PROVIDING SERVICE-BASED PERSONALIZATION IN AN ADAPTIVE HYPERMEDIA SYSTEM

by

Michael V. Yudelson

Eng. System Analyst, Ivanovo State Power University, 2001

Candidate of Tech. Sci., Ivanovo State Power University, 2004

Submitted to the Graduate Faculty of

School of Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2010

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This dissertation was presented

by

Michael V. Yudelson

It was defended on

September 17, 2010

and approved by

Daqing He, Associate Professor, School of Information Sciences

Heiko Spallek, Associate Professor, School of Dental Medicine

Michael Spring, Associate Professor, School of Information Sciences

Vladimir Zadorozhny, Associate Professor, School of Information Sciences

Dissertation Advisor: Peter Brusilovsky, Associate Professor, School of Information Sciences

PROVIDING SERVICE-BASED PERSONALIZATION IN AN ADAPTIVE HYPERMEDIA SYSTEM

Michael V. Yudelson

University of Pittsburgh, 2010

Adaptive hypermedia is one of the most popular approaches of personalized information access. When the field started to emerge, the expectation was that soon nearly all published hypermedia content could be adapted to the needs, preferences, and abilities of its users. However, after a decade and a half, the gap between the amount of total hypermedia content available and the amount of content available in a personalized way is still quite large.

In this work we are proposing a novel way of speeding the development of new adaptive hypermedia systems. The gist of the approach is to extract the adaptation functionality out of the adaptive hypermedia system, encapsulate it into a standalone system, and offer adaptation as a service to the client applications. Such a standalone adaptation provider reduces the development of adaptation functionality to configuration and compliance and as a result creates new adaptive systems faster and helps serve larger user populations with adaptively accessible content.

To empirically prove the viability of our approach, we developed PERSEUS – server of adaptation functionalities. First, we confirmed that the conceptual design of PERSEUS supports realization of a several of the widely used adaptive hypermedia techniques. Second, to demonstrate that the extracted adaptation does not create a significant computational bottleneck, we conducted a series of performance tests. The results show that PERSEUS is capable of providing a basis for implementing computationally challenging adaptation procedures and compares well with alternative, not-encapsulated adaptation solutions. As a result, even on modest hardware, large user populations can be served content adapted by PERSEUS.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION

Adaptive hypermedia (AH) is a field of research on the junction of hypertext/hypermedia and user adaptive systems. It evolved from static hypertext systems and local standalone adaptive systems into hypertext adaptive systems on the Web. From their inception in the 1990s up to this moment, adaptive hypermedia systems have come a long way and have become one of the major areas of human-computer interaction research. The growth of the field of adaptive hypermedia resulted in the creation of novel approaches to personalizing user information access (e.g. stretchtext, dimming fragments, adaptive link annotation, etc.) and development of state-of-the-art adaptive hypermedia systems (AHS).

The biggest impact of adaptation and personalization has been seen in education. When looking at the current state of research and development in the modern US IT field, Eric Benhamou and his co-authors (Benhamou et al., 2010) concluded that one of the highest returns-on-investment would be "personalized and collaborative educational tools for tutoring and just-in-time learning." Howard Gardner in his article in *Foreign Policy*(Gardner, 2009) renders personalized education as "the next big thing." Personalization (and adaptation as one of the frequent synonyms for it) starts to settle in the public perception as one of the state-of-the-art technologies, capable of advancing modern education significantly.

Despite the initial enthusiasm, after nearly two decades of research we are still only wishing that adaptive hypermedia systems and adaptive educational hypermedia systems

(AEHS) were capable of reaching their full potential and reaching large user audiences. In fact, this is one of the major pieces of criticism that AHS/AEHS as a field usually receives: The availability of adaptive systems is not as high as one may have wished. The explanation for the shortage of adaptive hypermedia systems is that for the first decade of AHS/AEHS research the focus was primarily on developing and evaluating novel adaptation methods. Only very recently has the interest started shifting to wider and more effective dissemination of adaptation support.

There are two principal ways of achieving wider dissemination of adaptive systems and adaptively accessible content to larger groups of users. The first is to support authors creating content in the case of a particular system. The second is to support developers in creating new AHS by reusing principal modules and components across several systems. While the first approach empowers a large group of professionals who do not need to possess only domain expertise and minimal hypermedia authoring skills, it ties the newly authored content to a particular AHS that often addresses only one domain. The second approach is cross-domain and cross-system by design, but reusing system components brings up tougher design questions and, potentially, performance challenges.

In this work we are taking the second approach, aiding the AHS developers. We are suggesting a novel solution aimed at providing adaptation for heterogeneous systems in a transparent manner. The essence of the proposed solution is to take advantage of the structural decomposition of the AHS – namely, take the adaptation functionality out of the AHS, package it into a standalone system, and provide adaptation on demand as a service. Utilization of such an adaptation functionality provider would let virtually any non-adaptive client system to become adaptive. Existence of such *server* of adaptation functionalities makes the task of creating new adaptive systems much easier than before by reducing development to compliance, and, as a

result, allows dissemination of adaptive content to larger user populations. Availability of a set of adaptation techniques to an unconstrained number of client applications promotes reuse and increases the value of each technique being reused.

In the rest of this chapter, we first will discuss several solutions that can enhance AHS/AEHS by increasing the amount of content accessed through adaptive interfaces and help reach a larger user population. Then we will discuss the solution of our choice and state the problem formally. This will include the goals and high-level hypotheses of the work. Finally, we will outline the content of the rest of the dissertation.

1.1 WAYS TO ENHANCE ADAPTIVE HYPERMEDIA SYSTEMS

There are several ways to help increase the volume of adaptively accessible content and the number of users it is available to. Below we mention the ones that we believe are the most potent and provide arguments for and against them.

1.1.1 Content Authoring Support Tools

The first way to enhance AHS is to build authoring tools aiding the process of content authoring for a particular system. There are more people capable of creating quality content (domain experts, instructors, etc.) than those capable of building quality AHS/AEHS. Empowering potential content authors would help significantly expand the volume of available material.

Authoring tools could serve several purposes. First, they offer a user interface for creating new resources that could be accessed in an adaptive way. Second, authoring tools could be used for editing and extending system infrastructure. For example, knowledge engineering

authoring tools address the single domain problem by aiding the development of the metadata structure for the content. Support for extending adaptation infrastructure to new domains could open opportunities for extensive content generation.

The advantage of the content authoring tools solution is that every man-hour spent on the development of the tools necessary is reciprocated manyfold. Existing research on authoring tools for AHS, e.g. (Aleahmad et al., 2008; Ritter et al., 1998; Sosnovsky et al., 2007) demonstrates the high potential of assisted interactive content creation.

The disadvantage is that authoring tools are tied to a particular AHS and are usually built for one domain. Despite the relative ease of creation, the new content would be limited to a particular system and one domain. To create new content across several systems and/or domains requires extensive knowledge expertise. In addition, the standards and processes of content quality self-control serve more as an inhibitor to the content creation process, rather than a quality assurance. Peer quality control (Cafolla, 2006; Denny et al., 2008; Gehringer et al., 2007; Hsiao & Brusilovsky, 2010; Masters et al., 2008) requires time, additional effort, and motivation and/or incentives.

Another important limitation of the authoring tools is that, despite numerous attempts to aid content designers, satisfactory support for authoring adaptation by non-programming and non-adaptation-expert users is yet to be achieved. In educational AHS, for example, there are quite a number of tools for aiding teachers in creating course materials (Aleahmad et al., 2009; Brusilovsky et al., 2006; Brusilovsky et al., 2005; Murray, 1999; Turner et al., 2005) but there are none that would help build adaptive support for the authored content.

Lastly, supporting AHS content designers would help generate more material for each of the systems that the authoring tools are provided for. However, authoring support would not

improve existing adaptive solutions, nor create new ones. Content creators, who usually are hardly familiar with adaptive technologies, are only capable of point-and-click adaptation configuration.

1.1.2 Development Support Tools

A second way to enhance AHS is to support the developers. This support can often come in the form of frameworks and architectures where one or several modules critical to the operation of an AHS are implemented in a centralized way and shared among a number of systems. Many of the modern AHS frameworks and architectures employ structural decomposition.

In the past, every new adaptive hypermedia system was built from scratch, which was time- and effort-consuming. To optimize the development of the new AHS, researchers decomposed the monolithic systems and reused fundamental components across several systems. First, decomposition attempts focused on the user models and user modeling functionality that became independent from the adaptive system before the age for hypertext and hypermedia (see Section 2.5.1 for more details). Later on, the introduction of the open corpus hypermedia systems made it possible to add new content at the run time, as opposed to closed corpus systems, where content is only added at the design time (see Section 2.5.2).

The advantage of the framework approach is the ease of creating new systems. The AHS designer who has a set of pre-engineered system modules can reduce part of the coding job to integration and configuration. Here modules could be reused in a number of different AHS. Several successful framework AHS approaches are known today, for example AHA! (Bra & Calvi, 1998) and Personal Reader (Dolog et al., 2004). In addition, supporting AHS designers in creating new mash-up-style adaptive systems from building blocks implies greater flexibility and can be the basis for the new hybrid adaptation approaches.

A disadvantage of the structural decomposition approach is that a great deal depends on how carefully module boundaries are outlined and communication protocols are defined. Carelessly designed modules would limit their usability and obstruct successful integration into numerous AHS's, only favoring a few. In addition to data exchange formats and protocols, one must consider how general each component is: how many of the known methods and techniques it supports.

Although the framework approach has been used for as long as the AHS field has existed, it has not been able to boost the development of new AHS by a significant margin. Arguably the major reason is that in almost all cases adaptation functionality is locked inside the AHS. Early attempts to reuse adaptation functionalities, for example APELS (Conlan & Wade, 2004) and Personal Reader (Dolog et al., 2004) favor reuse of adaptation within the boundaries of a single system or a framework, rather than open and transparent access to it.

1.2 PROBLEM STATEMENT, GOALS, AND HYPOTHESES

Despite the general attractiveness of the standalone adaptation provision idea, there are several questions we must answer. First, the design of this server should provide a good level of abstraction to support the majority (and possibly the entirety) of the known adaptation methods. Proof is necessary that a particular design of a standalone adaptation provider can successfully support a set of frequently used adaptation techniques and adaptation engines. These techniques and adaptation engines should form a representative subset. Namely, should be widely used in modern adaptive hypermedia systems and range in computational challenge.

Second, outsourcing adaptation functionality to an external system might present a computational challenge. Being disjointed from the host system, the adaptation methods no longer have instant access to the internal data of the AHS (hyperspace structure, user model data) that has to be acquired independently. As many adaptation techniques traverse large chunks of the available hyperspace and often require sizable portions of the user model to produce adaptation decisions, the performance of the adaptation engine has to be carefully appraised.

Third, given the highly probable growth of computational challenge, the standalone adaptation server should be compared to the non-encapsulated implementation of the adaptation in order to determine whether performance of the former degrades significantly as compared to the latter.

To answer these questions we need to achieve the following goals:

1. Propose the paradigm of the standalone adaptation provision – extracting the adaptation functionality and encapsulating it in a designed adaptation engine.
2. Build a prototype system implementing the suggested idea and realize a representative set of the adaptation techniques
3. Conceptually evaluate this server for the support of known adaptation techniques.
4. Empirically prove that the prototype system is capable of supporting a representative set of widely used adaptation technologies and reasoning engines ranging in computational complexity. This requires obtaining performance estimates for each of the techniques tested.
5. Determine the performance margins that an encapsulated adaptation implementation has over the suggested standalone implementation and whether it is at an advantage or at a disadvantage.

The high-level hypotheses of the work are as follows:

1. Providing adaptation functionality in an encapsulated manner is a conceptually viable idea. Namely, decomposing adaptation functionality is compatible with accepted de facto practices of separating expertise and labor when developing component-based systems on the field of adaptive hypermedia systems.
2. A standalone adaptation server is fully capable of supporting at least a representative subset of the known adaptation methods and techniques that are widely used and range in computational challenge theypresent.
3. Adaptation functionalities developed on the basis of a standalone adaptation server can be successfully deployed for a practical use on a small scale and exhibit acceptable performance characteristics on a large scale[1].
4. A standalone adaptation server as compared to an AHS with a more traditional adaptation should be comparable in terms of performance. The standalone server (a general-purpose system) is expected to lose to the traditional adaptation (a special-purpose system) with an acceptable margin.

[1] Practical small-scale use here is defined as ability to support a group of users the size of an academic class (e.g. up to 30 people) in a seamless and uninterrupted manner. Acceptable large-scale performance is defined as ability to offer uninterrupted support for hundreds to thousands of users. In both cases hardware and software of the machine where the standalone adaptation server is deployed should not exceed capabilities of a typical contemporary desktop machine.

1.3 DISSERTATION OUTLINE

The text of the dissertation will be organized as follows. Chapter 2.0 is devoted to the discussion of the related work. First, it covers adaptive hypermedia systems, methods of adaptive hypermedia, and adaptation reasoning engines. Then, we talk about the architecture of the AHS and discuss prior attempts at its structural decomposition. The chapter concludes with a discussion of methods for evaluating AHS.

Chapter 3.0 starts by presenting the general idea of implementing adaptation in a standalone server as well as a concrete implementation of this idea – the PERSEUS system. Conceptual evaluation of PERSEUS is given next, followed by discussion of adaptation techniques realized in it. A pre-study of PERSEUS ends the chapter.

Chapter 4.0 discusses the design of the main experiment and starts with a general overview. The chapter then talks about how the experiment was set up, how tested systems were configured, and what software and hardware were used. The chapter concludes with comprehensive, fine-grained experiment hypotheses.

Chapter 5.0 presents the analysis of the collected experimental data covering the results of the experiments. And finally, Chapter 6.0 discusses contributions, limitations, and future directions of the work.

2.0 BACKGROUND AND RELATED WORK

This chapter covers background and related work. First, an overview of adaptive hypermedia systems is given. Then, methods of adaptive hypermedia and adaptation engines are discussed. Architectures of AHS are discussed next, followed by an analysis of approaches to the decomposition of AHS and AHS component reuse. Finally, an overview of methods of AHS evaluation is presented.

2.1 ADAPTIVE HYPERMEDIA SYSTEMS

The first AH systems started to appear in the late 1980s to early 1990s. The first AHS – Hypadapter (Böcker et al., 1990; Hohl et al., 1996) – was developed as early as 1990. The adaptive hypermedia field took shape quickly and by 1996 (Brusilovsky, 1996) was ready to offer a plethora of techniques. In this chapter we will define what an adaptive hypermedia system (AHS) is, discuss various adaptation techniques and reasoning engines developed, and finish with a discussion of the architecture of AHS.

Adaptive hypermedia systems are a subset of user-adaptive systems in general. By definition, Adaptive Hypermedia Systems are "all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user" (Brusilovsky, 1996), p.88.

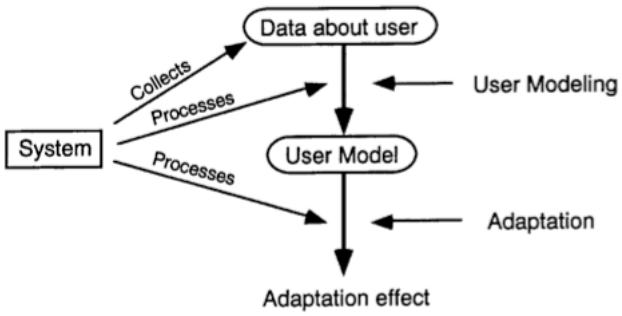


Figure 1. Outline of a typical adaptive hypermedia system (Brusilovsky, 1996)

A very basic sketch of an AHS is shown in Figure 1. As follows from the figure, an AHS should satisfy three basic criteria: It should be implemented using hypertext or hypermedia technologies; it should have a user mode; and it should be able to adapt its user interface based on the user model.

The major distinctive feature of an AHS is the kind of adaptation the system provides. The generally recognized classification of adaptation methods has been suggested in (Brusilovsky, 1996) and later refined in (Brusilovsky, 2001). The refined classification is shown in Figure 2. The classification taxonomy has two major branches of adaptation techniques: adaptive presentation and adaptive navigation support. Adaptive presentation techniques work on an intra-document scale – they deal with fragments of the hypermedia/hypertext pages. Adaptive navigation support methods work on an inter-document scale – they mainly operate on links to the content pages; the pages themselves are viewed as whole unbreakable entities. There have been several attempts to introduce changes to the classical taxonomy introduced by Brusilovsky.For example, two of the most recent ones are reported in(Bailey et al., 2007) and (Knutov et al., 2009).However, they only suggest refinements and do not affect the core of the classification schema.

The classification schema in Figure 2 organizes various techniques by what impact they have on the user interface of the adaptation system. In addition, adaptive systems might follow

different paradigms of implementing reasoning that drives the process of adaptation. We will continue our discussion of the adaptive systems by providing a survey of adaptation methods first and then will talk about various approaches to the realization of adaptive reasoning.

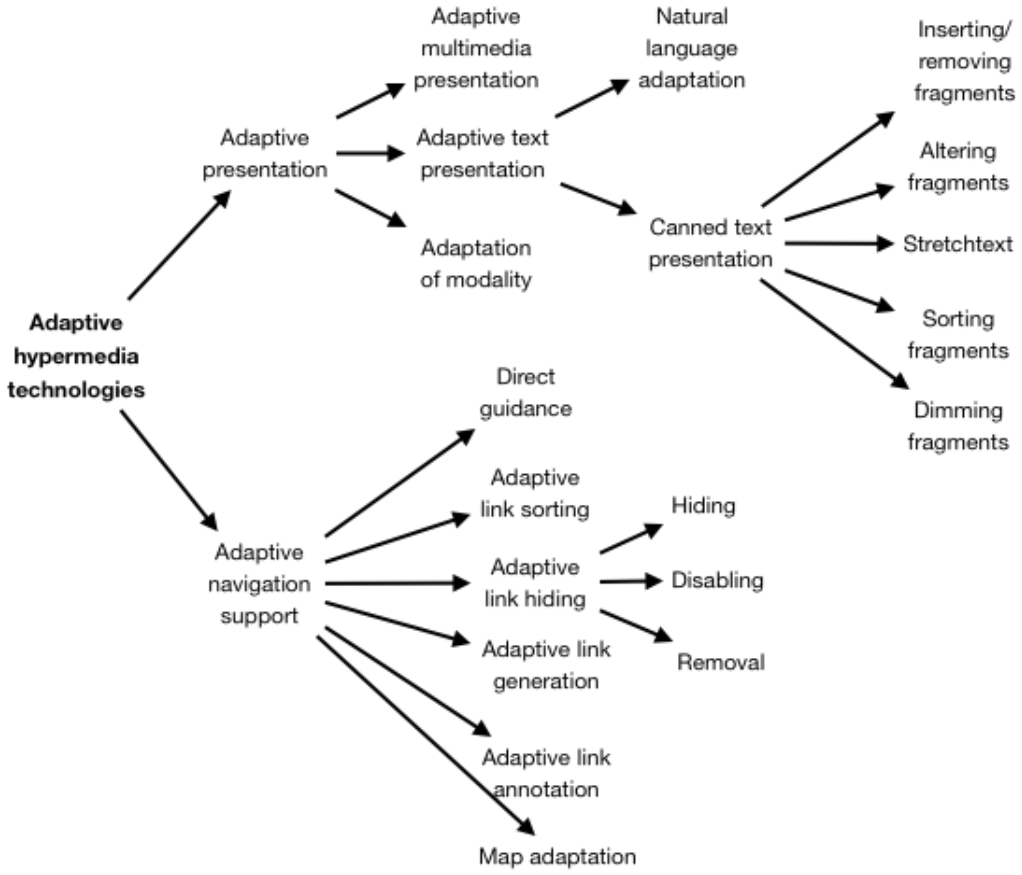


Figure 2. Taxonomy of adaptive hypermedia technologies (Brusilovsky, 2001)

2.2 METHODS OF ADAPTIVE HYPERMEDIA

In this section we will discuss the essence of various methods of adaptive hypermedia. Our discussion of the methods will be limited to the adaptive navigation support branch of the classification (Figure 2). The reason for this is the fact that methods of adaptive navigation

support are the most popular and widely used adaptation methods. They constitute a tangible subset of the taxonomy. Besides, as Bailey et al. (2007) rightfully mention, most of the methods in the adaptive presentation branch of the taxonomy could be reduced to one of the methods in the adaptive navigation support branch.

In the subsequent subsections we will cover the following methods: direct navigation, link sorting, link hiding, link generation, and link annotation. We will be talking about their pros and cons, as well as scopes of their applicability.

2.2.1 Direct Navigation

Direct navigation is one of the basic navigation support techniques, where at each point the user is presented with a limited set of adaptively selected "next best" options/links (usually one) (Brusilovsky & Pesin, 1998). Direct guidance does not necessarily entail adapting for a single step only. In the system KBS Hyperbook (Henze & Nejdl, 1999), a whole adaptive path is constructed, while the user is still given one "next hop" at a time.

Direct navigation has been shown to work best for beginner users, unfamiliar with the hyperspace and thus incapable of making informed navigational decisions (Brusilovsky, 2003; Weber & Specht, 1997). Several studies reviewed in (Brusilovsky, 2003), show that novices have problems dealing with alternative navigation choices. However, for experienced users, who do not follow the system's suggestions, direct guidance was found to have limiting utility. It is due to this problem that direct guidance, once popular in the early days of AH, has been largely replaced by other hypermedia techniques. Nonetheless, direct navigation still remains in demand, especially when curriculum sequencing is performed (e.g., in the area of Intelligent Tutoring Systems).

2.2.2 Link Sorting

Link sorting (sometimes called link ordering) gives more navigational options to the user. Here, link order is used to denote relevance of the linked documents. One of the first hypermedia systems to implement link sorting was Hypadapter (Hohl et al., 1996) back in 1990. One of the most well-known systems to employ link ordering is HYPERFLEX (Kaplan et al., 1993). HYPERFLEX used user-perceived link relevance to adapt link order. On every page, relevant links could be manually rearranged, which immediately causes user model update. Selection of a new search goal also influenced link order, as link rank was related to the goal as well.

Link ordering could significantly reduce navigation time and the number of navigation steps and bring users to what they were looking for more quickly(Kaplan et al., 1993). Link sorting works best in situations when the number of available links is quite large. Links closer to the top of the list automatically receive higher user attention. Good examples of such cases are modern search engines. On the other side, the order of links could be unstable: It might change every time the user revisits a particular page. This has been demonstrated to create a problem for some categories of users working with menus and toolbars (Debevc et al., 1996; Kaptelinin, 1993). As a result, link sorting is to be used carefully so that potential changes of the order do not hurt system usability.

There exist several contexts in which frequent link order changes do not pose a problem. One is adaptation to the long-term user characteristics, e.g. to the learning style (Kelly & Tangney, 2005). Every user would see a different list, yet, since long-term characteristics do not change, it is stable throughout the whole period of working with the system.

Another example of problem-free instability of the link order is the system where some or all pages have unstable sets of links. A good example of such a system is a class of systems

called collaborative resource gathering systems, such as COMTELLA (Cheng & Vassileva, 2005) and CoFIND (Dron et al., 2000). In collaborative resource gathering systems, users collect useful resources by posting links to thematic folders (links are sometimes removed from or moved between folders). The collection of links in each folder is hence unstable by definition. The same property is possessed by adaptive news systems that typically present links to fresh articles that are constantly added while links to older articles are constantly removed.

2.2.3 Link Hiding

The purpose of link hiding is to restrict the navigation space by hiding, disabling, or removing links to irrelevant pages. Reasons for rendering a page irrelevant could be many. For example, the page could be unrelated to the user's current goal, or the information on the page could be too advanced for the user to comprehend yet. Hiding reduces the complexity of the whole hyperspace and protects users from cognitive overload.

Educational systems have been the major area where link hiding has been intensively used and studied. Keeping too advanced or too simple material hidden and gradually shifting forward by masking well-mastered content and uncovering new resources as the user progresses is a popular educational approach that is easily implemented using adaptive link hiding.

Early AHS used a simple technique of link hiding – removing the link and the anchor from the page completely. ISIS-Tutor (Brusilovsky & Pesin, 1998) showed very few links when the user had only started to work with the system. As the user's knowledge grew, the number of visible links increased. The rationale of this approach was to focus user attention on the links to the content that the user was ready to comprehend. The links to the content that answers current goals and/or are important in a given context form a recommended zone (Brusilovsky & Pesin, 1998), (Vygotsky, 1978).

