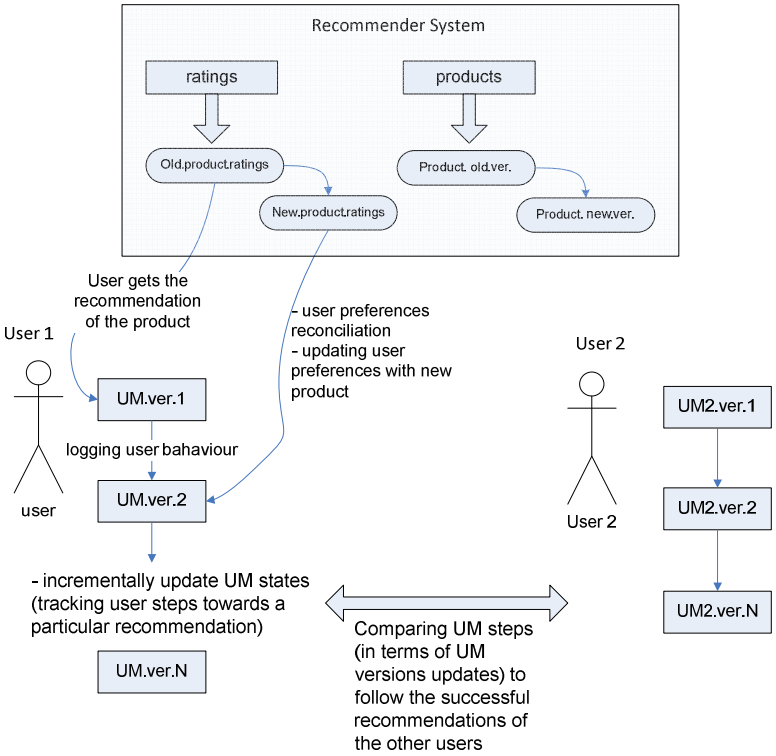

Fig. 4 Versioning in Recommender system

To conclude this use-case we would like to outline the ideas and principles used in a context of Recommender System. A new modification of product in the stock can be designated by a new version, it inherits corresponding properties and ratings. Versioning makes possible for products (product concurrent versions) to co-exist. It also helps to store product updates and a context of these changes efficiently. Thus we can create concurrent instances of products and ratings within different applications, inherit ratings, properties and merge these changes. The User Model versioning allows to analyze user step-by-step behavior (incl. user trend, etc.), compare changes with other users in order to provide similar recommendation patterns in a case of successful outcome.

4.3 Use-cases conclusions:
For the aforementioned use-cases we would like to conclude with a summary of approaches we suggested in the introduction and outline the advantages of versioning:
Authoring – versioning helps to create, maintain and re-use concurrent versions of an application, model or a particular property and value (e.g. course and a corresponding content), saving authoring effort.
Storing – versioning offers an efficient way to store changes, annotate them, label, present in a hierarchical structure. This saves space for a large scale system and keep track of the system history.
Maintenance – structured changes and a set of basic concepts and operations (merge, resolve, commit, tag/label, head and branch) are sufficient to maintain and reconcile application conflicts, create concurrent versions or comprise functionality implemented in different systems in one application (merging changes from different branches).
Logging – logging incremental changes (of the application, user model or a context) provides playback possibilities and serves as a ground of system analysis. Logging in terms of UM updates (or steps taken) will provide a basis for comparison of behaviour patterns and advance provided recommendations.
Analysis – versioning facilitates analysis of the step-by-step system and user behaviour to identify trends; tagging and labeling can facilitate more complex analytical approaches (such as OLAP) providing the required structure and description of the data; hierarchical incremental logging results in a clear and transparent structure of system changes and an overall evolution picture.

5 Conclusions and Further Work

In this article we tried to map a conventional versioning approach onto the field of Adaptive Hypermedia, providing clear parallels in terms of versioning concepts and operations. We tried to make first steps in this direction in order to show advantages of this approach and outline the perspective of applying these techniques and methodologies in order to facilitate the transparency of AHS evolution through versioning.

Considering further work directions 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 the framework and come up with the detailed structure and operations to devise these versioning facilities.

Acknowledgements. This research is supported by NWO through the "GAF: Generic Adaptation Framework" project.

References

1. Altmanninger, K., Gerti Kappel G., Kusel, A., Retschitzegger, W., Schwinger W., Seidl, M., Wimmer, M., AMOR – Towards Adaptable Models Versioning. In 1st International Workshop on Model Co-Evolution and Consistency Management in conjunction with Models'08, 2008.
2. Klein M., Fensel D., Kiryakov A., Ognyanov D., Ontology versioning and change detection on the Web. In Proceedings of In 13th International Conference on Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web. Berlin, Heidelberg: Springer, pp. 247-259, 2002.
3. Brusilovsky, P., Methods and techniques of adaptive hypermedia, User Modeling and User Adapted Interaction 6(2-3), pp. 87-129, 1996.
4. Heckmann, D., Ubiquitous User Modeling, PhD Thesis. Universitat des Saarlandes, 2006.
5. De Bra, P., Calvi, L., AHA! An open Adaptive Hypermedia Architecture. *Hypermedia*, 4(1), pp. 115-139, 1998.
6. Herlocker, J., Konstan, J., Terveen, L., Riedl., J., Evaluating collaborative filtering recommender systems. ACM Transaction on Information Systems, vol.22, No.1, pp.5-23, 2004.
7. De Bra, P., Houben, G.J., Wu, H., AHAM: A Dexter-based Reference Model for Adaptive Hypermedia, Proceedings of the ACM Conference on Hypertext and Hypermedia, pp. 147-156. 1999.
8. Cristea, A., De Mooij, A., LAOS: Layered WWW AHS Authoring Model and their corresponding Algebraic Operators, The Twelfth International World Wide Web Conference, Alternate Track on Education, 2003.