The approach used in ISIS-Tutor was later called link removal(Bra & Calvi, 1998).Three principal link features were defined for it: the anchor, the visual indication, and the functionality. According to these features, link hiding only removes the visual distinction of the link, but preserves functionality. This technique adaptively incapacitates the links by making them indistinguishable from the surrounding text (but still functional). Link disabling removes the functionality – the ability to follow the link and open the connected page. A number of studies of link hiding show that it works best when links are adaptively enabled. Disappearance of previously available links has been found to lead to user frustration.

2.2.4 Link Annotation

Link annotation is a group of methods that modify the way a link is presented to the user to convey additional information about the document behind that link so that the user can make an informed decision whether to traverse the link or not. ManuelExcel (La Passardiére & Dufresne, 1992), for example, introduced link annotation in the form of icons. ISIS-TUTOR (Brusilovsky & Pesin, 1998) changed the color and the intensity of the anchors. Hypadapter (Hohl et al., 1996) altered the size of the anchor text. Different kinds of verbal annotations can be shown next to the anchor (Geldof, 1998) or in the status bar (Brusilovsky et al., 1998), or as a pop-up each time the mouse is over the link (Zellweger et al., 1998).

Out of all these approaches, the most widely used one is icon annotation. There exist several approaches when it comes to the choice and design of the annotation cues as well as decision-making behind selecting the appropriate cue in each situation. In ELM-ART, the authors use the traffic light metaphor (Weber & Brusilovsky, 2001). Here the color of the bullet next to a link denotes user readiness to comprehend the linked material. In Syskill & Webert(Pazzani et al., 1996) annotations are based on peer users' feedback. Here an aggregated

user opinion is displayed in the form of an icon. In the system called NavEx (Yudelson & Brusilovsky, 2005) annotations are based on the model of user knowledge. Links in NavEx can be in normal font or bold face, and a progress bar icon is added to show amount of user work with the resource. In the social system Knowledge Sea II (Farzan & Brusilovsky, 2005), annotations visualize the density of users' click stream through the linked resource. Click traffic is expressed in terms of link background color: the darker the color, the larger the amount of traffic.

The major benefits of AHS implementing adaptive link annotation are an increased amount of student work (even with non-mandatory content), faster more optimal traversal of the hyperspace, more pronounced non-sequential navigation, and increased user satisfaction and trust in the system (Masthoff & Gatt, 2006). The effect has been found in several systems, including ELM-ART (Weber & Brusilovsky, 2001), QuizGuide (Brusilovsky & Sosnovsky, 2005), NavEx (Yudelson & Brusilovsky, 2005), etc. In a comparative study of two non-adaptive systems, WebEx and QuizPACK, and two adaptive systems, NavEx and QuizGuide (both implementing link annotation) (Brusilovsky et al., 2009; Brusilovsky et al., 2006), the authors have shown that adaptive link annotation has given a significant boost to students' motivation. The motivation increase resulted in manifold growth of the amount of work by users of adaptive systems. The growth could be traced in time spent working with the adaptive systems, number of page visits, number of user sessions, portion of hyperspace covered, etc.

2.2.5 Link Generation

Link generation, unlike the rest of the adaptive navigation support techniques that adapt presentation of pre-authored links, adds new links to the pages. Link generation is a fairly new navigation support technology. In the early days of the closed corpus systems, the number of

hyperspace nodes was quite small and there was no need to introduce new links. Early examples of link generation appeared in 1996 (Brusilovsky et al., 1998; Weber & Brusilovsky, 2001; Yan et al., 1996). As the corpora of the AH grew bigger, link generation became more popular.

There are three kinds of links generation: 1) discovering new, useful cross-document links and adding them to the permanent link set, 2) generating links for the similarity-based navigation between pages, and 3) recommending links dynamically based on the current context and the user (e.g. goal, knowledge level, interests, etc.). The first two kinds of link generation are typically non-adaptive.

Examples of the cross-document and useful link discovery could be found in (Bollen & Heylighen, 1998; Lutkenhouse et al., 2005; Yan et al., 1996). The authors of (Yan et al., 1996), for example, dynamically add links based on the navigation offprint of the user. Users are first classified by their information needs. Based on the user class, frequently accessed resource links are presented.

An example of adaptive similarity-based navigation is a system called ELM-ART (Brusilovsky et al., 1996). Here links are generated for similarity-based navigation. New links to the glossary items are added to the page if the user does not have a sufficient level of knowledge of the underlying concepts. In a similar system (Debevc et al., 1997) new links are added based on document similarity and frequency of document access mined from user logs.

Adaptive hypermedia systems with dynamic link recommendation are often hard to distinguish from currently very popular Web recommender systems. An important distinction is that adaptive hypermedia systems usually provide guidance given current user position in the hyperspace and focus on the interface. Web recommender systems focus on the underlying recommendation technology and often disregard user location, focusing on presenting items that

are of interest in general. Undeniably there exist systems that fit both classification criteria, for example Amazon.com[2], which recommends links to products that were purchased or inspected by other customers who viewed a particular product.

A system called ELFI employs a number of item-based and feature-based collaborative filtering techniques to generate new links matching explicitly or implicitly established user interests (Schwab et al., 2000). Another example of link generation adapted to users' knowledge is the system ALICE (Kavcic, 2004), an electronic Java programming textbook. There are no stable links between book chapters. Links are dynamically added in the end of each section and are separated into three groups: next possible units, required background units, and learned units. Evaluation of ALICE suggests that those users who follow links to the recommended units have better test scores.

2.3 ADAPTATION REASONING ENGINES

Adaptation methods that we have discussed in the previous section are only one part of the adaptation phenomena. They regulate the user interface part, a result of the adaptation. The implementation of the adaptation logic is defined by the so-called adaptation reasoning engines (Brusilovsky, 2001). The process of adaptation itself is often hard to cleanly separate from the process of user modeling (Brusilovsky, 1996). Arguably, one could say that low-level maintenance of the user modeling values is the realm of the user modeling, while high-level management of the user model data (e.g. propagation, aggregation, classification, etc.) and production of adaptation decisions areprerogatives of the adaptation reasoning engine.

[2] http://www.amazon.com

Reasoning engine implementation often depends on a specific type of user model representation that in its turn is closely connected with a problem domain. Adaptive hypermedia systems involve a wide range of reasoning engines, including but not limited to:

- **Social** reasoning, which operates on individual and group summaries of user interaction,
- **Topic-based** reasoning, which utilizes coarse-grained conceptualization of the problem domain,
- **Concept-based** (in some contexts **keyword-based**) reasoning, which uses finer-grained problem domain structures, such as taxonomies, ontologies etc.; keyword-based is only slightly different in the method of producing first-class objects (concepts) and their semantic meaning.

Table 1 presents a cross-table of three reasoning engines most widely used in AHS (columns) and adaptation techniques (rows), where cells contain references to exemplary AHS. Despite the intuitive connection between the choice of the engine and the computational/processing challenge, the relation is not always that straightforward.

Reasoning engine columns in Table 1 are ordered left to right, from the least computationally complex (social) to the most computationally complex (concept-based). The complexity here comes from the amount of objects that need to be processed to produce an adaptation decision for each of the adapted resources. The memory and computation time factors are usually contingent on that.

Table 1. Selected adaptation reasoning engines and adaptation engines in real systems

		Reasoning engine (complexity)		
		Social (low)	Topic-based (medium)	Concept-based (high)
Adaptation technique	Direct navigation			SQL Tutor (Mitrovic, 2003), Andes (Gertner & VanLehn, 2000), CTA (Anderson et al., 1995)
	Link sorting, generation	GroupLens (Sarwar et al., 2001)		InterBook (Eklund & Brusilovsky, 1999)
	Link hiding			AHA! (Bra & Calvi, 1998)* NavEx (Yudelson & Brusilovsky, 2005), QuizGuide (Sosnovsky et al., 2008), KBS-Hyperbook (Henze & Nejdl, 1999)
	Link annotation	KnowledgeSea II (Farzan & Brusilovsky, 2005)	QuizGuide (Sosnovsky & Brusilovsky, 2005)	

* Although concept-based adaptation is generally considered of high computational complexity, in AHA! each document corresponds to one concept, which makes computational complexity comparable to social adaptation.

2.4 ARCHITECTURE OF THE AHS

There are several views on the architecture of the AHS proposed in the literature (Bra & Calvi, 1998; Brusilovsky, 2004; Conlan et al., 2002; Paramythis & Weibelzahl, 2005). In this section we are going to present our own view on the architecture, which is an adaptation/variation on several of the mentioned predecessors. Although we are discussing the architecture of the AHS, many parts would be characteristic of virtually all user-adaptive systems. This architecture covers the majority of the AHS that differ from each other only by the kind of adaptation they provide.

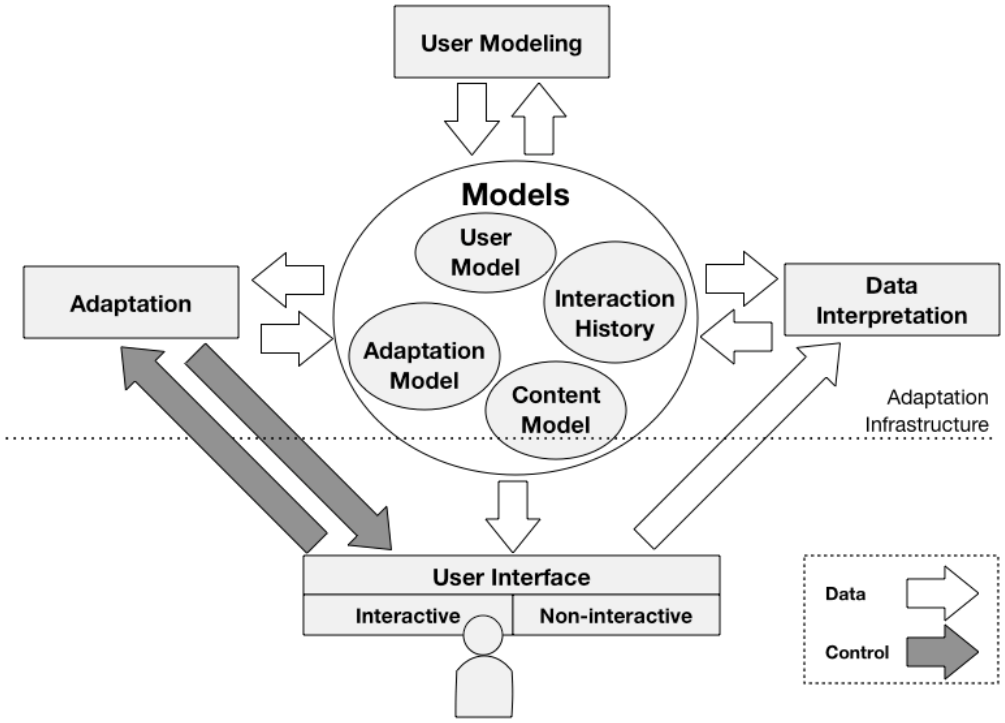


Figure 3. Architecture of the adaptive hypermedia system

Our version is process- and data-centric (see Figure 3). As with any conceptual visualization, it sacrifices some details for the sake of others. The architecture of the AHS is composed of the four core sets of models, four core processes, and a set of content repositories. In the next few sections we will discuss them in greater detail.

2.4.1 Models

In Figure 3 models are shown as ovals. The **interaction history** modelis a collection of a set of all user actions that were collected during users' work with the system. It includes basic actions such as traversing the hyperspace, scrolls, clicks, data submitted via Web forms, results returned by interactive pages, or even something as complex as saccades and fixations captured by eye-tracking equipment, etc. Interaction history is the source of all the information about the users, both directly stated and/or inferred.

The user model is a term defined differently, depending on the research area. Here we will define user model in a very general manner as information about the user, obtained by interpretation of, aggregation of, classification of, and inference based on raw data from the interaction history or information already stored in the user model. Often information stored in the user model is partitioned into static (never changed or changed infrequently) and dynamic (changed frequently), and comprises user demographic and socioeconomic information, preferences, beliefs, plans and goals, abilities, mastery of skills, possession of knowledge, emotional state, etc. In each of the mentioned categories there can exist alternative and/or competing models.

The content model is the most versatile model in the AHS architecture. It consists of the following types of data:

- Enumeration of the available pages/documents and static/dynamic link structure of the hyperspace,
- Model(s) of the subject domain(s) addressed in the documents, potentially structured in the form of taxonomy(ies) or ontology(ies), consisting of topics, concepts, and skills (often united under the term metadata items) of various granularity levels,

- Metadata index – the set of relations between pages/documents of the hyperspace and metadata items of the domain model(s); each relation can be assigned additional properties denoting its strength (with a larger value meaning greater importance of the metadata in describing the document) or type (the metadata item can be the main focus of the document, its prerequisite, etc.),
- The contextual information about the structure of the hyperspace is arguably a part of the content model as well; this information includes access policies and access control, conditional configuration of the hyperspace enabling the adaptation mechanisms for the (part of the) hyperspace.

The adaptation model's primary purpose is storing the adaptation strategies. These strategies prescribe what data the adaptation should be based on, how this data should be processed (retrieved, joined, aggregated, etc.) and how the result of the processing should be represented so that the user interface process could incorporate them in the content delivery process. In addition, the adaptation model could include a sort of working memory, or cache, to store run-time and intermediary results of the multi-stage adaptation procedures.

2.4.2 Processes and Data Streams

Processes in the AHS architecture (shown as squares in Figure 3) as well as the data streams between the processes and the models are discussed in this section.

The **user interface** process is the gateway between the user and the system. It is responsible for presenting the content of the hyperspace (extracted from the content model) and the augmentations introduced by the adaptation process. In addition, the user interface is the sensor that collects user interaction and communicates it to the data interpretation process. Naturally, the user interface is thought of as a purely interactive process; however, things such as

currently active access policies and the adaptive affordances could be hard-coded for a particular context and hence are not interactively changeable.

The data interpretation process is a data gateway to the AHS. This process handles raw bit stream, collected in a synchronous manner from the user interface, and enriches it with semantic meaning (Paramythis & Weibelzahl, 2005). The data to be interpreted includes requests for new pieces of content and traversals of the hyperspace, results of user interaction with content and/or adaptive augmentations, and manipulations with the interface (control). Once translated, the information is sent to the relevant model; for example hyperspace traversals are sent to the interaction history, interactions with adaptive augmentations can be sent to the adaptation model, and activations of interface features go to the content model.

User modeling is a process whose main purpose is producing assumptions about the users and storing them in the user model. The assumptions about users are produced on the basis of interaction history, content model, and the previous state of the user model itself.

For example, an overlay model of knowledge is a vector of values [0, 1] with respect to domain ontology from the content model. Initially all values are set to zero (no prior knowledge). As new user actions are added to the history model, the user modeling process looks up ontology concepts in the content model for the educational resource the user was interacting with. Depending on the result of the interaction, the knowledge of the corresponding concepts is updated.

Another example is a user model of user preferences in an adaptive recommender system. Upon arrival of a new rating of some item, say a movie, the user modeling process retrieves the item's features from the content model (e.g. actors, movie genre, director, etc). Depending on

whether the rating is high or low, the preferences of the user are updated. If the system uses collaborative recommendation, the distances to other user models are recalculated as well.

Adaptation is what makes AHS different from any other interactive system and what makes one AHS different from another (different AHS implement different types of adaptation techniques (Brusilovsky, 1996; Brusilovsky, 2001). Adaptation generally is a process of changing something in order to adjust it to new conditions or to make it suitable for a new use or purpose. In the case of AHS, the adaptation can be applied to the content and/or user interface providing access to that content. Historically, adaptation methods in AHS are divided into two major branches: adaptive navigation and adaptive presentation (Brusilovsky, 1996; Brusilovsky, 2001). Methods of adaptive navigation are concerned with helping the user to select an optimal path through the served content, while methods of adaptive presentation relate to altering the content itself to better suit the user.

Adaptive presentation works on the level of content items (pages, documents). Adaptive presentation methods include selecting appropriate format/modality of the content (text, audio, video), adjusting the structure of the text/multimedia (adding/removing fragments, [de]-emphasizing certain parts, [re-]ordering sections), generating natural language content on the fly, etc. Methods of adaptive navigation support are manipulating whole content items, usually by providing the user with a personalized set of links to them. The abundance of the adaptive navigation support techniques can be categorized according to the following dimensions:

- Number of options the user is given: only one link to a piece of content that the system thinks is best for the user, or a set of links the user can choose from.

- Presence and nature of changes to the [link] structure of the hyperspace. Less relevant links can be removed or new links can be generated, links can be adaptively filtered and sorted, or link structure can be intact but the presentation of links is changed (link annotation).

There are three parts to the adaptation process that are responsible for making a decision whether the adaptation should be applied in a given context (if adaptation is optional), selecting the adaptation method to be used (if several are available/applicable), and finally producing the augmentations to the user interface that will be visualized for the user. Let us discuss these three types of adaptation sub-processes in more detail.

The adaptation appropriation sub-process is responsible for making a decision about whether the adaptation should be applied to the user's view on the hyperspace at all. This decision is based on the current state of the user model and possibly the content model and the interaction history. For example, in an educational AHS, an adaptation appropriation sub-process can make a decision to invoke adaptation based on how many consecutive unsuccessful problem solutions have been put into the interaction history. If the threshold is exceeded, the flag would be set in the working memory of the adaptation model and the chain of sub-processes would continue; otherwise, no further action is taken.

The adaptation selection sub-process deals with choosing the right adaptation method, based on the information in the user model, context model, and interaction history. For example, if adaptation appropriation raised the flag after a series of problem-solving errors, then the adaptation selection sub-process would use the context model to retrieve metadata of the failed problems and query the user model for level of acquired knowledge of those metadata concepts. Judging from this data, the selection sub-process would then decide whether remedial exercises should be recommended (exercise recommender adaptation should be chosen) or the user should

be directed to relevant reading (reading recommender should be chosen). The choice is then stored/scheduled in the cache of the adaptation model.

A set of adaptation sub-processes is there to carry out the actual adaptation techniques. When a certain adaptation technique is scheduled, the appropriate adaptation strategy is retrieved from the adaptation model. This strategy is then applied to the current context of the user model, content model, and interaction history. For example, if the adaptation selection sub-process has chosen the exercise recommender adaptation method, the strategy would be to retrieve the knowledge level of the concepts that belong to the metadata of the recently failed problems. The recommended problems are selected from those that have not been solved or were solved no less than a certain time ago; those thataddress the concepts of the recently failed problems; and problems for which the user's knowledge level of these concepts is lower than a certain threshold. The identities of the recommended problems are temporarily added to the content model of the user. Their time to live can be fixed or be a subject of an additional cycle of adaptation loop.

It is worth mentioning that adaptation is the only required sub-process. Appropriation and selection sub-processes are optional. Enabling of the adaptation per se, as well as the choice of a specific adaptation technique, can be hardcoded in one of the policies of the content model.

2.4.3 The Orchestration

As we can see in Figure 3, there are two types of arrows. Pale ones denote data transmission and dark ones indicate the transfer of control. Since there is transfer of control between only two processes – user interface and adaptation – most of the computations in the AHS are done in the

asynchronous manner[3]. New information about the user/environment results in new data in the interaction history, which, in its turn, can trigger the user modeling process to start the user model update[4]. Accumulation of a sufficient amount of changes to the user model could launch the adaptation appropriation mechanism that forwards control to adaptation selection and adaptation itself that augments the content model as necessary. The user is presented with the result of this background processing after another loop of interaction with the AHS when the screen is updated, upon availability of new adaptive information (e.g., implemented via polling), or via a forced update from the adaptation process.

In Figure 3, the user interface is separated from most of the rest of the AHS architecture – adaptation infrastructure. The only part that "stays" with the interface is a fraction of the content model that describes the hyperspace structure, the content itself, and various policies regulating access and configuration of the hyperspace traversal. The only two channels for the user interface to communicate to the rest of the architecture – adaptation infrastructure – are through uploading new user interaction data to the data interpretation process, and through receiving information about availability of adaptive augmentations from the adaptation process.

[3] There are adaptive systems that have more synchronization than we shown in Figure 3, for example adaptive systems using eye-tracking data sensors, or adaptive cognitive tutors, but most hypermedia systems are predominantly asynchronous.

[4] The user modeling process does not necessarily start a new round of user model update upon receiving every new piece of evidence, the so-called data *push* approach (Yudelson et al., 2007). There are user modeling implementations that start producing new assumptions about the user only when requested (data *pull* approach). Both methods have their scopes of applicability and could be combined into a hybrid pull-push solution.

2.5 DECOMPOSITION OF AHS AND COMPONENT REUSE

The architecture of AHS, presented in Section 2.4, discusses principal elements of an AHS. Early AHS were implemented as all in one. Namely, the system was monolithic and the developers had to develop all AHS components from the start. However, not all of the modern AHS are realized like that. In many cases developers separate one or several parts of the architecture into a standalone system, thus splitting or decomposing the AHS. The general motivation behind decomposing monolithic AHS is the ability to reuse successful implementations of one or several AHS components. In particular, this is true for those parts of the AHS architecture that require high levels of expertise and a lot of effort to build. Standalone architectural components offer convenient abstraction and free the developers of other AHS elements from the necessity to build/replicate the components again.

The borders, along which the monolithic architecture of AHS is split, although seemingly artificial for those unfamiliar with AHS, are often rigidly defined by de facto specialization of expertise and labor when developing modularized AHS. This specialization is even more visible in component-based AHS, where system modules could be reused and/or pooled across several research and development teams. The boundaries of every potentially decomposable AHS component are thus defined by the field-specific notion of a finalized product one developer or a team could potentially outsource to be used by other teams of developers.

This section discusses the evolution of the AHS architecture with respect to efforts and the rationale behind attempts to divide a monolithic AHS into separate standalone parts. First, we will talk about separation of user modeling. Next we will focus on the so-called open corpus AHS that marked (partial) separation of the content model. Then we will move to ways of

separating the adaptation model and process. We will finish with the implications that the distributed nature of the AHS brings.

2.5.1 User Modeling and Its Separation From the AHS

The user model and user modeling process is a cornerstone of any adaptive system. User models were first mentioned in the literature in the early 1970s (Carbonell, 1970). By the late 1980s the field of user modeling had matured enough for a new class of general user modeling systems (or shells) to emerge. User modeling shells (UMS) were standalone systems providing user-modeling services to client applications via a set of communication protocols. Thus, even before the first adaptive hypermedia systems were built, UMS had already become an independent component.

2.5.1.1 Structure

UMS, when split from the rest of the monolithic AHS, span several models and processes. All UMS include a user modeling process and the user model. Many of the user models also include the process of data interpretation as well as part of the interaction history that is utilized for producing assumptions about the user. In many UMS, especially those in the educational domain and recommender systems area, metadata of the hyperspace items that users interact with is queried very frequently.

2.5.1.2 Rationale and Benefits

The main reason for devising a decoupled UMS was that, out of all other components of the user-adaptive system, building the user model requires the highest amount of cognitive and man

machine interaction expertise. Namely, one must be knowledgeable about how to represent the user's preferences, goals, beliefs, knowledge, skills, etc.[5]

Separation of UMS from the user-adapted system makes it possible to abstract from the problem of collecting and processing user-related data and building user models. When building a new AHS, one can avoid building a UMS component from scratch and choose an appropriate UMS that already exists. The problem of building the UMS is thus reduced to the problem of complying with data-exchange protocols of an existing one.

2.5.1.3 Examples

The first ever UMS (then called a user modeling shell) built was GUMS (Finin, 1989; Finin & Drager, 1986). GUMS allowed creation of simple stereotypic hierarchies, each stereotype described by a set of Prolog rules. At runtime GUMS accepted new facts about users, verified the consistency of the currently held assumptions, and answered queries regarding system's assumption of the users.

Another noteworthy example from the era of UMS separation is BGP-MS (Kobsa, 1990; Kobsa & Pohl, 1995). BGP-MS is a general purpose UMS capable of delivering user modeling to any application that complies with a simple message exchange protocol. On the basis of the input from the served application, BGP-MS infers the user's beliefs and goals.

GUMS and BGP-MS have shaped the thinking of the user-adaptive systems community to such an extent that a dedicated stream of user modeling systems research was formed. As part

[5] The rationale and benefits of separating UMS into a standalone system are only applicable to adaptive hypermedia systems. In contrast to the AHS, where the model of domain is viewed as partially independent from the modeling itself or even replaceable, there is another class of systems – intelligent tutors –where the domain model is the central entity, incorporated into the modeling itself.

of that stream, a number of successful UMS have been created, including *um*(Kay, 1995), *TAGUS*(Paiva & Self, 1995), Doppelgänger (Orwant, 1995), *CUMULATE*(Brusilovsky et al., 2005), LDAP-based UMS by Kobsa and Fink (Kobsa & Fink, 2006), and many others.

2.5.2 Separation of the Content Model: Open-Corpus AHS

Less then 10 years after UMS separation, researchers in the adaptive hypermedia field developed and successfully tested a rich corpus of adaptation techniques (Brusilovsky, 1996). With the maturity of AH in general, by the late 1990s came the understanding that one of the major shortcomings of AH was that adaptation is most often implemented on a limited corpus of resources from a single domain. Such systems were named closed-corpus hypermedia systems. Often adding a resource to the closed corpus hypermedia system (also called localizing the resource) is a high-cost operation.

At this time various researchers started focusing on the problem of the openness of the AHS. In late 1990s – early 2000s a new class of hypermedia systems has been suggested – open corpus hypermedia (Henze & Nejdl, 1999; Henze & Nejdl, 2001), where the set of the served resources is not known at the systems' design time. The main improvement in these systems as compared to closed corpus is the lower threshold for adding resources, which is done by using various techniques for reducing or amortizing the resource localization costs.

2.5.2.1 Structure

An open corpus AHS is a system that does not mandate the presence of a large portion of the content model or has mechanisms of extending the content model. The extendibility includes:

- The possibility of adding new documents as well as inter-document links,
- Connecting already existing hyperspace items with the new ones,

- Appending new and/or alternative domain models,
- Adding metadata indexes, connecting domain model items to the documents (both native to the system and added under the open corpus paradigm).

By definition, open corpus pattern does not specify the amount of work necessary to localize new items in the hyperspace and/or presence and extent of tools automating this process. However, the general understanding is that localization should be a reasonably easy process, doable by a specialist with a fair amount of knowledge of the target domain and basics of the adaptive systems.

2.5.2.2 Rationale and Benefits

One of the major criticisms that AHS receive is a limited amount of content that adaptation can be provided for. The introduction of the open-corpus paradigm has broadened the perspectives for the AH, making significantly larger volumes of content potentially adaptable. Open corpus hypermedia opens the road to richer systems allowing adaptive hypermedia researchers to move the focus to adaptation and allowing them to at least partially abstract from content authoring that now could be acquired elsewhere.

The introduction of open corpus systems stimulated out-of-the-box thinking in the field of adaptive hypermedia. It became evident that it is easier to find/train a content author, capable of creating quality content, and teach him/her how to make this content localizable in an AHS, than for AHS designers to play the role of content authors themselves. In the case of already existing content, it is in most cases easier to localize good resources than to create them anew specifically for the AHS.

The major obstacle in creating the localizable “open” content is a knowledge engineering task frequently used in the process of localization. Here a content author has to be able to

properly connect the document to the rest of the hyperspace and the domain model metadata. Automation of these tasks is still a research problem, and existing prototypes leave much to be desired.

2.5.2.3 Examples

From the conception of the idea until now, open-corpus AHS have been predominantly created in the educational domain, hence giving the name to the AEHS (adaptive educational hypermedia systems). First, the idea of openness and resource sharing is more attractive for academia rather than industry. Second, at the time when open-corpus AHS emerged, a large community of instructors were already publishing educational content in the hypermedia format and provided naturally trained content authors for the AEHS. In addition to educational AHS, a number of collaborative resource discovery open-corpus AHS were built, for example, CoWeb (Dieberger & Guzdial, 2003) and FOHM (Millard et al., 2000).

Soon after the first open corpus systems were deployed, researchers realized that this was a great opportunity for content reuse across existing hypermedia systems. There were several two-system content sharing/exchange attempts made: AHA! And InterBook (Ramp et al., 2005), MOT and WHURLE (Cristea et al., 2005), and MOT and AHA! (Cristea et al., 2005).

Initial attempts to build open corpus AHS mostly dealt with by-hand localization in a pre-constructed hyperspace (Henze & Nejdl, 1999; Hirashima et al., 1997), when the linking of the new documents had to be done manually. Later works attempted to (partially) automate hyperspace construction via the use of keyword extraction (Smith & Blandford, 2003). The negative side of these approaches was inaccuracy of keyword extraction that resulted in document misplacement in the hyperspace. Adding a semantic metadata layer to document indexing in the form of conceptual structures such as thesauri or ontologies (Carr et al., 2001;

Dolog et al., 2003) complicated the process of indexing but resulted in creating more representative structures of the document spaces.

2.5.3 Adaptation in the AHS: the Emerging Split

Since the onset of the AHS in the early 1990s, the classical way of implementing adaptation was to fuse the adaptation process and the adaptation model with the rest of the system that was monolithic. In the beginning, the adaptation could hardly be separated from the user interface, and adaptation strategies were hardwired and predominantly part of the content or domain model. However, by the mid-1990s, as the AHS field took shape, the adaptation was conceptualized into a distinct part of the AHS architecture.

2.5.3.1 Structure

Out of all other components of the AHS, the adaptation (adaptation process and adaptation model) took the longest time to become differentiated in the architecture and potentially reusable across several AHS. Nevertheless, unlike the user modeling and the content, it is the most naturally distinguishable. Components of the adaptation model are the storage of one or more adaptation strategies and the three adaptation sub-processes (appropriation, selection, and adaptation itself) that are responsible for enabling, selecting, and applying them at a given moment in a given context.

2.5.3.2 Rationale and Benefits

The reasoning behind separating adaptation and adaptation model is that one should not lock a successful adaptation within a particular system, when it can be reused. Reusable adaptation strategies provide for the ubiquitous nature of adaptation that can be potentially reused by

multiple systems. First, availability of reusable adaptation strategies could offer a greater selection of the adaptation strategies themselves: The AHS or even the user can change to the adaptation that is most appropriate or preferable in a given situation. Second, availability of reusable adaptation strategies provides fault tolerance. If several alternative implementations of the same adaptation strategy are available (e.g. from different providers), they can be substituted should one fail and stop functioning for some reason. Finally, presence of reusable adaptation reduces the question of development to the question of connecting to the "hooks" of an existing adaptation provider, and this is easier than developing the functionality from scratch.

2.5.3.3 Examples

There are several ways of implementing reusable and/or standalone adaptation components in the AHS, featuring different degrees of integration with the rest of the AHS infrastructure.

The traditional approach is tight integration of the adaptation into the user interface part of the AHS infrastructure (Figure 4). In such an introvert solution, the adaptation is realized as part of the user interface, for example InterBook (Eklund & Brusilovsky, 1999), KBS Hyperbook (Henze & Nejdl, 1999), QuizGuide (Brusilovsky et al., 2004), NavEx (Yudelson & Brusilovsky, 2005). It thus has direct access to native and/or localized foreign resources. The whole content model or part of it are usually integrated along with the adaptation.

The advantage of such a solution is that minimal or no effort is necessary to provide adaptation to an already localized resource. A strong disadvantage is the high cost of localizing foreign content. One more drawback is that the implemented adaptation techniques are limited to working with one AHS only.

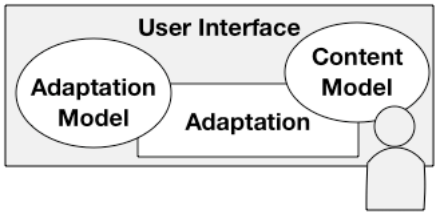


Figure 4. Integration of adaptivity directly into user interface component

The other (non-traditional) way to implement adaptation is to package it as an intermediary adaptation provider (Figure 5). First introduced as a way to manipulate Web content in general by Barrett and Maglio (Barrett & Maglio, 1998), intermediaries are an additional access point to the content or an alternative user interface. In addition they often carry part of the content model with them. Our own experience with intermediaries providing adaptive navigation support (Brusilovsky & Sosnovsky, 2005; Yudelson & Brusilovsky, 2005) has shown that it is quite successful with users. Availability of adaptive navigation support implemented as value-added services has doubled and sometimes tripled user persistence of working with available resources (as compared to users who did not have access to adaptive navigation shells) (Brusilovsky et al., 2009; Brusilovsky et al., 2006).

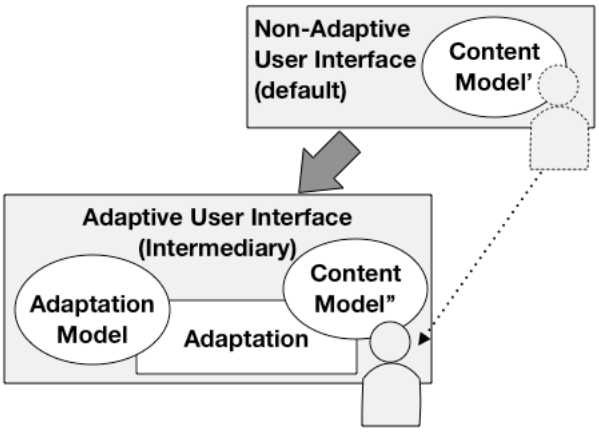


Figure 5. Implementing adaptation via an intermediary

If adaptation is realized as an intermediary added value shell, the user interface can be very simple and lightweight. Usually, an intermediary adaptation provider targets a pool of

similar resources. For example, QuizGuide provided navigation support to parameterized quizzes (Brusilovsky & Sosnovsky, 2005) and NavEx covered annotated examples (Yudelson & Brusilovsky, 2005). Adding more similar resources to the pool is quite easy. The disadvantage of this approach is that for every new pool of resources, either the existing intermediary has to be substantially changed, or a completely new intermediary has to be devised. From our experience we can attest that the amount of effort necessary for configuring an intermediary to work with a new class of content is quite large (Brusilovsky et al., 2005). Another negative side of the intermediary adaptation provider approach is that an intermediary creates a disruption with the original AHS user interface – users have to go to yet another access point and familiarize themselves with it.

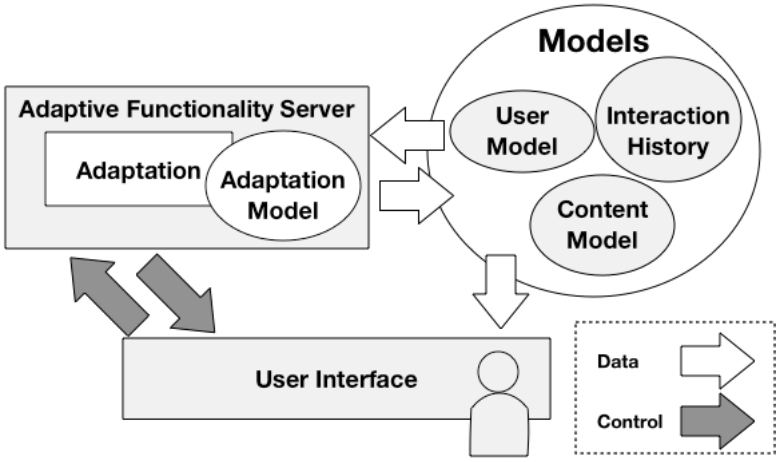


Figure 6. Adaptation implemented as a standalone server

The third, currently emerging, way of implementing adaptation is to outsource the adaptive functionality to an autonomous system (Figure 6). It is this approach that we are focusing on in this dissertation. Such a standalone or 3rd-party (Koidl et al., 2009) adaptive functionality server (AFS) abstracts adaptation provision in the same way UMS does: The adaptive support is provided upon request according to a predefined (set of) protocol(s). The idea of separating adaptive functionality into a widely reusable set of services has existed for some

time. One of the first examples of AFS was the APELS system (Conlan & Wade, 2004). In APELS, however, various adaptive functionalities could be reused only by a limited number of federated adaptation consumers.

Authors of the Personal-Reader framework (Dolog et al., 2004) came closer to creating truly reusable adaptive functionality. Personal-Reader is a sophisticated, well-developed architecture for adaptive access to educational resources in which the adaptive functionality server and the user modeling server are abstracted as standalone systems. However, Personal-Reader is still more of an introvert framework with a threshold to pass in order to start using its benefits.

Our own work on the architectural issues in AHS is codenamed ADAPT2 (read adapt square) – advanced distributed architecture for personalized teaching and training. First mention of ADAPT2 in the literature can be found in (Brusilovsky et al., 2008), however, the conceptual basis was laid in (Brusilovsky, 2004) when it was called Knowledge Tree. ADAPT2 features all major components of the modern AHS framework including an experimental version of the adaptation service provider (Yudelson & Brusilovsky, 2008). One of the major features of ADAPT2 in general and its adaptation server in particular is its extrovert nature. The adaptation server is not built to work with ADAPT2 or its components exclusively. It is designed to be abstract enough to fit arbitrarily into any AHS system that needs ubiquitous adaptation support. The shortcoming of ADAPT2's adaptation server is that so far it offers a limited set of adaptation services and not all of them were evaluated in terms of performance. Overcoming these shortcomings is one of the foci of this work.

2.5.4 Summary

One of the key critiques the AHS field as a whole has received is, despite a significant volume of research, a limited representation in the hypermedia world today. This is one of the reasons why the focus of a solid part of the AHS community is shifting from developing and testing new adaptation approaches to making the already developed approaches applicable in a larger number of contexts, equipping a larger number of hyperspaces with adaptation support, and, as a result, reaching larger audiences.

A promising way to do that is involving more people in the development of the AHS. This can be done through delineation of the "zones of responsibility" or "stakes" in the AHS architecture and charging classes of professionals with the responsibility over these "stakes." While many AHS even today are designed and developed by small teams of one or two people, there are multiple roles that developers play. These roles require different levels of domain-specific and adaptation-related proficiencies. Figure 7 provides a rough summary of the developer roles, levels of expertise, and amount of professionals available.

In Figure 7 the domain-specific expertise covers the subject of the principal content AHS serves to its users. Developers of the AHS infrastructure require low to moderate levels of domain knowledge. User model developers do not require deep knowledge of the domain at all. Developers of adaptive functionality have to have a better understanding of the subject to provide better working adaptation. In contrast to AHS infrastructure designers, content developers have to be proficient in the subject of the principal content, since their primary goal is creation of the principal content.

The distribution of adaptation-related expertise is the opposite. User model designers are to be proficient in the mechanisms underlying the adaptation. Proficiency requirements for

adaptive functionality developers are less strict, since they are abstracted from the user modeling. Content developers do have to be aware of the basics of adaptation, but the level of proficiency does not have to be as high as for AHS developers.

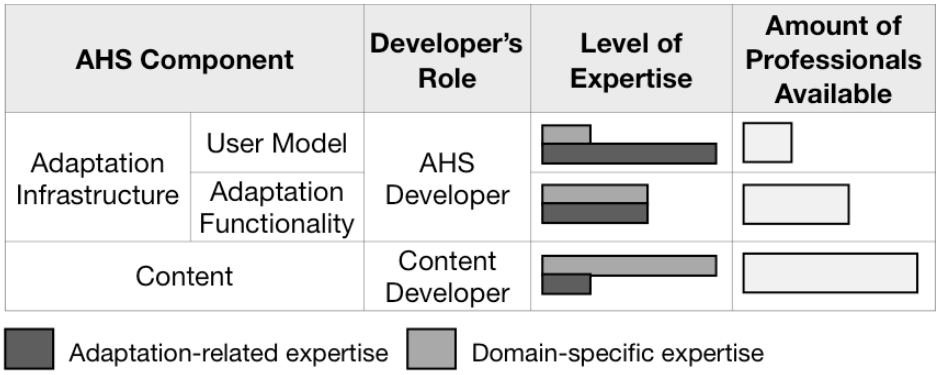


Figure 7. Summary of developer roles, expertise levels, and number of relevant AHS professionals available

As we can see from Figure 7 the number of trained professionals for each of the AHS core components is inverse to the adaptation-related expertise necessary. The more adaptation-related expertise that is required for an AHS component, the fewer professionals there are who can design this component well. The most adaptation-related expertise intensive component – the user model – has been thought of as an independent entity for nearly two decades now. As a result, both the machine interface between the user model and the rest of the system and the developers' zone of responsibility have been clearly outlined.

Content became separable a little less than a decade ago with the onset of open-corpus hypermedia. Just as in the case of the user model, for the open corpus, the responsibilities of the developers as well as the architectural delineation are agreed upon. It is the adaptation functionality that is "everyone's and no one's land" at the same time. Either user model designers have to be knowledgeable of the domain and devote additional effort to developing adaptive support, or those who are responsible for the user interface are charged with implementing adaptation techniques. Sometimes adaptation accompanies the content itself. In

such cases “smart content” (Farzan & Brusilovsky, 2005; Farzan & Brusilovsky, 2008; Lundgren-Cayrol et al., 2001) takes part of the responsibilities of the user interface with embedded elements of the content model. While there are clear advantages to either of the mentioned architectural “mergers,” in this work we advocate the solution in which adaptation functionality is implemented as a standalone service provider, an engine offering various kinds of adaptation support.

Despite the benefits of implementing a standalone adaptation server (discussed above in section 2.5.3.2), such server has to be able to offer a necessary level of abstraction and provide a general adaptation service that can be adopted across multiple contexts and systems. This would call for a thorough design of the adaptation process and the logistics of the communication. Additionally, performance issues have to be addressed. Since at this point all of the AHS components could be residing on separate systems integrated only via network protocols – the need for synchronization, the network and processing overhead could play an important part in the overall performance. Poor design of the adaptation functionality engine – the central point of the AHS – will immediately fail all its other parts as well.

Taking into account the discussed features of the modern AHS, proper evaluation has to be performed to make sure that the implemented component-based system provides effective adaptive support and performs well when used by many users. The following section devotes special attention to the questions of AHS assessment.

2.6 METHODS OF ASSESSING ADAPTIVE SYSTEMS

Adaptive hypermedia systems (AHS) conceptually are a subset of adaptive systems (AS) in general. Thus, virtually all evaluation methods suggested for AS are applicable for AHS, because from the evaluator's point of view AS and AHS are the same. The main focus while evaluating AHS/AS is to assess the impact of adaptation, since adaptation is the feature that makes AHS/AS different from the rest of the systems.

Evaluation is generally considered to be an essential part of research and practical application of the adaptive technologies. It is important to know whether a particular adaptive technique works, the context where it is most potent, and the scale of the adaptation effect it produces. Lack of evaluation data and/or inability to generalize from it is one of the major barriers that stand in the way of adaptive methods becoming mainstream technology (Höök, 1998).

Over the last 20 years, in the field of adaptive hypermedia systems a large number of technologies have been proposed and successfully tested (Brusilovsky, 2001). Given the abundance of existing techniques and their implementations, the evaluation of the already suggested approaches is as important as coming up with novel ones (Brusilovsky et al., 2004).

This section will discuss various approaches to assessing AS/AHS. We will start with usability evaluation. Then we will move to methods for assessing the value of adaptation. After that we will discuss performance evaluation. A short summary will conclude this section.

2.6.1 Evaluating Usability of the AS/AHS

Usability evaluation methods were not created for assessing adaptive (hypermedia) systems alone. Usability, as opposed to the adaptation, is not the focus of AS/AHS systems research; the

adaptation can not only improve the system's effectiveness but also degrade it. Thus usability-related evaluation is useful to prevent collecting conflicting evidence about AS/AHS.

The heuristic evaluation method is guided by a set of rules of thumb, general principles, that argue for or against certain design decisions (Nielsen & Molich, 1990). In heuristic evaluation, a small set of specialists inspect the system and look for problems and violations of the general principles. In the field of adaptive Web in general and adaptive hypermedia in particular, a set of universally recognized heuristics is still missing. This deficiency can only be remedied by conducting experiments and publishing results demonstrating, for example, the context of most successful applicability of a certain technique, or supremacy of one adaptation technique over the other in certain situations, etc. The most valuable feature of such experiments is the ability to replicate and reuse them in similar contexts.

A good example could be (Sears & Shneiderman, 1994). This work discusses experiments with menus, where items are sorted according to frequency of their use. The authors have found that usage frequency sorting disorients users and instead the so-called split menus should be used to avoid confusion.

Expert review is the evaluation method specifically useful during early stages of a system's life cycle. For example, experts can help in picking user modeling dimensions and adaptation features. In (Magnini & Strapparava, 2001) the authors used expert opinion to help populate the initial recommendation data. Experts can also contribute towards designing an uncertainty-based user model (Brusilovsky & Millán, 2007). In terms of evaluation per se, experts are a good resource for assessing interface adaptations (Gena & Torre, 2004) as well as the inference mechanism underlying the adaptation (Ardissono et al., 2003).

Cognitive walkthrough is a method of evaluation where experts take on the role of actual users in attempt to find potential problems (Polson et al., 1992). Traditionally, this method only focuses on general HCI aspects and does not delve deeper into the design of adaptation.

Unfortunately, these predominantly analytical methods do not involve the user directly. Instead they assume that the evaluator and/or the expert assess the system from the point of characteristics of a "typical" user. Since the concept of "typical" is contrary to the very definition of the adaptive system, it has limited applicability in evaluation of AS/AHS (Paramythis et al., 2001).

2.6.2 Evaluating the Value of Adaptation: Holistic Approach

Evaluation of the value of adaptation in the area of AH/AHS is tightly connected with the so-called **controlled experiments**. This group of methods is one of the most effective, most frequently used, and most often discussed techniques in the area of AS/AHS (Chin, 2001). The idea of a controlled experiment is changing one element of the environment (the independent variable) and controlling the effect on the user measured behavior (the dependent variable). The aim of the experiment is to support hypotheses about causal relationships between experimental variables. Controlled experiments can be used to assess things like the accuracy of modeling, quality of produced adaptations and recommendations, etc (Jameson, 2003). Controlled experiments can be short-term or long-term. Short-term experiments are usually associated with a greater level of control over the factors that can potentially influence the measured outcomes. Long-term experiments offer less control over the factors, but in return offer a more established pattern of the collected data due to an extended duration. The following three methods belong to the class of controlled experiments.

With and without is a popular technique of assessing the value of adaptation. Here an adaptive version of the system is compared to the one lacking such support (see e.g. (Boyle & Encarnacion, 1994; Brusilovsky & Eklund, 1998; Brusilovsky & Pesin, 1998; Kaplan et al., 1993; Meyer, 1994; Weber & Specht, 1997). A frequent criticism of this method of evaluation is that a non-adaptive version of the system cannot be good for comparison, since often adaptation is "built into" the system. Without it the systems can become absurd and even useless. Besides, the "with and without" method often does not tell why the adaptive version is successful (if it is) (Höök, 1998; Höök, 2000). In some systems, especially those based on machine learning algorithms, the adaptation simply cannot be switched off (Krogsäter et al., 1994; Pohl, 1997). Attributing all the success to the presence of the adaptation is not always possible, either. For example, (Brusilovsky et al., 2009) describes two companion adaptation shells that combine the effect of adaptation with collocation of the content. Up until now the interaction of these two factors was not possible to pinpoint. Sometimes with and without studies can give contradictory results, for example see (Brusilovsky & Eklund, 1998).

Single conditioning is a method where adaptation is rendered as an integral part of the system and no comparison is made to the non-adaptive version. In single conditioning all variables are dependent. This approach was pioneered in (Oppermann, 1994) and (Krogsäter et al., 1994) for evaluating of the system Flexel II. **Correlation study** is an example of single conditioning. A good example of correlation study is (Iglezakis, 2005), where the authors perform an empirical evaluation of an adaptive help system. The ACT-values of procedural knowledge were correlated with measures of user performance. Another example would be correlating task success (in terms of time, performance, or number of objectives met) and usage of the adaptively generated system suggestions (Brusilovsky et al., 2009). Single conditioning

and correlation studies suffer from the same problem as the with and without method, namely, problems attributing observed user behavior to the overall "success" or "failure" of the system.

Controlled experiments are not limited to **objective measures** of user performance, but also include **subjective measures**. These include user satisfaction with the system and perceived utility of the system as a whole or a subset of its features. For example, the authors of WebEx – an annotated code example server (Brusilovsky, 2001) – were not able to find an impact of system usage on users' measured performance in class. However, perceived usefulness has been scored high in the exit surveys.

All of the evaluation methods discussed in this section so far are holistic. They look at the AS/AHS as a monolithic entity. No special attention is given to the internal composition of the system. It has been shown, however, that there is a great benefit in going beyond such a "black box" approach. The next section is devoted to such approaches.

2.6.3 Layered Evaluation of the Value of Adaptation

The layered evaluation method is based on the idea that the evaluation does not treat the AS/AHS as a monolithic process. Instead, the process is broken into constituents. Each of these constituents, often called layers (Brusilovsky et al., 2001; Karagiannidis & Sampson, 2000) (hence the name of the method), has to be evaluated separately. A strong advantage of layered evaluation is that it successfully works with monolithic systems as well as systems that are composed from modules or even distributed systems. The first mention of breaking down the process of adaptation can be traced to (Totterdell & Boyle, 1990), where the authors proposed assessing the user modeling process separately from the process of adaptation (incorporating knowledge about the user into the adaptive interface).

Layered evaluation is especially effective when assessing component-based systems like modern AS/AHS. However, layers are not only the modules that provide some sort of service (user modeling, adaptation, etc.), but also the components that do not actively participate in the process of adaptation (e.g. content model, adaptation strategies as part of adaptation model).

Instead of the "black box" approach, in which the system is appraised as a whole, each component and/or process is appraised individually, potentially even using a different method. An example of decomposing the process of adaptation is shown in Figure 8. Here the authors break the process of adaptation into data collection, data interpretation, user modeling, and making adaptation decisions. Brusilovsky and colleagues (Brusilovsky et al., 2004; Brusilovsky et al., 2001), who are among the pioneers of layered evaluation, argue that assessment of the AHS covers a multitude of questions, including general user interface usability, accuracy of user model, effectiveness of the chosen adaptation technique, etc. Reducing system evaluation to a binary outcome: yes (successful) or no (unsuccessful) is rendered as incorrect.

Due to synergy of the underlying factors, it could be the case that a great adaptation is buried by an ill-constructed user interface, or conversely, a well-done user interface might smooth the flaws of the adaptation and/or user modeling components. To take this phenomenon into account, the evaluation task should target specific layers of the system.

While breaking the process of adaptation into stages and evaluating them separately allows one to answer questions impossible to approach in a monolithic setup, nevertheless there are a number of assessment questions that require some layers to be combined or the system to be considered as a whole. For example, evaluating the user modeling combined with the process of adaptation appropriation can address the question of user model accuracy expressed in terms

of decisions about whether to apply adaptation or not in a particular situation or context (Paramythis et al., 2001).

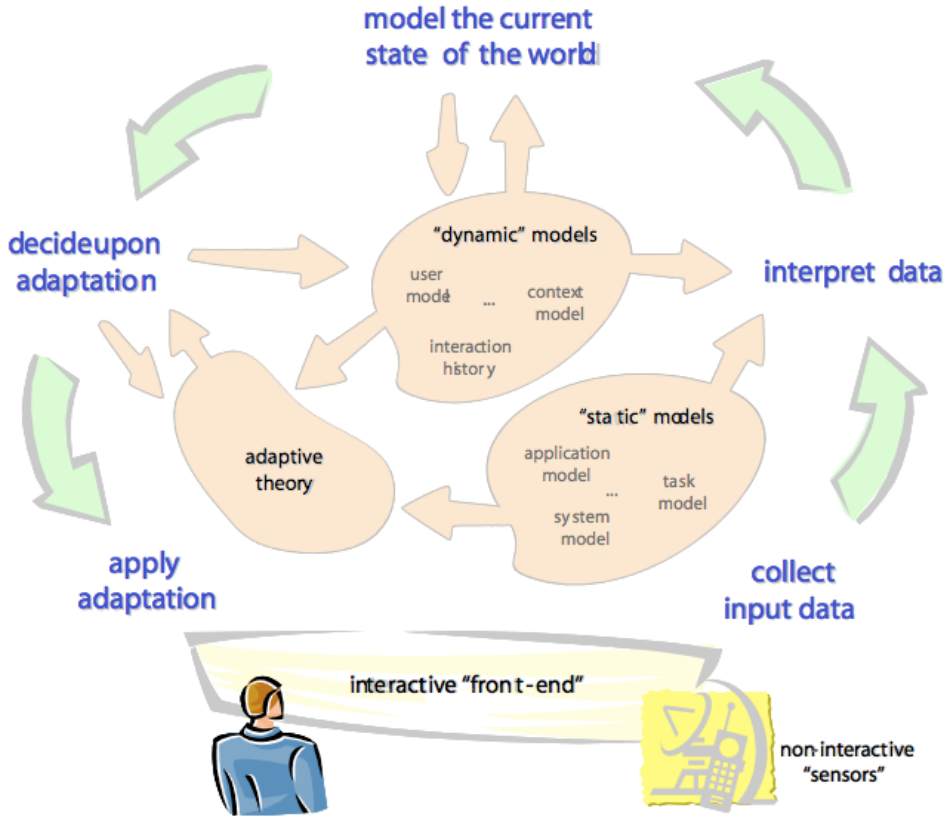


Figure 8. Adaptation decomposition (Paramythis & Weibelzahl, 2005)

Since its formal introduction in 2001, layered evaluation has become an actively researched topic in the AS/AHS field. This has been seen in a number of publications that focused on the methodological and organizational issues of the layered evaluation (Masthoff, 2003; Paramythis et al., 2001; Paramythis & Weibelzahl, 2005; Weibelzahl, 2005; Weibelzahl & Lauer, 2001; Weibelzahl & Weber, 2003).

In our own work, the layered evaluation of the AHS systems plays one of the central roles. In (Yudelson & Brusilovsky, 2005) we discussed results of studying the effect of the adaptive value-added service NavEx – a shell system for accessing code examples. The adaptive link annotation was evaluated separately from content presentation and user modeling. The

presence of the adaptive annotations was shown to increase user session length in terms of the time spent and the number of clicks, as well as to increase the number of sessions themselves. A comparative study of NavEx and QuizGuide (a value-added shell for accessing problems) showed that the found phenomena persist across several user groups, types of educational content, and types of user modeling (Brusilovsky et al., 2009; Brusilovsky et al., 2006).

In (Yudelson et al., 2007; Zadorozhny et al., 2008) we devoted special attention to the user modeling layer of the AS/AHS. The user modeling process was further broken down and the object of study was the type of evidence propagation: push based (in which the user model is updated as soon as new user data arrives), and pull based (in which the user model is updated on demand, when the user data is requested). The study showed a decisive superiority of the push based method in the context of modern fast-paced AEHS.

Despite greater flexibility of assessment and ability to measure things previously impossible to measure, layered evaluation has not become widespread. It is a primary focus of a small group of like-minded researchers who organized a series of workshops on empirical evaluation of adaptive systems. The bulk of the publications on layered evaluation come from co-organizers of these workshops.

2.6.4 Performance Evaluation

When the AHS field matured enough, it became evident that making sure whether adaptation works conceptually is not enough. Even AHS intended for small user populations (e.g. several dozen users) can be complex enough for performance to be a probable issue. In the light of the wider AHS dissemination problem that we are focusing on in this work, performance becomes one of the key questions.

2.6.4.1 State of the Art in Performance Evaluation

Although classical methods of evaluating AH/AHS (including the layered approaches), do not limit assessment to appraising the value of adaptation alone, it usually receives the bulk of the attention. It is important, however, to assess the architecture as well. One of the architectural features especially important in a component-based architecture is performance. The questions of acceptable performance and/or scalability of AS/AHS components are often mentioned in passing by the adaptive hypermedia community in general and the AS/AHS layered evaluation community in particular. However, as more and more modern AS/AHS become distributed, one of the major questions in their evaluation is whether such a system would be able to offer user experience (read performance) comparable to the standalone AS/AHS. The distributed nature of the system raises an immediate question of additional communication overhead. The communication itself has to be well thought through to exclude, for example, thrashing.

There exist a number of types of performance tests. Each of them addresses a different question regarding the system's operation. A subset of the ones relevant to our discussion is listed below (Maccaux, 2008).

- Benchmark test. The main characteristic of the test is the consistently reproducible results. The goal is to establish a set of performance values – a benchmark – to serve as a baseline for future performance evaluations.
- The capacity-planning test aims to determine how far a given system can scale under certain conditions. Reproducibility is not important in this case. Often random factors are included as part of capacity planning tests. A question this test attempts to answer could be: "How many concurrent users/connections/transactions can the system effectively support?"

- The soak test is a long duration test, usually with a control condition. The purpose of this test is to check the overall robustness of the system and to find performance degradation, memory leaks, etc., over a long period of time.

The problem of assessing performance of the AS/AHS has not received much attention in the literature. Below is an overview of the few publications that have addressed this issue.

In (Carmichael et al., 2005) the authors evaluate a location-aware user modeling component behind the MyPlace system. MyPlaceis a system that supports ubiquitous social communication in a location-aware context. The user modeling component in MyPlaceis extended beyond modeling just users. It also represents devices, sensors, rooms, and buildings. MyPlaceis set up in an office building. User locations are traced via personal mobile Bluetooth-enabled devices registered on the local Bluetooth network or via login/logoff events on the static workstations. When moving within/in/out the building the user location is updated with the system by the means of a special "tell" operator. The information about other users of the system is available via an "ask" operator.

The authors performed a soak-test of the user modeling component of the MyPlacesystem by running it for approximately 4 months and collecting the log data. During this period, roughly 2,500 user accounts were active and about 500 devices (mobile/terminal) were involved. The main questions behind the analysis of the user model performance were the scalability of "tell" and "ask" operators. The scalability was measured in terms of CPU time taken by the user model process versus the amount of data the user model has to deal with. For the "tell" operator, no load classes were defined. Two types of "ask" operator (simple and complex) were analyzed for low, medium, and heavy users.

In (Kobsa & Fink, 2006) the authors were evaluating the performance of an LDAP-based UMS. The server was subjected to a number of simulated workloads. The test bed was configured to imitate user sessions representing various types of Web usage. Each type of Web usage had a click rate and a class of information of interest associated with it. The load was manipulated by varying the number of simulated users and their click rate. In addition, the authors varied the distribution factor of the UMS by installing its components either on one or on several machines. The authors performed a capacity planning test and obtained the request delay vs. request frequency curves in the case of the single machine hardware configuration. Request delays of single- and multiple-machine hardware configurations were compared with respect to several types of requests. In addition to overall "black-box" performance tests, the authors compared delay-frequency performance of the two inference components of the UMS for the single-machine configuration.

As an extension to the previous work, (Wang & Kobsa, 2009) revisited the problem of LDAP-based UMS performance. They focused on the distributed LDAP server, where each node is a part of a cloud that is as a whole responsible for providing user modeling support. The authors focused their attention on one node of the cloud and tested its ability to instantiate critical numbers of user session objects. Here a ram-up style capacity planning test is used. Only the peak part is evaluated – the costly process of initializing sessions. The rest of the procedures that comprise the lifecycle of the user session object are not run.

2.6.4.2 Our Approach to Performance Evaluation.

In our own work, we have performed a comparative performance assessment of the two versions of user modeling server (Yudelson et al., 2007; Zadorozhny et al., 2008): One using a pull approach to computation (computation is done only when resulting data is needed) and the other

using a push approach (all changes to the user model are made as soon as new data arrives). Three setups of the user modeling structure were used, each reflecting low, medium, or high amount of computation needed to aggregate new pieces of evidence about the user. Each of the setups of the UMS was then subjected to a series of simulated update streams constant in length and varying in the number of user model *write* requests. In the end, push and pull versions of the UMS were tested in the ability to respond to user model *read* requests. Results of the comparison gave clear advantage to the push version of the UMS. Although losing on user model writes, where push has to perform necessary computations and pull only stores the new information, the push version won with a significant margin on the read. Here push almost instantaneously forms a response to the query, since all the user model values are already computed, and push [PULL?] has to do lengthy computations. The results of the performed capacity-planning tests showed that a push-based UMS can effectively support up to several thousand users even on very modest hardware.

The novelty of the performance evaluation method we introduced in (Yudelson et al., 2007; Zadorozhny et al., 2008) is in the multiple stage design and integrating observed user behaviors into the evaluation process. The method consists of the following stages. First of all, before deploying the adaptation service, statistical data on user navigation behavior is collected to be used at a later time. At the second stage, the personalization service is deployed for a proof-of-the-concept testing. At this point one has to make sure it works properly and delivers the navigation support it was built for. Here the service is functioning under relaxed conditions. At the third stage, the personalization service is subjected to extreme loads for capacity-planning performance tests. The outcomes of this stage are the characteristics of the highest load at which the personalization service in question can function properly. And finally, we use the results of

the first stage (user navigation patterns), and the results of the third stage (highest load with proper functioning), to turn parameters of the load into the number of users that the service can effectively support without compromising performance.

2.6.5 Summary

In the literature today, there is a limited number of publications that recognize the importance of considering the component-based/distributed nature of the modern AS/AHS for the evaluation. The two implications of the complex modular structure of the AS/AHS are that: a) in addition to the success of the adaptation as a whole, contribution of each layer to the overall process of adaptation has to be studied in detail, and b) due to the tendency of deploying decomposed AS/AHS, where certain modules are given relative independence and often reside on separate servers, special attention has to be given to the performance issues of such distributed infrastructures.

So far almost all of the layered evaluation has focused on studying the process of adaptation itself and the role each of the AS/AHS infrastructure components plays in adaptation's overall success. The question of performance has not been addressed in the layered evaluation literature.

In the performance evaluation realm, the bulk of the work is devoted to studying the user modeling server (Carmichael et al., 2005; Kobsa & Fink, 2006; Wang & Kobsa, 2009; Zadorozhny et al., 2008). The performance issues of the multi-staged process of adaptation were addressed in one of our publications, see for example(Yudelson & Brusilovsky, 2008). And this publication only discusses a very basic adaptation technique.

There is a shortage of work on evaluating modern multi-component adaptation infrastructures. Recent advancements in the field of adaptive hypermedia create an increasing

gap between suggested and evaluated approaches to building complex AS/AHS infrastructures, especially when speaking of performance and scalability. Closing this gap is one of the foci of our work.

3.0 ADAPTATION FUNCTIONALITY AS STANDALONE SERVICE

This chapter discusses the implementation of the idea of full extraction of adaptation functionality. After a brief introduction it presents a general overview of the implemented prototype system – PERSEUS – and principles of its operation are given. Then, the discussion moves to the implementation details of PERSEUS and the description of its architecture and structure and purpose of its internal models. Conceptual evaluation of PERSEUS is given next. Here we evaluate whether adaptation extraction still allows implementation of the known methods of adaptation or at least a representative subset of them (those that are widely used in adaptive hypermedia today and range in computational challenge). After that, a detailed description of several adaptation techniques implemented in PERSEUS is presented. Special attention is given to the computational complexity of each of them. The chapter concludes with the results of PERSEUS's pre-study, including pre-study limitations.

3.1 INTRODUCTION

In our prior work on providing adaptation via intermediaries such as NavEx (Yudelson & Brusilovsky, 2005) and QuizGuide (Brusilovsky & Sosnovsky, 2005) we saw that implementing adaptive support independently from the main "portal" user interface shows great potential and helps bring the high value of adaptive navigation much more quickly than in the cases when

adaptive support has to be "built in". Having intensively researched the intermediaries, we can say that this method provides a partial solution to the problem. Each shell is typically designed for one class of resources (NavEx for WebEx annotated examples, QuizGuide for QuizPACK quizzes), and adding navigation support to one more resource of the same class was quite easy.

However, adding support for a new class of resources required a significantly larger amount of effort. For example, adding navigation support to SQLKnoT quizzes led to creating a specialized version of QuizGuide – SQLGuide (Sosnovsky et al., 2008). As a result the level of abstraction and reusability of the already implemented and tested adaptive navigation support tools was not high. This serves as an additional motivation for our idea that full extraction of adaptive methods would help to achieve greater reusability of the adaptation, and as a result would help speed up the process of providing adaptive value for the new content and reach a greater user population.

3.2 PERSEUS

PERSEUS is a standalone server of the adaptation functionality. It has been designed to empirically test the idea of separating adaptation functionality from the AHS and providing adaptation in a service-based fashion. Unlike built-in personalization or intermediary shell personalization solutions (see Section 2.5.3 for details), PERSEUS offers a full abstraction from the adaptation models and adaptation techniques it hosts. Client applications can obtain adaptation and personalization support from PERSEUS by communicating with it via a pre-defined network protocol. PERSEUS offers a set of reusable adaptive navigation support

strategies that can be deployed in several hyperspaces and across a virtually unlimited number of problem domains.

The conceptual idea of how PERSEUS works is shown in Figure 9. A system responsible for user interface (e.g., a content management system or a portal) provides access to a pre-constructed hyperspace. To render a personalized view on some node of the hyperspace, the portal consults with the personalization service engine. To do that, the portal sends the structure of the currently viewed portion of the hyperspace (consisting of a list of pre-authored resource links) and context information (e.g., user and group identities, device information, etc.). PERSEUS consults user modeling server(s) (and/or other data sources known to it) and performs the adaptation that was requested. The returned result is an updated structure of the hyperspace in the format that the portal can parse. The new structure of the node may have original links reordered or removed, new links inserted, or annotations added to links. Finally, the portal displays a new node view for the user.

The general scenario of how PERSEUS works is that it takes a list of links plus context information and returns an annotated list of links. To utilize services offered by PERSEUS, a portal has to comply with the format of the request for adaptation and be able to parse the response. Requests for adaptation are accepted as simple HTTP POST requests. The main parameter of the request is the structure of the page with links to be personalized (or the URL pointing to the location of this structure). This structure of the page needing personalization is in RDF[6] document / RSS 1.0[7] feed formatted. The use of RDF as a lingua franca for data interchange has become standard in modern AHS (2003; Denaux et al., 2005; Dolog et al.,

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

[7] RDF Site Summary http://web.resource.org/rss/1.0/spec

2004). The use of RDF/RSS 1.0, however, is not mandated. Any other format (such as plain text, comma or tab separated values, etc.) can be used. Since there is no central input parsing authority in PERSEUS, each adaptation service is free to enforce its own format as long as it conveys the necessary information – ordered link URLs accompanied by anchor text. PERSEUS's response follows the same format as the request with an addition of link annotation. Again there is no strict convention. Currently, PERSEUS supplies annotations in the form of HTML span tags that contain an image and possibly JavaScript code snippet for displaying explanatory tooltips.

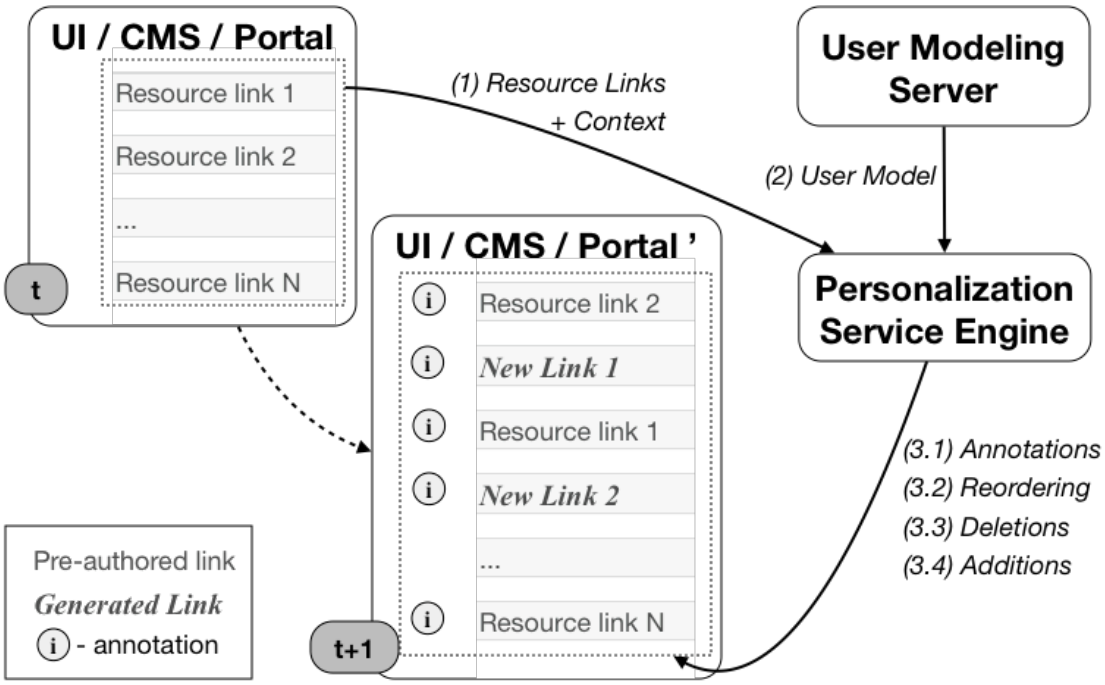


Figure 9. Scenario of personalization service engine's operation

PERSEUS was not built as merely a proof-of-concept for the standalone adaptation provision idea. At this moment it was already deployed in several courses at the University of Pittsburgh, Dublin City University, and Universidad Autónoma de Madrid (seeFigure 16 for examples). PERSEUS is also intended as a major adaptation and personalization provider for the

Ensemble Computing Portal[8]. See Figure 18 for the Ensemble portal screenshot with adaptive link annotation icons added as a mockup of the final result.

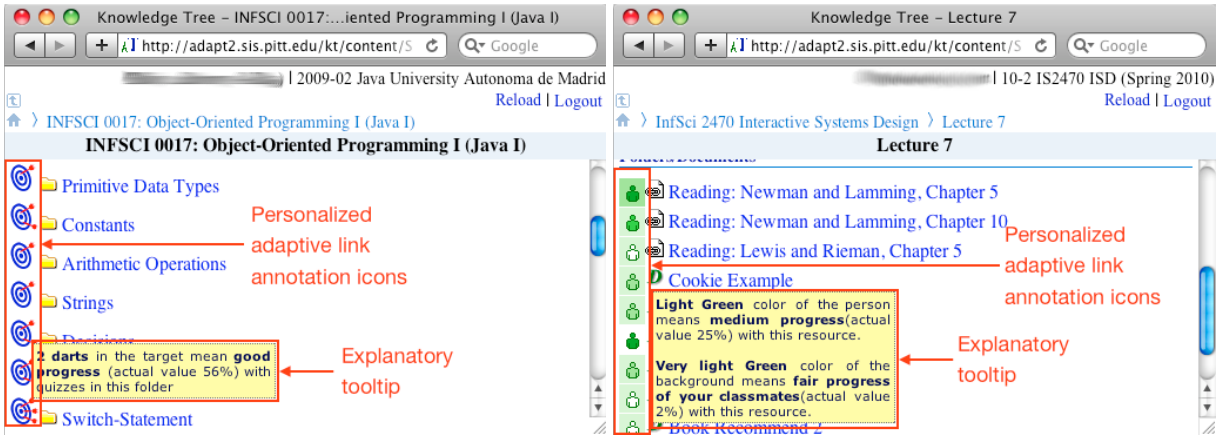


Figure 10. Examples of PERSEUS's deployment in actual courses offered via Knowledge Tree portal. Left – Java course for the Universidad Autónoma de Madrid, right – HCI course for the University of Pittsburgh.

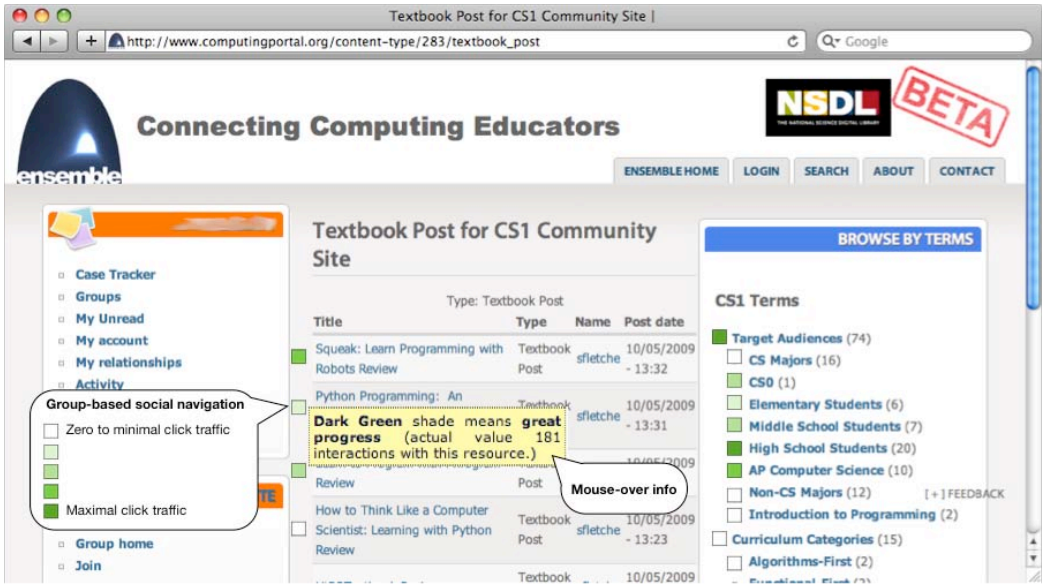


Figure 11. Mockup of anticipated PERSEUS deployment on Ensemble Computing portal. Green bullet icons and tooltips are to be provided by PERSEUS.

[8] Ensemble, NSF NSDL Pathways project (http://nsdl.org/about/?pager=pathways&subpager=ENSEMBLE)

3.3 PERSEUS IMPLEMENTATION

PERSEUS's architecture and the surrounding environment are shown in Figure 12. PERSEUS's internal structure is comprised of adaptation strategies, visualization model, and data model. All three components form the adaptation model of AHS architecture presented earlier in Figure 3.

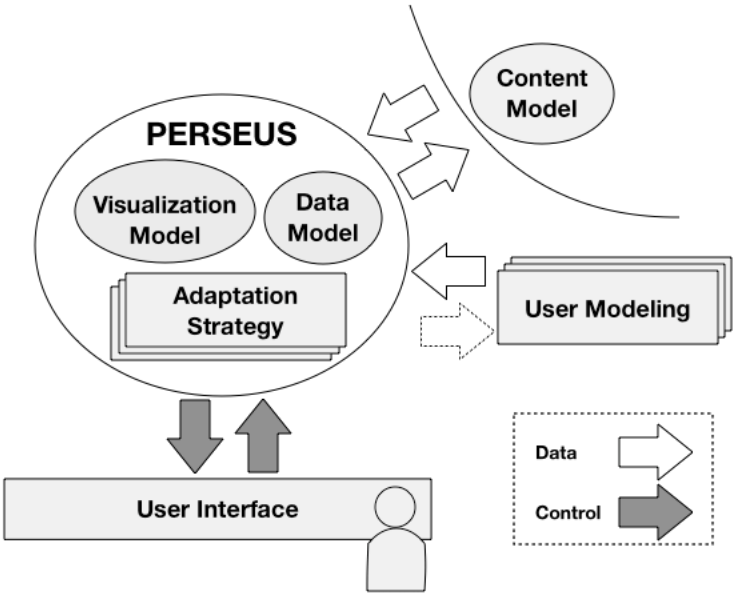


Figure 12. Architecture of PERSEUS and the surrounding environment

The **data model** plays a central role in structuring the process of acquiring data from the environment. PERSEUS works in close collaboration with one or several user modeling servers and content-managing applications (e.g., portals). Items of the data model – called data contexts – contain several types of information. These include addresses of the user modeling servers for acquiring the user data, access points to data stores (relational and/or semantic), URIs and URLs marking entry points to data stores, URI/URL patterns to filter/select resources of interest in the content model, etc.

Adaptation strategies, also called personalization services (Dolog et al., 2004), are the scenarios used while producing the adaptation. They prescribe what kind of data should be

obtained from the available sources and how it should be processed. The sources are specified in data contexts or could be the results of another adaptation strategy (chained adaptations). Relative independence of adaptation strategies allows them to be (re)used with multiple data contexts.

One of the most effective classes of adaptation techniques in general is adaptive annotation (Brusilovsky, 2001). In PERSEUS, adaptation techniques that provide adaptive link annotation utilize the visualization model. The **visualization model** is comprised of special entities called visualizers. Visualizers contain rules prescribing how the continuous data produced by adaptation strategies (for example, various measures of interest, importance, knowledge, etc.) can be visualized for the user. Often these rules control how ranges of continuous values are mapped to discrete visual annotations. Visualizers, just like the data contexts, are relatively independent from adaptation strategies. This allows several different visualizers to be selectively used with one adaptation strategy and also the same visualizer to be used with different adaptation strategies.

There are two basic types of annotations supported by visualizers: icons, and styles. Icons are displayed next to a link, for example a vertical progress bar icon can discretely visualize five levels of user progress, from 0% to 100% with 25% steps. A visualizer using these cues would have to bin continuous progress measure to these five levels and assign an appropriate icon. Styles are used to change the appearance of the adapted link. For example, instead of the five-level progress bar icon mentioned above, an eleven-level background color intensity could be used for the same purpose. Alternatively, a boldfaced font style could be used to denote insufficient progress and so on. Visualizers can combine several messages to the users by encoding them in the visual cues, be they icons or link anchor decorations.

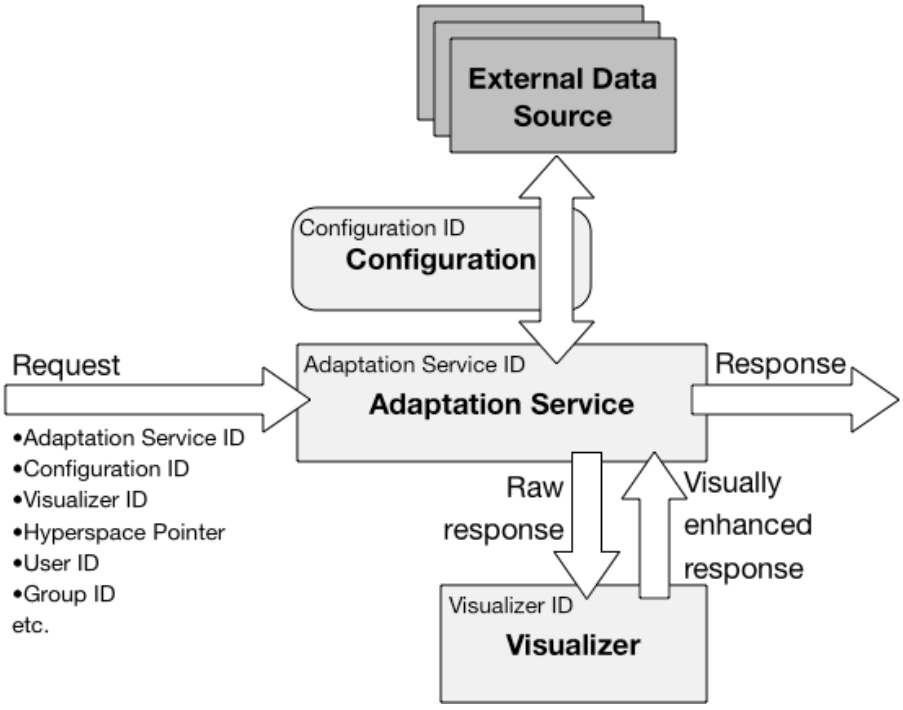


Figure 13. Data and control flows during PERSEUS adaptation service lifecycle

Each adaptation strategy is invoked in PERSEUS by an HTTP POST call. The call is made to the base URL of PERSEUS. Parameters help PERSEUS identify what adaptation functionality to launch, in what context, and include the following (see Figure 13).

- The adaptation functionality identifier is mandatory. It determines the actual implementation of an adaptation procedure that is requested.
- The data context identifier is mandatory. It determines what data sources the adaptation functionality has available for its operation.
- The visualizer identifier is optional and prescribes a particular method of visualizing raw adaptation values.
- The hyperspace pointer is mandatory in most of the currently implemented adaptation methods in PERSEUS. The hyperspace pointer is the specification of the link structure to be adapted. For each adaptation procedure the content, format, and the meaning of this parameter can be defined individually. In most cases, the hyperspace pointer identifies the

portion of the hyperspace that requires adaptation. Using this identifier, the adaptation procedure should be able to reconstruct the link structure of the hyperspace subset.

- User identity (login) is not always mandatory, but is required when personalization is performed. There exist methods of adaptation that do not distinguish individual users (e.g., group-based). In the case of PERSEUS, most of adaptation procedures, however, do discriminate between users. User identity is often used for acquiring user profile data or user model (Brusilovsky et al., 2004). Addresses for user data inquiry are specified in the data context.
- Group identity is not mandatory; however, sometimes forms of adaptation are done on a group level. For example, social navigation can visualize individual progress alongside group progress.
- Other parameters can be added on an individual adaptation procedure basis.

PERSEUS maintains a record of all implemented and available adaptation techniques. Data contexts are matched to all adaptation techniques and can be reused across several of them. Individual items of data contexts, such as addresses of the external data sources, URIs/URLs of specific resources, etc., can also be present in multiple data contexts. Each visualizer is tailored for a specific type of output that an adaptation procedure produces. If the structure and the meaning of several outputs match, one or several visualizers could be used interchangeably for both.

3.4 CONCEPTUAL EVALUATION OF PERSEUS

The question of whether various adaptation techniques are effective in a human-computer interaction sense has already been studied thoroughly enough. The general conclusion is that each of them works well given the proper choice of the task, domain, target user population, and context. In our work on PERSEUS we will not re-evaluate the individual adaptation methods. Instead, we will focus on the idea of providing adaptation in a standalone fashion. Fortunately, conceptual evaluation of the idea itself can be done separately from empirical evaluation of the actual system prototype – PERSEUS. In this section we are abstracting from implementation and focusing on the conceptual design of PERSEUS, namely, whether the system is capable of setting the basis for implementing known adaptive navigation support techniques classified by Brusilovsky (Brusilovsky, 1996; Brusilovsky, 2001) and whether it is in agreement with the commonly accepted structure of the adaptation process.

3.4.1 Support of Known Adaptation Techniques

In the literature there are only a few publications that address the question of AHS's flexibility in providing a range of adaptation techniques and/or capability to support a set of reasoning engines. Here we could mention two works: one by De Bra and colleagues (Bra et al., 2006) and the other by Bailey and colleagues (Bailey et al., 2007). In both publications, the authors discuss not only the architecture of their adaptive systems, but also how the architecture enables them to deliver a set of know adaptation techniques classified by Brusilovsky in (Brusilovsky, 1996) and (Brusilovsky, 2001). While the work of De Bra et al. (Bra et al., 2006) simply goes over adaptation methods confirming and explaining that a certain method is supported, in (Bailey et al., 2007) the authors focus their discussion on the process of transforming/manipulating the

hypertext content to fulfill the goal of adaptation. The fundamental open hypermedia model (FOHM) that the authors present allows them to show procedural similarities between a number of pairs of adaptive navigation support and adaptive presentation techniques (see Figure 14 below). These similarities can be exploited to unify the implementation of the techniques (e.g., adaptive link sorting is identical to sorting fragments with the only difference in the granularity of the manipulated entities) and thus make the underlying system more flexible and capable. As a result, such an adaptive system should implement a reduced set of methods to be able to support a large set of techniques.

As stipulated in (Bailey et al., 2007), among the variety of adaptation techniques classified by Brusilovsky in (Brusilovsky, 1996) and (Brusilovsky, 2001), several operate on identical principles – for example, adaptive link sorting (from the adaptive navigation branch) and fragment sorting (from the adaptive presentation branch), adaptive link hiding (from the adaptive navigation branch) and removing sorting (from the adaptive presentation branch), etc. This is why we are going to assess PERSEUS's potential in support various adaptation techniques on a smaller set of techniques. Yet, using the above-mentioned argument, the scope of PERSEUS's applicability could be extended to other adaptation techniques easily. Below we will examine a set of techniques and align their implementation against PERSEUS's conceptual scenario of operation discussed in Section 3.3 and in Figure 9.

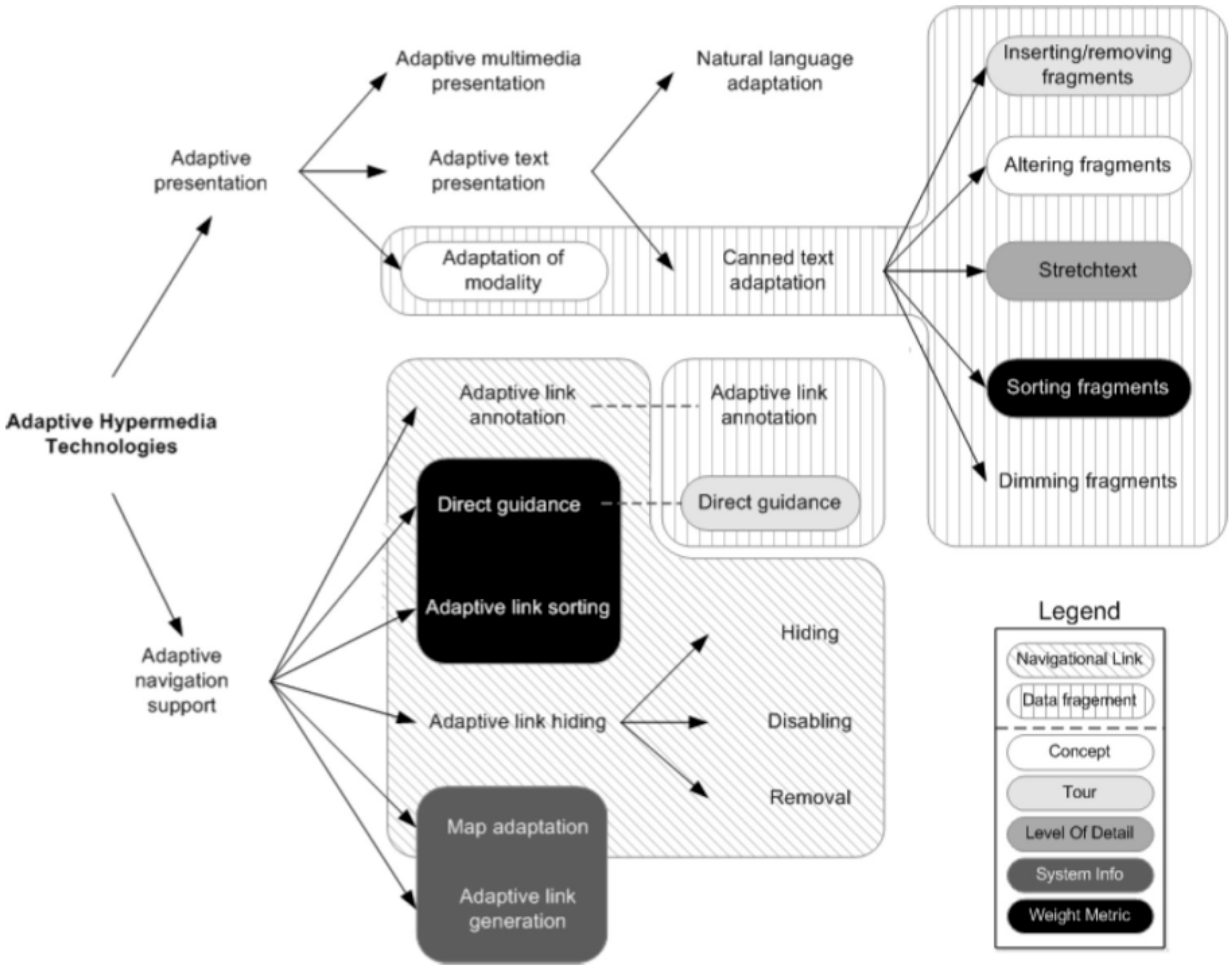


Figure 14. Brusilovsky's taxonomy (Brusilovsky, 2001) of AH techniques showingtheir similarities as a basis for unification of the implementation (Bailey et al., 2007)© 2007 Association for Computing Machinery, Inc. Reprinted by permission.

Adaptive link sorting (and sorting fragment by analogy) alters the order of the received list of links based on some criteria. A PERSEUS service implementing this technique would simply reorder the supplied list of link URLs and anchor labels and return it back to the requestor.

Link hiding, disabling, and removing (and removing fragments by analogy). Removing a link is simply purging it from the input list. Link disabling would require keeping the link anchor text but changing the link URL to a blank (for example, a hash sign – #, or

`javascript:void(0);`). Changing the style of the link's anchor text decoration would effectively achieve link hiding. However, that would only make sense if links were surrounded by text; otherwise, if links are presented in the form of a list, the hiding is less effective.

Direct navigation is a variation of adaptive link sorting and link removal, where only the top link from the input list (sorted by some relevance criteria) is returned), while others are removed.

Link generation (and inserting fragments and stretchtext by analogy) is achieved by inserting previously a absent link into a received list at a position appropriate in a given context.

All methods discussed above (except for link hiding) manipulate links; link annotationuses a separate output type reserved for this particular purpose. Annotations (including empty annotations) are a list of style changes and/or visual cues that correspond to each link in PERSEUS's output. The number and the order of annotations correspond to the resulting list of links itself. Style annotations alter font type, weight or size, underlining, text color, foreground and background color, etc. Visual cues include textual additions, graphic icons, tooltip messages, etc.

Map adaptation in PERSEUS could be viewed as a special case of intersecting link removal, link generation, and link annotation, where the presence or absence of links (removal and/or generation) and relative location of links (coordinates as a type of annotation) are combined.

As we can see, out of 16adaptation methods inthe classical taxonomy (see Figure 2), only 3 methods that cannot be conceptually covered by PERSEUS are: adaptive multimedia presentation, natural language adaptation, and adaptation of modality. PERSEUS effectively supports he rest of the methods.

3.4.2 Composition of the Adaptation Process

Knutov and colleagues in (Knutov et al., 2009) focused on the decomposition of the generic process of adaptation in an adaptation engine. The basis for the adaptation decomposition is the classification criteria for the adaptation techniques presented in (Brusilovsky, 1996). These criteria, refined in (Knutov et al., 2009) are formulated in the form of questions listed below.

- Where to adapt (application areas)?
- When to adapt (context)?
- What to adapt (underlying domain/hyperspace structure)?
- Why to adapt (goals)?
- To what to adapt (user features)?
- How to adapt (adaptation methods)?

These six questions capture the full spectrum of the issues that every AHS has to address. They offer *degrees of freedom* that an AHS designer can explore. Often, several of these dimensions are controlled and variability is offered only for one. For example, the decision of when to adapt could be limited to only one option – always to adapt – as opposed to a more flexible solution when necessity for adaptation is dynamically estimated. The choice of the adaptation method (how to adapt) could be fixed (if the system only offers one kind of adaptation support) or varied based on some criteria. With respect to these six adaptation questions, PERSEUS is positioned as follows.

Where to adapt (application areas)? PERSEUS is intended as a general engine capable of providing adaptation in virtually any application area. In this work we are going to consider the area of educational AHS and forgo the discussion of applicability.

When to adapt (context)? The decision of when to provide adaptation is orthogonal to PERSEUS's conceptual design. One might imagine an adaptation service implemented in PERSEUS having a decision procedure intended specifically for adaptation appropriation. This procedure could either be encapsulated into a particular adaptation service or be available to several PERSEUS adaptation services. We are not going to discuss adaptation appropriation in this work.

What to adapt (underlying hyperspace/domain structure)? PERSEUS assumes that the structure of the hyperspace being adapted is supplied along with the adaptation service invocation call. Domain structure, if necessary, is supposed to be linked via data model contexts.

Why to adapt (goals)? The goal of adaptation in general and adaptation provided by PERSEUS is to personalize the view of the hyperspace to the characteristics of the individual users and prevent the situation of being lost in the hyperspace (Hammond, 1989).

To what to adapt (user features)? PERSEUS was originally intended for use in the educational AHS and hence adapts to user knowledge. However, there is no limitation on what linked sources of user-related information each PERSEUS adaptation service can use.

How to adapt (adaptation methods)? Client application makes a decision about which PERSEUS adaptation service to invoke; one could implement a meta-adaptation that would be able to first make a decision on what adaptation is most appropriate in a given context and then forward the request to it.

3.5 ADAPTATION TECHNIQUES IMPLEMENTED IN PERSEUS

As we can see, the conceptual design of PERSEUS sets the basis for realization of nearly all of the major adaptation methods. However, these techniques describe an interface outcome of the adaptation process. The essence of the adaptation lies in the decision support mechanism – the reasoning engine. Since the design of PERSEUS does not regulate the way internal processes of the adaptation should be coded, PERSEUS can support virtually any reasoning engine.

In this section, we will show that PERSEUS is capable of providing a basis for realizing various adaptation engines by implementation. We are going discuss four adaptation services implemented in PERSEUS that represent different adaptive reasoning engines: social navigation support service represents a social reasoning engine and is of low computational challenge; topic-based navigation support represents a concept-based engine with coarse-grained concepts and poses a medium computational challenge; concept-based navigation support represents the concept-based engine and presents high computational challenge; and finally link-recommendation service is a wrapper of an external recommender that represents a concept-based engine and poses high computational challenge.

The first three adaptation services (social, topic-, and concept-based) implement adaptive link annotation, while the last one (link-recommendation) implements links generation. Adaptive link annotation is one of the most frequently used adaptation techniques. It is based on the principle that link structure of the original hypermedia document remains the same, and the only changes are the way links are made to stand out using text decoration or text/graphic annotation markers placed next to them. The choice of the adaptation technique in our case does not influence the core complexity of the adaptation engines. Social and concept-based adaptation engines are one of the most popular adaptation engines today. With the rapid development of

social media and social networks, social engine (used with several adaptation techniques, including hybrid solutions) is the most widespread. Concept-based adaptation is frequently used in education hypermedia and knowledge-based systems. In several instances keyword-based information retrieval engines and personalized search engines could be viewed as isomorphic to a concept-based approach. Topic-based engine in comparison to social and concept-based is less widespread, but is useful in our case as its computational complexity is in between those of social (low) and concept-based engines.

In this section, for each PERSEUS's adaptation service implementing one of the chosen adaptation engines we will state the intended scope of their use, outline interface outcomes, and identify the sources of computational complexity connecting it to the reasoning engine chosen.

3.5.1 Social Navigation Support Adaptation Service

Social navigation is the first adaptive navigation support technique implemented in PERSEUS. An instance of this technique requires a standard set of inputs: user and group identities and a pointer to the hyperspace fragment that needs personalization. The social navigation implemented in PERSEUS is built for content rather than index nodes of the target hyperspace. Content folders have terminal non-container nodes – resources. For each of these resources an individual and average group progress is retrieved from the user modeling server, here CUMULATE (Brusilovsky et al., 2005). Progress here is abstracted by CUMULATE. For example, in the case of a problem-solving exercise, progress is the level of student's mastery of the underlying knowledge, and in the case of a document intended for reading progress is the normalized time spent reading (Claypool et al., 2001).

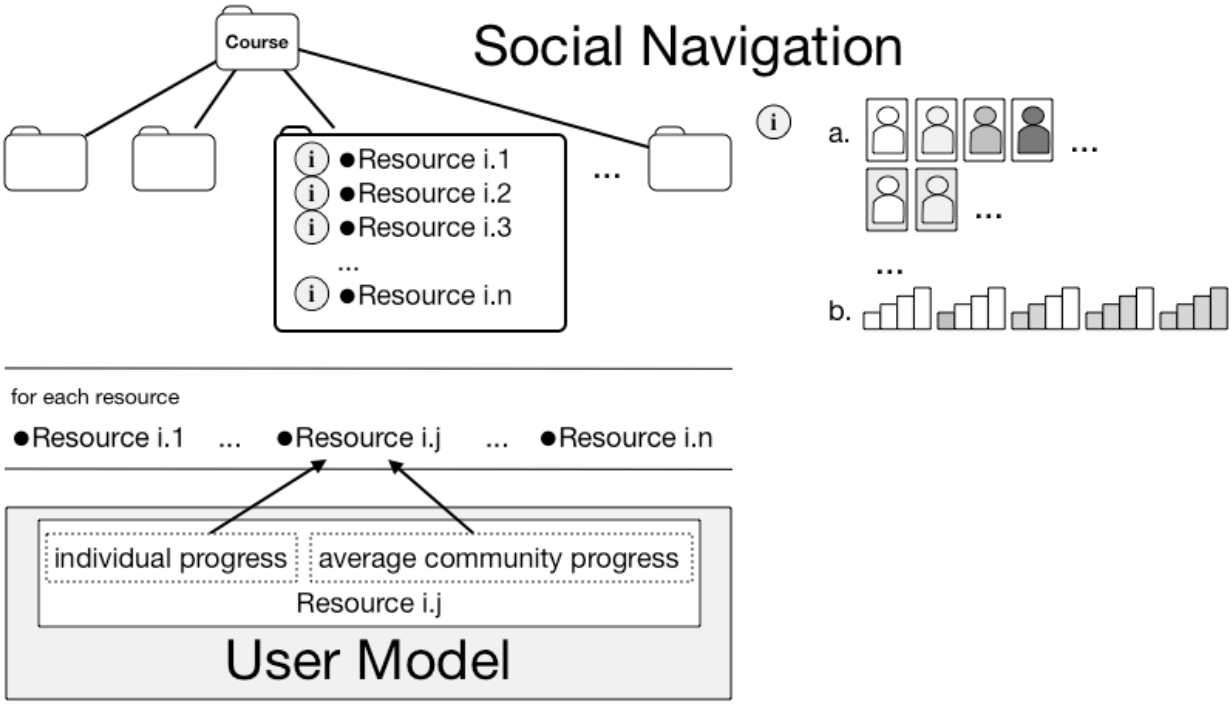


Figure 15. Social navigation support adaptation service

In terms of interface changes, the social navigation service adds a visual cue for every link that has user/group navigation offprint. To produce the cues, continuous values of user/group activity are binned. There are two major sets of cues. Set a. (see Figure 15) is comprised of person-on-the-background icons. This icon was first introduced in (Brusilovsky et al., 2004). Here the shade of the person shows the individual's progress, while the shade of the background shows average group progress. In both cases, the darker the shade, the bigger the progress. Set b. conveys only one type of information in the form of colored bars; this information can represent either personal progress (and the social nature of the navigation support will be lost) or the average group progress.

The computations done by this social personalization service are very simple. They are reduced to user model lookup, link matching, and cue selection. For each of the adapted links, the social navigation service has to lookup one-two values from the back-end user model: user's

individual progress and, optionally, average progress of the group. Progress is an indicator of user activity in terms of time, number of clicks, or success rate. The social navigation service, being the first to be implemented, was the most studied and underwent a formal evaluation (see Section 3.6 for details).

Figure 26 presents a control flow diagram of the social navigation support service lifecycle. At stages S1 and S7 the client requests adaptation and receives the response. At stage S2 initialization takes place. Here inputs are parsed and the object structures necessary at the further stages are created. Stages S3, S4, and S5 are devoted to requesting the user's individual and group progress from the user modeling server. At stage S6, the service parses user model response, makes an adaptation decision, and serializes the response for the client. The amount of data that has to be processed and the processing time depend on the size of the input (number of resources to adapt) in a linear fashion.

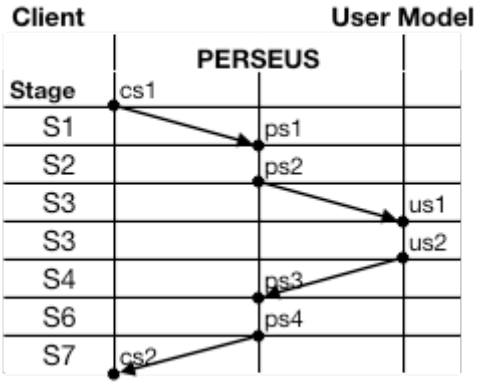


Figure 16. Control flow diagram of the social navigation support service. Capital S marks various stages of the processing, *cs* marks events on the client, *ps* – on PERSEUS, and *us* – on the user model side. Arrows denote transfer of control.

3.5.2 Topic-Based Navigation Support Adaptation Service

The classical topic-based navigation support adaptation technique has been developed and evaluated in (Sosnovsky & Brusilovsky, 2005). The rationale behind it is that before following a link leading to a set of resources (united under a common topic), the user should know what awaits him/her there in the form of a short summary. Each of the topics represents a large chunk of the domain knowledge. The number of topics should be no more than a few dozen per domain, since topic-based navigation is not aimed at fine-grained domain models but rather at high-level conceptualization.

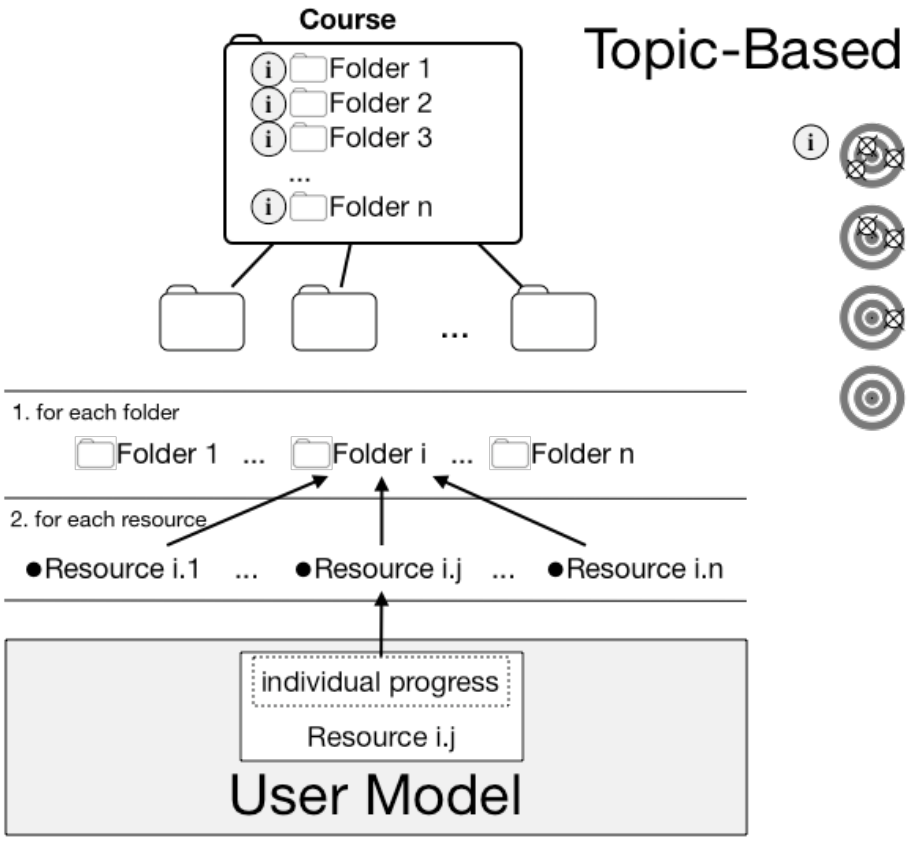


Figure 17. Topic-based navigation support adaptation service

PERSEUS's implementation follows the classic example and is fairly similar to the social navigation service in terms of inputs. User and group identity and pointer to the hyperspace

fragment are required. The major difference of the topic-based service, however, is that it is intended for the index nodes of the hyperspace (folders-of-folders). The folders inside the folder – topics – contain terminal resource nodes. The purpose of the services is to come up with a summary for these topic folders.

Before querying the user model, the topic-based service builds a map of the folders' content and retrieves the resource links from them. Then it requests user progress with respect to each resource and aggregates user progress within each folder. Next, progress that the user achieved with several types of resources (progress with some non-interactive resources, for example links to lecture slides, is not counted towards topic progress) is aggregated by the folder and visualized as a target with 0, 1, 2, or 3 darts (Figure 17). The greater the progress, the more darts there are.

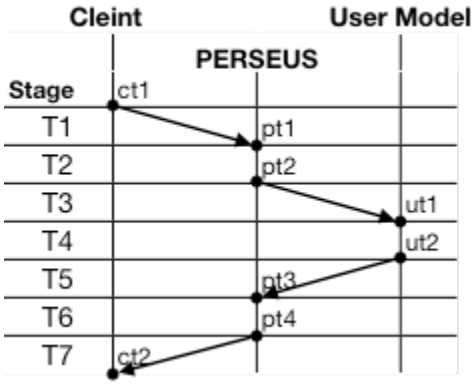


Figure 18. Control flow diagram of the topic-based navigation support service. Capital T marks various stages of the processing, *ct* marks events on the client, *pt* – on PERSEUS, and *ut* – on the user model side. Arrows denote transfer of control.

Figure 18 shows a control flow diagram of the topic-based navigation support service lifecycle and is isomorphic to the control flow diagram of the social navigation support service (see Figure 16). At stages T1 and T7 the client requests adaptation and receives the response. Stage T2 is the initialization phase, where inputs are parsed and internal use object structures are

created. Stages T3, T4, and T5 are devoted to querying the UMS for the user's progress with the terminal resources in the sub-folders. At stage T6, the service parses user model response, makes an adaptation decision, and serializes the response for the client.

Both social and topic-based services work with aggregated user progress. However, the social navigation service relies on aggregated (average) progress of peer users with a particular resource. The social layer of the user model conveniently produces this value. In the case of the topic-based navigation support service, the aggregation (weighted averaging) is done across individual user progress values with resources in a folder. The user model is traditionally agnostic of the folder-resource structure of the hyperspace and the aggregation has to be done by the service itself. Because of this, the topic-based service has to manipulate a larger array of data, not only the list of folders requiring adaptation, but the lists of resources in them. The size of the data exchange with the user model is hence larger. The amount of data that has to be processed and the processing time, unlike the social navigation service, does not depend on the size of the input in a linear fashion: It is rather super-linear.

The topic-based navigation support service has been implemented and proof-of-the-concept tested in a number of classes already. However, its performance characteristics have not been properly evaluated.

3.5.3 Concept-Based Navigation Support Personalization Service

The core idea of the concept-based navigation support is using estimated user knowledge of domain concepts not only to obtain aggregated per-resource progress, but also to exploit the relations between concepts and between resources via concepts, as well as distinguishing prerequisite and outcome concepts for each resource. The concept-based adaptation service

implemented in PERSEUS is based on the legacy approach introduced in the system NavEx (Yudelson & Brusilovsky, 2005).

In this approach, on top of the concept indexing that is standard in the majority of the knowledge-based systems, additional focus is given to the roles of the concepts. The distinction is made between outcomes – the concepts that are directly addressed by the resource – and prerequisites – concepts that serve a secondary support role. This distinction was not a novelty in NavEx, but often required by-hand expert work (Henze & Nejdl, 1999; Smith & Blandford, 2003). In NavEx the split of the concepts into prerequisites and outcomes was done on the basis of the course structure. In the C language course that NavEx was built for, resources were grouped into ordered lectures. Resources that were in the very first lecture were set to have outcomes only. The concepts of the resources from each of the following lectures were split on the basis of whether they overlapped with the concepts of the resources from prior lectures or not. The overlapping concepts became prerequisites; the new, previously unseen concepts became outcomes. This procedure, although relying greatly on the quality of resource grouping into the ordered lecture folders, offered a fast and easy way to process concept metadata automatically.

Existence of the two types of concepts – prerequisites and outcomes – created an opportunity to determine whether the user was ready to work with the resource or not. If all of the prerequisite metadata concepts were mastered well enough (e.g., by working with resources from prior lectures), the resource was rendered to be ready-to-be-learned, and not ready to learn otherwise. This feature of the resource hyperspace was exploited in NavEx in the form of zone-based adaptation (Weber & Brusilovsky, 2001). Resources that were not ready to be learned

were zone 3, resources ready to be learned but not fully learned were zone 2, and the rest were zone 1.

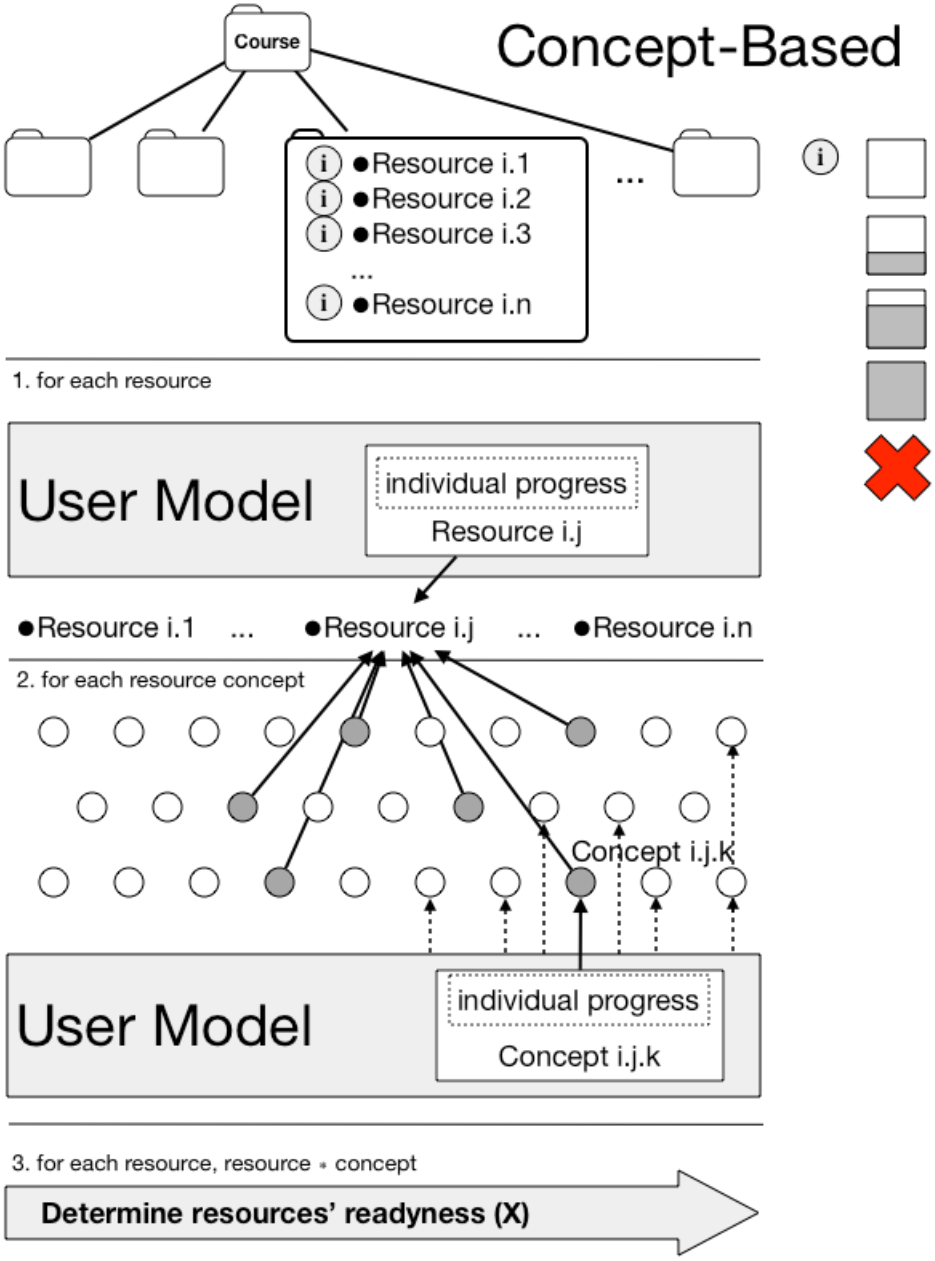


Figure 19. Concept-based navigation support adaptation service

The concept-based navigation support realized in PERSEUS implements an updated version of the adaptation featured in NavEx. Just as in NavEx, for every hyperspace where the concept-based service is intended to be used, a procedure of prerequisite-outcome separation has

to be run beforehand. This procedure in PERSEUS is implemented as a companion service, and its results are then statically bound to the data context(s) of the main concept-based service. Since in most of the cases the structure of the hyperspace does not change often, the companion procedure should not be run frequently.

With the prerequisite-outcome data available, the concept-based service can determine the readiness of the resources; if they are not ready to be learned, the service produces a red x-shaped annotation icon (zone 3). Resources ready to be learned are annotated as per their progress with a vertical filled bar (zone 1 if wholly filled, and zone 2 if partially filled). The progress (and respectively the filling level of the bar) is computed as an average knowledge of the outcome metadata concepts (see Figure 19).

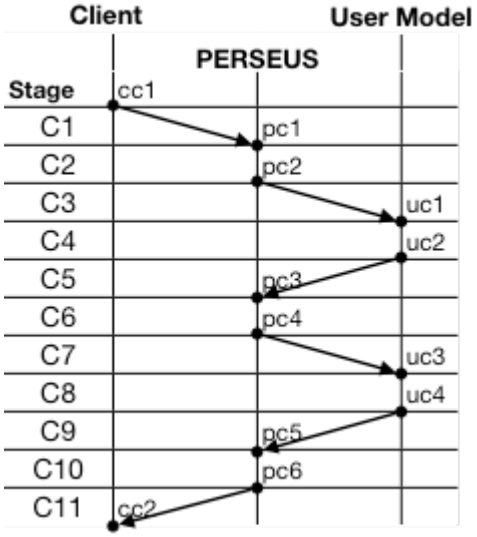


Figure 20. Control flow of the concept-based navigation support service. Capital C marks various stages of the processing, *cc* marks events on the client, *pc* – on PERSEUS, and *uc* – on the user model side. Arrows denote transfer of control.

Figure 20 shows a control flow diagram of the concept-based navigation support service lifecycle. At stages C1 and C11 the client requests adaptation and receives the response. Stage C2 is the initialization phase, where inputs are parsed and internal use object structures are

created. Stages C3, C4, and C5 and stages C7, C8, and C9 are devoted to querying the UMS for the user's progress with the resources and for user mastery of the concepts. It does not matter which query comes first. At stage C10, the service parses user model responses, makes an adaptation decision, and serializes the response for the client.

In terms of performance, the concept-based navigation support adaptation service is more computationally demanding than social or topic-based. There are several reasons for this. First, concepts – finer-grained metadata – increase the amount of data the service needs to query the user model for. In topic-based service there are several resources per each topic, while in concept-based service there are up to several dozen concepts per each resource. This results in a larger traffic between PERSEUS and the user modeling server and higher processing and aggregation costs in general. Additionally, the concept-based service still has to query for the user's individual resource progress values.

Second, the zone-based adaptation (in particular zone 3 – not-ready-to-be-learned resources) requires consideration of user concept progresses for all resources in the course. Namely, regardless of the links on the page currently being adapted, the concept-based service needs progress for concepts related to all resources in the hyperspace. Thus, in addition to computing the readiness for the resources in question, the service has to perform a linear "scan" of potentially the entire hyperspace (see Figure 19).

In our case, the structure of the adapted hyperspace changes very seldom and one does not have to run the prerequisite-outcome miner companion service frequently: only when there are changes made. However, in a more dynamic hyperspace, prerequisite-outcome separation would have to be run more often, thus increasing the computational complexity.

The concept-based navigation support service has been implemented and proof-of-the-concept tested in a number of classes already. Its performance evaluation has not been done yet.

3.5.4 Link Generation (Recommendation) Personalization Service

The link generation (recommendation) service in PERSEUS is one of the most recent additions to the fleet of adaptation methods. In contrast to the three personalization services described above, its main purpose is not to annotate the existing resources, but to generate new resource links (although annotation is added to differentiate statically authored links from generated). The rationale behind resource recommendation in our case is the following. There exists a structure of the course hyperspace pre-authored by the teacher. In addition there exists a large pool of relevant resources that are not statically connected to the course. It would be cumbersome to statically bind the resources from the pool to the course hyperspace. As an alternative way, a special personalization service is set up to recommend relevant resources given the list of resources currently in the user's view and/or given the state of the user model.

The recommendation service is also different from the rest of the previously described services in that it does not implement all the functionality pertinent to adaptation. Instead, it relies on an external recommender and serves as a wrapper to it. Although the chosen wrapper design pattern adds an additional communication link (and potentially overhead), it allows the recommendation engine to be improved independently, without changes to the service.

The recommendation service is intended for the content non-terminal nodes (content folders) of the hyperspace just like the social navigation and the concept-based services (see Figure 21). The functionality of producing the recommended resource links is outsourced to an external recommender engine for which the service becomes a wrapper. The concrete

recommendation engine at this point does not matter; it could be knowledge-based, content-based (Pazzani, 1999), as well as collaborative (Herlocker et al., 1999), for example.

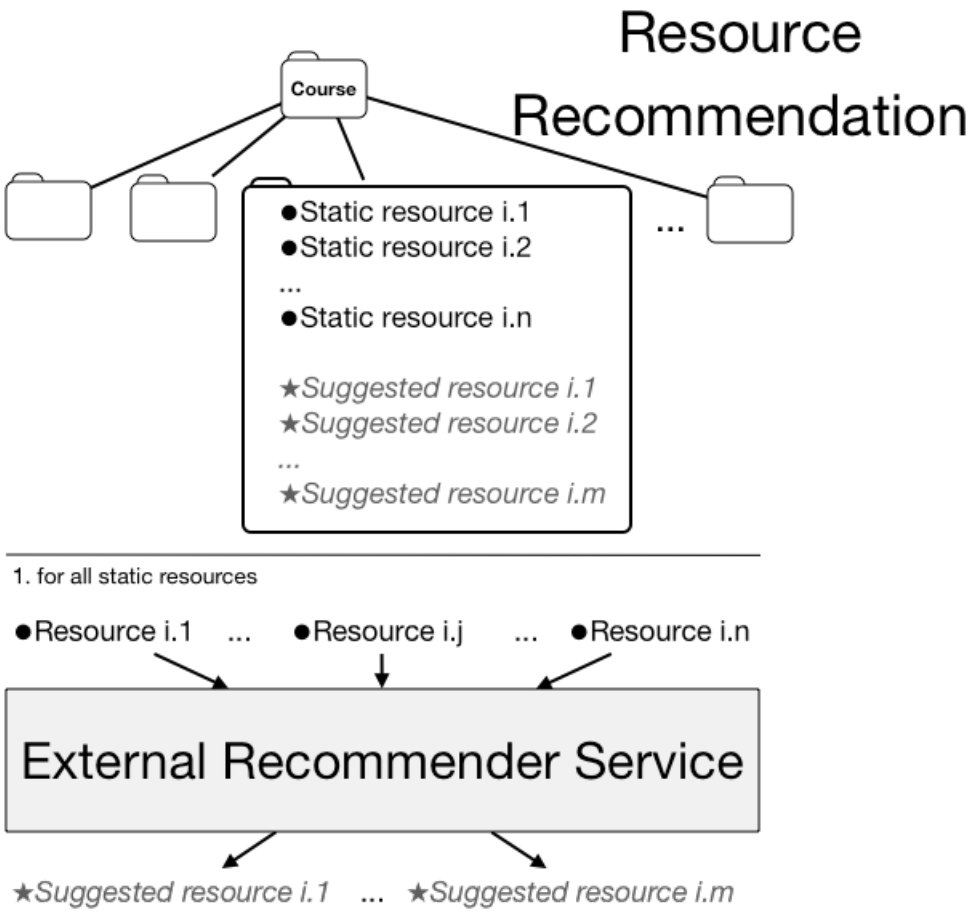


Figure 21. Recommendation service

3.6 PRE-STUDY OF PERSEUS'S SOCIAL NAVIGATION ADAPTATION

In this section we will discuss the first study of PERSEUS. In this study we empirically evaluated performance of the social navigation support service. We will start with the pilot study

and then cover the setup of the main experiment and data collection procedure used. Next we will present study results. We will finish with the limitations of this study.

3.6.1 Pilot-Test

We began the empirical evaluation of PERSEUS's social navigation support adaptation service with a proof-of-the-concept pilot test. As part of the pilot-test, the social navigation support adaptation service was deployed during the Fall 2007 semester at the School of Information Sciences, University of Pittsburgh, on a portal where materials were set up for an undergraduate database course. The class had 37 students. The course was very intensive and, as a result, the amount of student work exceeded our expectations. The load our adaptive tools had to cope with was quite high. Overall each student made over 50 accesses to the index pages serviced by PERSEUS's social navigation. The maximum number of accesses per minute was 15, the median was 1, and the 95th percentile was 7. Every call to the personalization service contained about 20 resources awaiting personalization. It required the personalization service engine to construct models of 103.5±0.86 RDF triples. Each such call took the personalization service engine 43.5 ms, on average, to complete. This means that the personalization overhead was .42 ms per RDF triple or a little more than 2 ms per resource. The results of the pilot evaluation have shown that even though the active usage of adaptive tools created a very adaptation-intensive environment, this did not challenge our social navigation service engine enough.

3.6.2 Experimental Setup and Data Collection

The next and major step in evaluating PERSEUS's social navigation service was to test it under heavy loads and determine the loads critical to its performance. To accomplish this, we used a technique previously employed by us and described in (Yudelson et al., 2007; Zadorozhny et al.,

2008). We set up two machines: one with the personalization service engine, and the other with a special "flooder" installed. The role of the "flooder" was to imitate the personalization service client, subject the personalization engine machine to various types of load, and record the observed parameters (Figure 22).

To simplify the experiment, we decided to use discrete values for the load. The parameters of the load were the following.

- **Complexity of the request**. The majority of our portal lecture folders had roughly 20 resource links to personalize (this includes lecture folders of the pilot course described in Section 3.6.1 and several other courses that are deployed on the portal). In addition to that we used two more values for complexity: 5 resources, to represent a "lightweight" folder, and 50 to signify a folder "overloaded" with resources. Thus we had 3 values: 5, 20, and 50 resources per request (35, 125, and 305 RDF triples respectively).
- **Request delivery rate** – delay between consecutive requests. From our experience with user modeling services (Yudelson et al., 2007; Zadorozhny et al., 2008) and initial experiments with a personalization service engine, we had already learned that a delay of 10 ms between requests is critical for our hardware/software configuration. In addition, delays between requests of 160 ms and more did not present any challenge. Hence, we varied the request delivery rate parameter between 10 ms and 160 ms. Rates between boundaries were doubles of the previous value, giving us 5 loads: 10, 20, 40, 80, and 160 ms.
- **Duration of the load**. From prior experimentation we knew that the duration did not really matter, unless it was a peak load of 10 ms or 20 ms between requests. During these peak loads, the personalization server would stop responding to any requests at all after 30

seconds. We decided to keep the load sessions fairly short – a little less than 4 seconds (3,840 ms, divisible by all delivery rates).

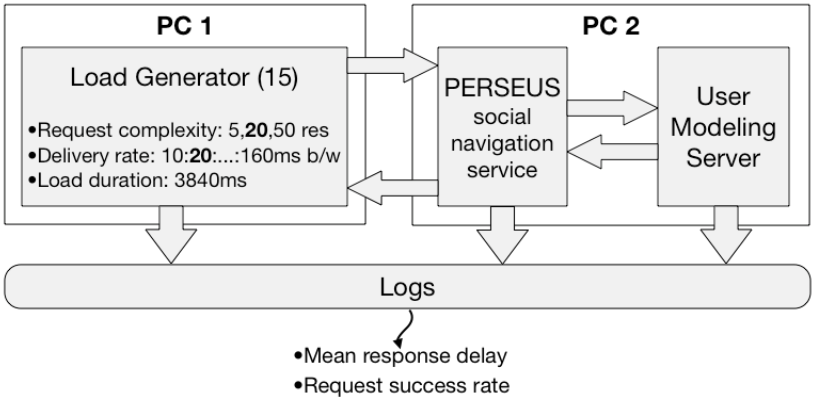


Figure 22. Pre-study test system setup

To obtain more data we repeated the flooding sessions 5 times for each of the three request complexities and each of the five request delivery rates, giving us 3 x 5 = 15 different settings. During these sessions we observed the following parameters:

- **Mean response delay** – the average amount of time it takes to complete a request.
- **Request success rate** – denoting the fraction of requests that are completed successfully. For the least demanding load of 160 ms between requests, the amount of requests sent per each flooding session was 3,840/160=24. For the highest load of 10 ms between requests, it was 3,840/10=384.

The personalization service engine was run on a machine with Pentium 4 dual core 2.8Mhz processor and 1Gb RAM. The user modeling server that the personalization service engine depended on was running on the same machine. To compensate for the high speed of the school's wired network, we used a Wi-Fi network to communicate with the personalization engine. It also provided a realistic scenario for students who would be accessing adaptive content outside their fast university campus LAN.

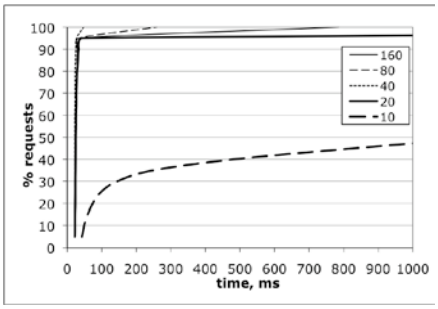
3.6.3 Results

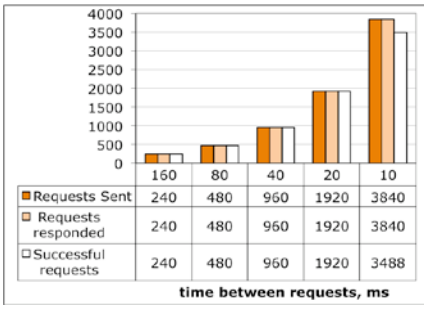
Figure 23 shows a summary of the evaluation results. Charts in the left column are the percentile plots. Each curve there corresponds to one of the five request delivery rates. Each point on a percentile plot indicates the maximum response delay (x-axis) for the given percentile of requests (y-axis).

The right column of charts denotes request success. For each of the request delay rates, we show only the total number of requests sent, the number of requests responded to – both successfully completed and "gracefully failed" (had an empty response or an error message) – and the number of successfully completed requests. Each row within the charts corresponds to a different request complexity: The top was 5 resources (35 RDF triples) per request, the middle, 20 (125 RDF triples), and the bottom was 50 (305 RDF triples).

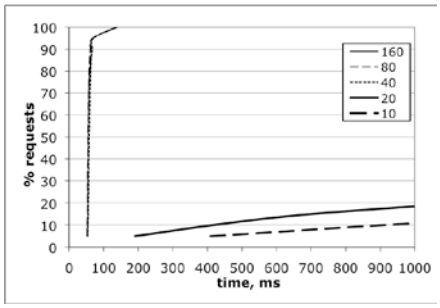
As we can see from Figure 23 (a-1, a-2) – low request complexity – for almost all loads, 95 percent of the requests finished in about 25 ms. Only the peak load of 10 ms between requests slowed the personalization service engine considerably, with the 95th percentile being off the chart at 4,150 ms. This resulted in (3,840-3,488)/3,840 (9 percent) of the requests returning error messages.

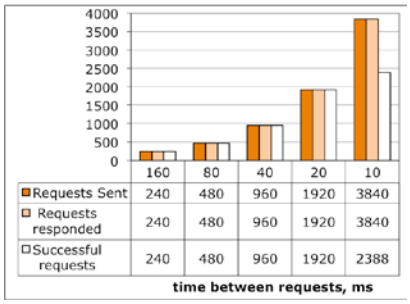
In the case of medium request complexity, (Figure 23 b-1, b-2) there were two peak loads of 10 and 20 ms between requests that resulted in deteriorated personalization service engine performance. The 95th percentiles for them were both off the chart at 12,530 and 6,300 ms for 10 and 20 ms rates, respectively. While all other loads had 95 percent of their requests finishing in about 50 ms, around 38 percent ([3840-2388]/3840) of requests at 10 ms resulted in errors. On the other hand, despite large delays, all requests under the 20 ms load completed successfully.



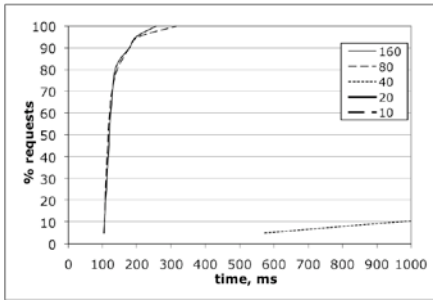


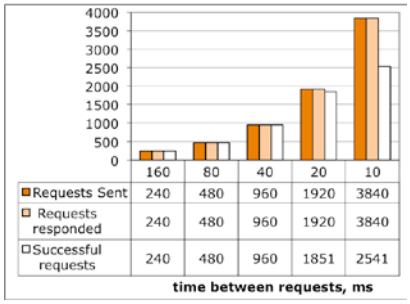
(a-1) (a-2)





(b-1) (b-2)





(c-1) (c-3)

Figure 23. Percentile plots for three request complexities (left column) and request success barcharts (right column) for 5 resources per request (top row), 20 resources per request (middle row), and 50 resources per request (bottom row) across all request delivery rates

In the case of high request complexity (Figure 23 c-1, c-2) three loads of 10, 20 and 40 ms between requests worsened the personalization service engine performance. The 95th percentiles for them were well off the chart at 24,230, 16,000 and 6,750 ms, for 10, 20, and 40 ms rates, respectively. For all other loads, 95 percent of the requests finished in about 200 ms.

Also, instead of going almost vertically until the 95th percentile (for low and medium request complexity), curves bent in the direction of delay increment. About 34 percent ([3,840-2,541]/3,840) of the requests at a 10 ms load resulted in errors. Only 4 percent ([1,920 - 1,851]/1,920) of requests at 20 ms load returned errors. All other loads (even 40 ms load) did not result in errors.

Let us move from a discussion of delays and percentiles to focusing on how many students could be effectively served by the personalization service engine. We based our estimation on user activity observed during the proof-of-the-concept testing of the social navigation service mentioned in Section 3.6.1. As previously stated, there were 37 students. Whenever they worked with the personalization services, they spent 95 percent of their time, collectively, making no more than 7 requests per minute. Since the typical request contained roughly 20 resources that needed personalization, we based the estimation on the results for the medium complexity requests obtained above. Loads of 10 and 20 ms between requests were clearly too high. A load of 40 ms between requests seemed to be quite plausible for a personalization service engine to handle. A rate of 40 ms between requests is 60,000 / 40 = 1,500 per minute. In a class of 37 students, 95 percentof the time had a maximum 7 requests per minute, whenever they were working with the system. To reach the allowed request maximum of 1,500, (37*1500 / 7) ≈ 8,000 students have to be actively involved.

It is important to mention that PERSEUS was not tested alone. It was intensively querying the user modeling server CUMULATE. Hence, one of the important evaluation characteristics for us was what fraction of the total time needed for requests to complete do they "spend" in PERSEUS (not including waiting for CUMULATE's response), in CUMULATE, and in the network (total time minus time spent on PERSEUS and CUMULATE). As it turned out,

the split between the three was roughly equal, without significant advantage of one or the other across all request complexities and delivery rates (see Figure 24).

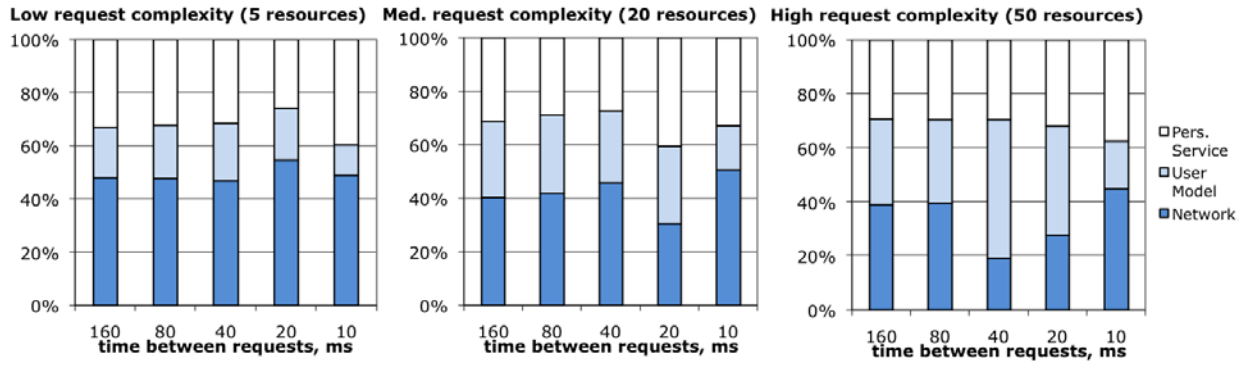


Figure 24. Request time distribution: network communication, CUMULATE and PERSEUS

3.6.4 Pre-study Limitations

These results were quite promising. However, there are several questions this preliminary work did not answer. First, despite such a good performance, it is important to understand that during these tests (Yudelson & Brusilovsky, 2008) the system provided a computationally superficial adaptation – social navigation. This type of navigation support requires a mere matching of per-resource user and group progress (obtained from the user model) to the actual resource links. Other types of adaptive navigation support that we worked with prior to PERSEUS and found very effective, such as topic-based navigation (Sosnovsky & Brusilovsky, 2005) and concept-based navigation (Sosnovsky et al., 2008; Yudelson & Brusilovsky, 2005), are more computationally elaborate than social navigation. The complexity of the respective reasoning engines is higher than that of social navigation. To ensure PERSEUS is capable of effectively supporting adaptation methods and reasoning engines of various computational challenge properly, one has to perform additional evaluation.

Second, the measured results are not aligned against human perception of acceptable delays. Although 95 percent of requests finishing 20, 50, and 200 milliseconds for low, medium, and high request complexities seem to ensure seamless adaptation, these delays were obtained for a low computational challenge social navigation support. If a more complex adaptation method is evaluated, we must align obtained performance estimates to the established acceptable response delays from the literature.

Early work on human perception of the system response delay gives us a 2-second interval during which the natural flow of the interaction is rendered uninterrupted (Shneiderman, 1984). Later work in this direction has refined the interval and produced several levels of delay tolerance. Up to 0.1 seconds, the user feels that the system response is instantaneous; up to 1.0 seconds of system response delay, the user's flow of thought stays uninterrupted; up to 10 seconds' delay keeps the user engaged with the system (Nielsen, 1993). While 0.1 seconds' delay is the desired one, 1-2 second delay seems to be the one we should take as the largest tolerable.

Third, the pre-study focused on PERSEUS's implementation of social navigation support, without establishing a performance baseline for the tool adaptation that PERSEUS is going to compete with. We are going to choose the concept-based adaptation implemented in NavEx (Yudelson & Brusilovsky, 2005) as such a baseline. Concept-based adaptation has been known as one of the most precise and effective adaptation techniques. The encapsulated version of concept-based adaptation implemented in NavEx has been a model for the one implemented in PERSEUS. This makes the choice of both the technique and the implementations to be the perfect candidates for the baseline comparison.

Last but not least, the analysis of the past user activity has been quite shallow. More detailed information on temporal characteristics of the user sessions (session length, click frequencies, etc.) needs to be obtained to get a more realistic picture. The duration of the experiment itself might need to be adjusted and likely would need to be increased. A duration of 3840 milliseconds is a quite short period of time, putting the experiment into the peak-rest test category rather than the capacity planning test category (Maccaux, 2008).

3.6.5 Pre-Study Summary

The pre-study was an important first step in the evaluation of PERSEUS. PERSEUS's social navigation support adaptation functionality was successfully tested in a real environment. It has shown that it is generally feasible to encapsulate an adaptation technique into a standalone server and offer it as a service. Performance-wise, the pre-study demonstrated that PERSEUS could cope with relatively high loads and serve thousands of users.

To meet the goals of this dissertation, we need to extend our evaluation beyond just one adaptation technique and consider several of them. The conceptual possibility to implement a representative subset of known adaptation methods should be carefully examined that are widely used in modern adaptive hypermedia systems and range in posed computational challenge. To properly test PERSEUS's performance, more computationally challenging adaptation procedures should be included in the evaluation. And finally, implementation of adaptation based on PERSEUS should be compared to traditional ways of realizing adaptation, e.g., an intermediary approach.

4.0 RESEARCH DESIGN

In this chapter we will discuss the main experiment of this work. This experiment is an extension of our previous work on performance evaluation of AHS components (Zadorozhny et al., 2008)and (Yudelson & Brusilovsky, 2008). As before, we are going to subject tested systems to various loads and collect performance data. The target of the experiment is to empirically prove the feasibility of implementing adaptation in a standalone fashion and to outline its performance capacity.

The chapter starts with a general overview of the experimental series comprising the experiment, including overall design and metrics used. Then, the setup of the experiments series is discussed, including implementation of the tested systems and relevant configuration data. Next, the actual hardware and the software used for the experiments are discussed. Finally, comprehensive hypotheses about the experiments' outcomes are presented.

4.1 EXPERIMENT OVERVIEW

The main experiment consists of several smaller-scale experiments. In each of the smaller experiments we are using three separate computers hosting three major components (Figure 25): adaptation provider, user modeling server, and the so-called load generator. As in our prior work (Yudelson & Brusilovsky, 2008; Yudelson et al., 2007), the load generator (or the flooder) is

responsible for simulating user activity by sending requests to the tested adaptation provider with some frequency(ies). All experiment components intensively log every stage of the requests originated from the load generator.

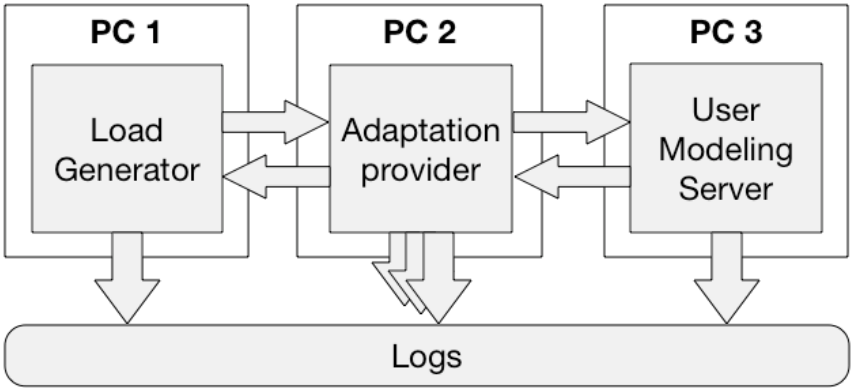


Figure 25. General schema of the experiments

The experiments are divided into two parts. The first consists of the capacity/soak performance tests of the 3 PERSEUS adaptation services (social, topic-based, and concept-based) plus an encapsulated implementation of the concept-based adaptation in NavEx as a baseline. Here adaptation providers are subjected to a set of loads described by the following characteristics:

- **Request complexity** – number of resources to provide adaptation for in each request. The more resources there are in each request, the higher the load. This value is varied across different experiments.
- **Request delivery rate** – delay between consecutive requests. The smaller the delay between requests, the higher the load. This value is varied across different experiments.
- **Duration of the load** – a period of time during which the load generator sends requests to the adaptation provider (PERSEUS or NavEx). Optimal duration is determined by analyzing invocation patterns of the PERSEUS adaptation services or their analogs implemented in an

encapsulated manner. This value is held constant for each of the 3 types of adaptation provided (social, topic-based, and concept-based).

The number of experiments for each implementation of the adaptation is thus a multiple of the number of request complexities and the number of request delivery rates used (duration remains unchanged).

Observed log data is viewed from the following parameters.

- **Response delay** – the amount of time it takes to complete each request. As an aggregate measure, we are interested in the 95th percentile of the response delay for each adaptation provider and load configuration.
- **Request success** – denotes whether the request was completed successfully or not. This involves whether the request was responded to at all and whether the response was correct. As an aggregate measure, success rate (percent requests successful) is used. In addition to the request being answered and the response matching the expected value, a supplementary condition is added – whether the response delay is within the boundaries of human-perceived latency tolerance.
- **Size of the supported user population** – this estimate is based on the summary of the previous three across all load configurations and is computed based on the highest load a tested adaptation provider can handle while demonstrating acceptable performance.
- **Number of requests in the system** – this is a measure derived from timestamps of request start and finish. It shows the amount of requests still in processing. Number of requests in the system over time, in addition to the response delay and request success, shows how capable the adaptation provider is in completing requests under different loads.

Number of requests in the system, although, a secondary measure of performance, requires additional attention. Our intuition is that the load is not constant during each experimental series and is likely to be distributed in the manner depicted in Figure 26. Here we see that in the initial ramp-up period system temporarily gets overwhelmed – number of requests in the processing grows with time and the request-processing rate is low. Then the system request processing rate grows and the number of requests in the system drops by the end of the experiment.

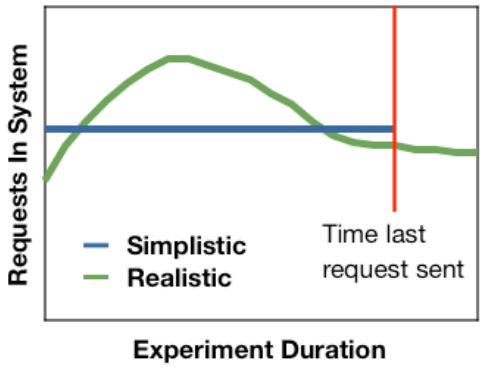


Figure 26. Number of requests processed during one of experimental series: simplistic vs. realistic (assumption).

Several heuristic features can be used to determine whether the behavior of the requests-in-the-system curve corresponds to acceptable performance. The first of them is the time of the last request completed less the time of the last request sent. The smaller this value, the better performance is. Second, the size and the extent of the ramp-up bell-shaped curve (see Figure 26), especially its position with respect to the time the last request is sent, can tell a great deal about the performance of the adaptation provider in a particular situation. If the curve drops soon, or at least before the last request is sent, then the adaptation provider has successfully coped with the load and the saturation point corresponds to a stable performance.

The results of the first part of the experiments are the performance characteristics for 4 adaptation implementations (3 PERSEUS: social, topic-based, and concept-based, and NavEx

concept-based) in the form of the highest handled load. In addition to determining the peak loads that adaptation providers can successfully cope with, we also need to compare PERSEUS's concept-based adaptation service to the baseline encapsulated NavEx's implementation. While the first part of the experiments is enough to draw that comparison, the second is aimed at explaining the differences in observed performance between the two.

In the first part of the experiments, the load generator produces requests that are supposed to be handled by the adaptation providers in a parallel fashion. It is unlikely that in one of the experimental series at some point all of the requests would be pending reply. More likely is the case when earlier requests would be gradually answered, as the load generator sends new requests. Thus at some point of the experiment, some of the requests would already be completed, some would be still pending being sent, and some (more than one) would still be being processed.

The second part of the experiments is devoted to the detailed comparative study of concept-based adaptation in PERSEUS and NavEx only. As opposed to the first part, requests are not intended for parallel processing; instead, the next request is sent only when a previous one is responded to. Sequential generation of requests is selected to avoid the influence of computational resource (CPU cycles, memory, network bandwidth) on different stages of the request lifecycle.

In the second part of the experiments, we are performing a benchmark test. Here we are not interested in the external characteristics of the adaptation providers' performance, namely, overall response delay, request success, and the derived values – number of requests in the system and supported user population size. Our goal is to study the phases of the internal request processing in PERSEUS or NavEx. Load intensity (delay between requests) is no longer

applicable here. Request size is selected based on the results of the first part of the experiments. Duration of the load is transformed into the fixed total number of requests sent. We are not recording request success and only log durations of the principal stages of request processing. These stages include the following durations.

- File input/output (FIO) – this value is the sum of durations of all file input/output operations performed during request processing.
- Data input and initialization (DAT) is a sum of durations of operations that are dedicated to parsing request parameters and creating the internal object models to be used in producing the adaptation value (not counting file input/output operations).
- Querying the user model (UMK, UMP), comprised of two request durations: One asks for the user's conceptual knowledge of concepts, the other for the user's progress with resources.
- Uploading user model values (UPL) to the internal object module of adaptation.
- Production of the adaptation decision (PRO), including the serialization of the result.

Table 2. Summary of the experiments

	PERSEUS			NavEx
Adaptation technique	Social navigation support	Topic-based navigation support	Concept-based navigation support	Concept-based navigation support
Experiment series				
Capacity-planning/soak tests. Parallel request processing	X	X	X	X
Benchmark tests. Sequential processing			X	X

Table 2 has a summary of the experiment series. In the next section we are going to discuss the setup of the experiments in more detail. Namely, we are going to cover load configurations, what is logged by each component, and the data pertaining to hyperspace and user model structure. Also, we will discuss special versions of PERSEUS and NavEx to be used in the experiments.

4.2 EXPERIMENTAL SYSTEMS SETUP

In this section we will discuss the setup of the experiment in concrete numbers. First, data collection procedures are discussed. Then an experimental version (altered from the production version) of each of the tested systems is discussed.

4.2.1 Data Collection and Implementation

The systems that we are going test – NavEx, CUMULATE, and PERSEUS – were not designed solely for the experiments. NavEx has been used in various classroom studies since 2004;see for example (Brusilovsky et al., 2004). CUMULATE in its current implementation was used as a primary user modeling server since 2005 (Brusilovsky et al., 2005). PERSEUS was deployed to be used in class as early as Fall 2007 (Yudelson & Brusilovsky, 2008).

For the experiments, we changed the original programming code of the production systems. That was done in order to factor out some of the overhead processing not related to the adaptation process and as a result to make the performance testing less biased and the comparison of the PERSEUS and NavEx systems fairer. Some of the changes were implemented ubiquitously across all systems involved. Others were specific to particular realizations of

adaptation techniques. In this section we provide a detailed account for all the changes made in the code.

4.2.1.1 Data Collection (Logging)

In PERSEUS, CUMULATE, and the previous version of the load generator (Yudelson et al., 2007; Zadorozhny et al., 2008), logging was originally done in the database. Every transaction was saved as a row in a special table. During the normal mode of operation that does not seem to be problematic. However, during load testing the database communication overhead becomes a tangible factor influencing the performance of the system. This is why the logging mechanism was changed for simple tab-separated files. The structure of the logs remained unchanged: A row is a transaction and a tab-separated column is a field. NavEx had no logging implemented originally. Tab-separated file logging, identical to that of PERSEUS, was added.

Logging procedures in PERSEUS's concept-based adaptation service and NavEx were different in the first and second parts of the experiments. In the second part of the experiments, both systems, in addition to logging request data, were recording the time it takes to perform particular operations (file i/o, data initialization, making the adaptation decision, etc.). Table 3 is a summary of all data logged by all systems involved in the experiment.

Besides logging, several other aspects of the systems' code were changed for the experiments to make the comparison unbiased. Changes to each of the systems are discussed below

Table 3. Summary of logged experimental data

System	Experi-ments	Logged data
Load Generator	all	request ID*, start timestamp†, finish timestamp, request duration, response size, response status¶ (6 fields)
PERSEUS	1	request ID, start timestamp, finish timestamp, request duration; user model query for concept knowledge start, finish and duration; user model query for resource progress start, finish and duration; response status (11 fields)
NavEx	1	same as above
CUMULATE	all	request ID, request type (concept knowledge or resource progress)._start timestamp, finish timestamp, request duration, response status (6 fields)
PERSEUS	2	request ID and durations of… request, file input/output, data input and initialization, user model query for concept knowledge, user model query for resource progress, uploading user model values, production of the adaptation decision (8 fields)
NavEx	2	same as above

* request ID encodes load parameters (delay between requests and request size), name of the tested systems and adaptation technique, and a unique number of the request in the experiment.

† start and finish timestamps correspond to the span of the request lifecycle in the target system. Load generator creates requests, while other systems register events of requests received/sent.

¶ response status – OK, if success, error stack trace otherwise.

4.2.1.2 PERSEUS

PERSEUS is the system that was changed the most. The most significant change is related to adaptation model initialization. In the production version of PERSEUS, the snippet of the hyperspace needing adaptation is passed by a URL reference. Using this reference, PERSEUS retrieves the link structure of the hyperspace snippet via an additional HTTP request to the content manager (portal). NavEx, in contrast, has the hyperspace structure cached locally and does not make additional HTTP calls. To make the comparison of PERSEUS and NavEx more fair and to avoid the manifestation of a C10K problem (Kegel, 2010), the structure of hyperspace has been cached locally for PERSEUS and made accessible in the form of RDF files – the format identical to the one used by the production version of PERSEUS.

In the case of the social navigation support and concept-based navigation support services, the cache files contain the link structures of the referred hyperspace snippets requiring adaptation. In the case of the topic-based navigation support service, where the hyperspace snippet is a set of folders containing resources, additional files with each representing structures of every folder are cached. Also, as mentioned above, database request logging was substituted with file-based logging to reduce computational overhead.

4.2.1.3 NavEx

NavEx has been altered in several ways. The original version, upon every invocation of NavEx, sent an acknowledgement to CUMULATE that the interface had been loaded; code related to this report has been purged. NavEx was a state-full system that kept user data in session variables as the user was working with the system. User data was cleared when a session timed out. In the experimental version, NavEx was made stateless to be more comparable with PERSEUS: No

user information was kept in session variables between requests. As was mentioned above, file-based logging was added to NavEx. Also, the order of operations was changed to mirror the implementation of concept-based adaptation in PERSEUS. The output HTML code was simplified for compactness.

4.2.1.4 Load Generator

The load generator is virtually identical to the one used in our previous work on performance evaluation and the pre-study (Yudelson & Brusilovsky, 2008; Yudelson et al., 2007; Zadorozhny et al., 2008), with the exception of the change from database logging to file-based logging. The only change in CUMULATE used for the experiments is file-based request logging.

4.2.2 Data Model and Configuration

There are many factors that influence the operation of the adaptive applications. In our experiments, we are going to control some of the factors and vary the others. In this section we present the concrete parameters of the data models used in the experiments and for the data analysis and the configuration settings for the applications. We will consider the following parameters and settings:

- Size of the hyperspace (number of resources AKA nodes or links).
- Complexities of the request to the tested adaptation providers in terms of number of resources to be adapted.
- Number of metadata items per resource.
- Duration of user session of working with adaptive tools.
- Frequency of user access to adaptation.

- Frequency polling of adaptation providers in terms of delays between consecutive requests in the capacity-planning/soak test experiment – parallel request processing.
- Number of requests in the baseline comparison benchmark test – sequential request processing.

For all of the above we will provide justification using our prior work on performance evaluation and data collected during various prior studies of AEHS. For some of the parameters, we will also indicate the working point – the value with the highest expectancy of occurrence.

4.2.2.1 Size of the Hyperspace and Complexity of Request

Out of many complete courses that are offered via Knowledge Tree, we selected 4 that were used the most and designed most thoroughly. Our estimation was broken down into two parts (see Table 4). It is worth mentioning that in the same course several adaptation services were offered. For example, Introduction to Programming had topic-based navigation support deployed on the folder level, while social navigation support was offered inside folders.

The first part is related to within-folder navigation methods: social and concept-based. The size of the hyperspace here, with the exception of one course, is close to 200 resources. The size of the hyperspace does not mean the total number of resource links aggregated by the course, but the number of links serviceable by selected adaptation techniques. There are roughly 20 folders per course and on average about 7-10 resources in each (27 being the maximum).

The second part of the summary is related to the between-folder adaptation technique – topic-based navigation. In the two courses where topic-based navigation is deployed, the size of the hyperspace is 100 resources. Here the number of resources considered for adaptation is almost twice as small as for the concept-based and social navigation support. The number of

folders ranges approximately from 10 to 20 and the number of resources per folder is roughly from 5 to 10 (15 being the maximum).

Table 4. Structure of the 4 selected courses with respect to different adaptation techniques

Course	Total no. resources	No. folders	Resources in folders	
			Maximum	Mean±STD
	Social and concept-based navigation support			
Intro to Programming (C)	203	24	20	8.46±5.36
Database design	191	19	27	10.05±7.66
Intro to Programming (Java)	185	25	20	7.40±4.60
Interactive systems design	100	13	16	7.69±5.33
	Topic-based navigation support			
Database design	100	11	15	9.09±4.18
Intro to Programming (Java)	102	21	6	4.86±0.85

Given the data, in the case of within-folder navigation support methods (social and concept-based), the expected size of the request is 10 resources. Since the standard deviation of the number of resources in a folder is quite high and the maximum number of resources in folders often reaches 20 or even goes beyond, we are changing it to 20 resources per request – a less likely, but highly probable case. Just as in our previous work, we are adding more values below and above the expected one(s) for a comprehensive picture and we fix the request complexity values at the following levels: 5, 10, 20, 30, and 40 resources per request. The size of the hyperspace here does not matter, since within-folder methods only deal with a portion of it.

In the case of between-folder navigation support (topic-based), the situation is not uniform: Either we have more folders (20) of smaller size (5), or fewer folders (10) of larger size (10). Here, we are going to set our folder size to 10 resources per folder and the expected size of request to 10 folders. After adding values below and above the expected one, we get the following request sizes: 5, 10, 15, and 20 folder-resources per request (10 resources in the folder). This gives us hyperspace sizes of 50, 100, 150, and 200 resources, respectively.

All of the above correspond to the parallel request processing experiments. For the sequential request processing experiments, limited to the concept-based navigation support technique, we are going to use the reduced set of request complexities. Namely, the two expected values: 10 and 20 resources. These two complexities would be enough to reliably establish the distribution of time between various operations pertinent to the procedure of concept-based adaptation.

4.2.2.2 Number of Metadata Items per Resource

Our prior work on AEHS covered mainly three domains: C programming language, Java programming language, and SQL. The four major tools that we were working on providing adaptation for were: WebEx (Brusilovsky, 2001) – annotated examples viewer (covered all three domains); QuizPACK (Brusilovsky & Sosnovsky, 2005) – problem solving support tool for C programming language; SQLKnoT (Brusilovsky et al., 2008) – SQL problem solving tool; and QuizJET (Hsiao et al., 2008) – a problem solving support tool for Java language. Altogether these tools serve 450 learning activities, and the respective domain ontologies combined have 284 leaf concepts. Table 5 provides a summary of concept metadata statistics.

Across all 4 domains the number of metadata concepts is from 10-20 concepts per resource. The median number of concepts across all 450 resources is 11. We will use this value

as a constant for the concept-based navigation support technique. Metadata concepts possess an attribute that is important for concept-based adaptation: the ratio of outcome concepts to prerequisite concepts. In the examined dataset of 54 QuizPACK problems the ratio is 23 percent. Thus out of 11 concepts, 2 will be marked as outcomes. The total number of concepts will be 100. There are 284 concepts in 3 domains, 100 is close to the mean.

Table 5. Concept metadata statistics for 3 domains and 4 AEHS

Domain	No. concepts	Application (content type)	No. resources	Mean concepts/ resources	Median concepts/ resources
C	54	QuizPACK(problems)	171	9.48±3.09	10
		WebEx(examples)	66	14.53±5.86	14
SQL	130	SQL KnoT(problems)	46	11.22±2.48	11
		WebEx(examples)	64	10.63±4.01	10
Java	100	QuizJET(problems)	103	19.46±13.84	13
All	284	All	450	12.84±8.4	11

4.2.2.3 Duration of User Sessions and Frequency of User Accesses to Adaptation

We analyzed the access behavior of 586 active users[9] from 54 courses offered via the Knowledge Tree portal (Brusilovsky et al., 2008). Table 6 and Table 7 provide a summary of user access characteristics pertinent to the 4 adaptation implementations in question.

[9] The number of registered user accounts is 1467, but those users who have contributed at least one transaction are considered for this analysis. The activity of users registered in multiple courses is considered as one activity pattern. Activity of a user relevant to several adaptations is analyzed separately, grouped by adaptation implementation.

Topic-based navigation access patterns are most relaxed. This is quite expected, since topic-based navigation provides an overview of overall course progress. User sessions on average are only about 3 minutes long and the time between invocations of the adaptation functionality is about 47 seconds. Social navigation support is used more intensively. Session length on average is a little over 4.5 minutes, with 11 seconds between user actions leading to the refresh of adaptation cues. Access patterns related to concept-based adaptation differ between NavEx and PERSEUS. NavEx sessions are a little over 4 minutes long on average, and a typical time between user actions is 14 seconds. Concept-based navigation offered by PERSEUS has been used more intensively. Here an average session lasts over 6 minutes and invocations come close to each other – only 8 seconds in between. Taking into account that NavEx was a narrow focus added-value shell intended for C code examples and has not been used as intensively recently, we will utilize access patterns obtained from concept-based navigation support offered by PERSEUS.

Thus, for the experiments we will use the following durations of the simulated loads:

- Social navigation support: 278.97 sec.
- Topic-based navigation support: 182.56 sec.
- Concept-based navigation support: 372.36 sec.

Table 6. User patterns of accessing the adaptation implementations

System (adaptation)	Estimations base			Estimates			
	No. courses	Active (reg-d) users	Inter-actions	Sec. b/w inter-actions	Session length† (sec)	Hrs. b/w sessions	Max users supported*
NavEx (concept-based)	14	108 (222)	5922	14.04	246.35	101.95	1404
PERSEUS (social)	23	280 (626)	32001	10.77	278.97	47.25	1077
PERSEUS (topic-based)	10	160 (359)	6981	46.62	181.56	12.16	4662
PERSEUS (concept-based)	7	40 (260)	5293	7.68	372.36	23.42	768

† When estimating session parameters, 20 minutes was set as the maximum time between consecutive requests. Namely, if 2 requests of the same login sessions were more than 20 minutes apart, the login session was logically split at that point.

* The estimate is based on the assumption that the system successfully copes with the maximum tested load of 10 ms b/w requests. All users are assumed to work concurrently

Table 7. Co-occurrence of user sessions

		Co-occurring session (non-zero)			
System (Adaptation)	No. active (registered) users	Min	Mean	Max	95th percentile
NavEx (Concept-based)	108 (222)	1	1.01	2	1
PERSEUS (Social)	280 (626)	1	1.55	14	5
PERSEUS (Topic-based)	160 (359)	1	1.26	8	3
PERSEUS (Concept-based)	40 (260)	1	1.03	2	1

To estimate the size of the user population, based on the most intensive successfully sustained load, we will use the following intervals between adaptation invocations:

- Social navigation support: 10.77 sec between invocations. This corresponds to 1077 concurrently working users effectively supported if the load of 10 milliseconds between requests is successfully sustained. Taking into account that the 95th percentile of the number of co-occurring sessions for this type of navigation support is 5, the number of effectively supported users could be up to 1077 * (1077/5) ≈ 232 000.
- Topic-based navigation support: 46.62 sec between invocations. This corresponds to 4662 concurrently working users effectively supported if the load of 10 milliseconds between requests is successfully sustained. Taking into account that the 95th percentile session co-occurrence is 3, the number of effectively supported users could be up to 4662 * (4662/3)≈ 7 250 000.
- Concept-based navigation support: 7.68 sec between invocations. This corresponds to 768 concurrently working users effectively supported if the load of 10 milliseconds between requests is successfully sustained. Taking into account that the 95th percentile session co-

occurrence is 1, the number of effectively supported users could be up to 768 * (768/1) 590 000.

4.2.2.4 Hyperspace Structure and User Model Data

Hyperspace structure and user model data is one more important part of the experiment data model. The hyperspace and user model – resulting from the above decision on the hyperspace and request size – consists of 40 resources (for social and concept-based navigation support) or 200 resources (for topic-based navigation support), 100 concepts, and 800 user accounts. Each resource was randomly linked to 11 unique concepts; 2 of them were marked as outcomes.

User log data consisted of 10000 transactions (for social and concept-based navigation support) or 40000 transactions (for topic-based navigation support). Each transaction contained user id, resource id, and interaction outcome. User ids were chosen randomly from 1 to 800 without repetitions. Interaction outcome was randomly chosen to be either 0 (incorrect) or 1 (correct). Randomization of resource ids was more complicated.

Resource ids were assumed to be in the order appropriate for studying, from the easiest (smallest id) to the hardest (largest id). For each of the generated user transactions, resources were chosen randomly from subsequences marked by the lower and upper boundaries. Initial values of lower and upper boundaries were 1 and 5, respectively. When the number of attempted resources/exercises was greater than or equal to 50 percent of the upper boundary, the value of the upper boundary was increased by 5. When the difference between the lower and upper boundaries exceeded 5, the lower boundary was incremented by 5. Separate boundary values were kept for different users along with a list of attempted exercises. Except for the total number of generated transactions and number of resources, procedures were identical.

4.2.2.5 Delays between Requests in Parallel Processing Experiments

Just as in our previous work, we are going to use an exponential scale for the load generator's polling frequencies in parallel request processing experiments. From our prior experience we know that 10 ms between requests is quite a challenging load. We are going to use this value as the upper boundary. The lower load boundary would be 160 ms between requests. This gives us five different values: 10, 20, 40, 80, and 160. We cannot determine now what load, if successfully sustained, is the desirable working point for us. Our prior work used a different experiment length condition and only one computationally simpler adaptation technique – social navigation. Except for "the higher the sustained load, the better," there are no prior expectations we could formulate.

4.2.2.6 Duration of Sequential Processing Experiments

The parallel request processing experiment only deals with concept-based navigation support implemented in PERSEUS and NavEx. Here the length of experiment does not matter. What we need is to detect the distribution of processing time across various stages of the services' operation reliably. We are going to set the limit at 5000 sequential requests. This number would definitely allow us to collect the desired pattern data.

4.2.2.7 Experiments' Summary

Figure 27 and Figure 28 provide a condensed view of the parallel and sequential request processing experiments. The parallel request processing experiment (Figure 27) consists of 25 series for social navigation support technique, 25+25 series for concept-based navigation support techniques run against both PERSEUS and NavEx, and 20 series for topic-based navigation

support technique. The sequential request processing experiment Figure 28 consists of 2+2 series for concept-based adaptation only run against PERSEUS and NavEx. Overall this gives us 25+25+25+20+4=99 experimental series.

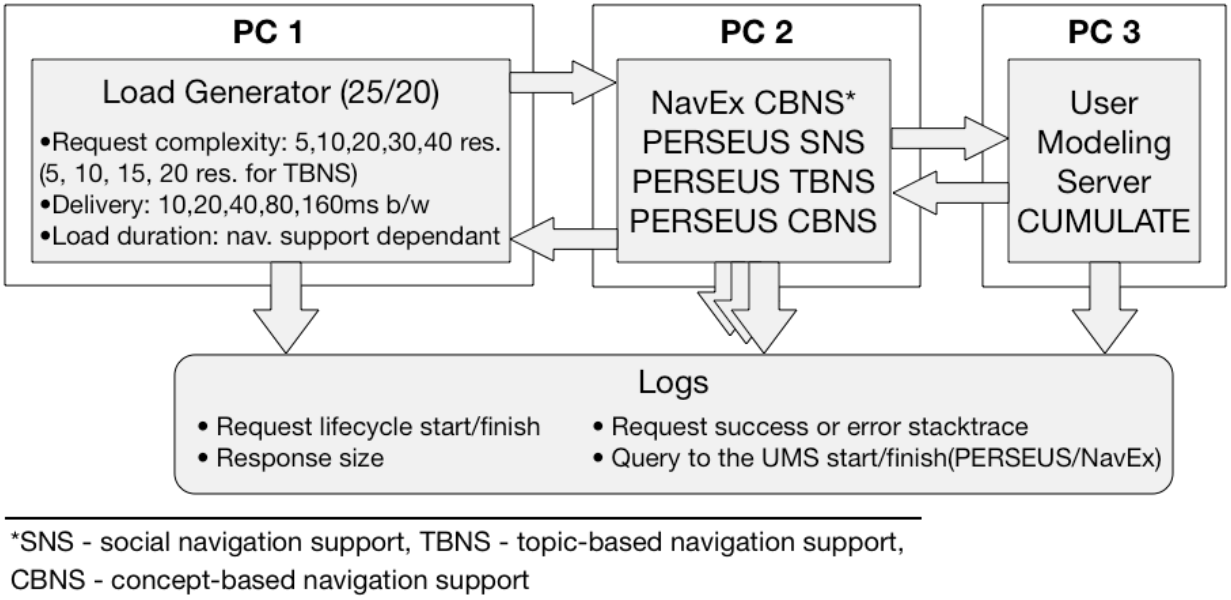


Figure 27. Summary of the parallel request processing experiments

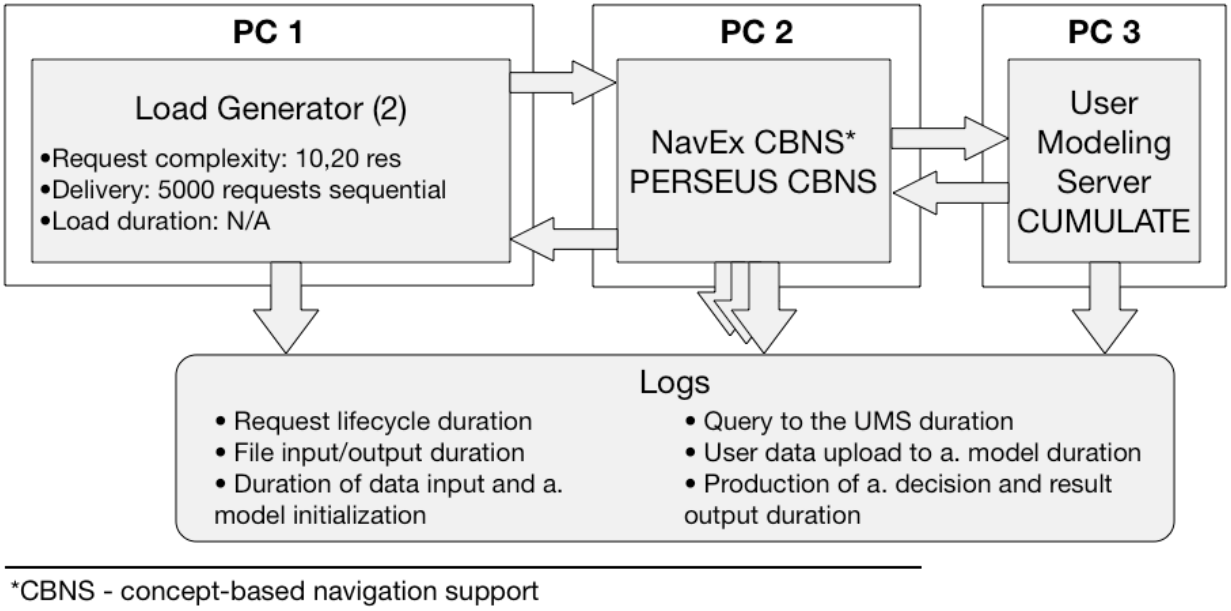


Figure 28. Summary of the sequential requests processing experiments

4.3 HARDWARE AND SOFTWARE

In our earlier work on performance evaluation (Yudelson & Brusilovsky, 2008; Yudelson et al., 2007; Zadorozhny et al., 2008) we used several physical machines connected by a wired/wireless network. While that is a more ecological approach, it loosens the control on the factors that are secondary to our research: hardware variability and inconsistency, network delays unrelated to the experiment, etc. In this study we are choosing a method that has become plausible very recently – using virtual computers connected by a virtual network.

The host system accommodating virtual machines is a Mac Pro (model #MacPro1,1) with two Quad-Core Intel Xeon CPUs running at 2.26 GHz, 8GB (4x2GB) DDR3 1066 MHz RAM, and 600 GB SATA HDD, running Mac OS X 10.5.8.

Guest systems were run using Sun Microsystems VirtualBox version 3.1.8.r61349. The three guest systems were all running 64 bit Ubuntu Linux 9.10 and each was configured to have 1 dedicated CPU Core, 2GB RAM, 8GB HDD, and PCnet-FAST III network adapter. All applications (Load Generator, PERSEUS, NavEx, and CUMULATE) were implemented as Java Servlets and were run in Tomcat 6.0.20. PERSEUS, NavEx, and CUMULATE used MySQL version 5.1.37 database server via MySQL Java Connector version 5.1.12. Guest machines were connected by a dedicated internal network (intent) with static IP assignment and could not communicate to other machines on the network except to each other.

The host machine was restarted prior to each of the 99 experimental series. Startup of the experimental guest machines and the virtual network was done via shell script. Configurable shell script managed the operation of the load generator. Every guest machine had a dedicated shell script for saving and archiving the experimental data and performing system cleaning after each of the experimental series.

4.4 COMPREHENSIVE HYPOTHESES

4.4.1 Parallel Processing Capacity-planning/Soak Tests

As our capacity-planning test conditions have significantly changed from our previous test of PERSEUS (Yudelson & Brusilovsky, 2008) with regard to the duration of the experimental series, we should expect to see worse performance, i.e., a lower load intensity would be the highest tolerable. From the other hand, the use of virtualization would be able to counter-balance this to some extent. Nevertheless, at least for the social navigation support that we studied in the pre-study, the highest tolerable load should be lower across all request complexities.

Hypothesis 1. Due to more demanding conditions, the highest tolerable load for the social navigation support would be at least one step of frequency metric lower (between-request delay higher) than the one reported in the pre-study.

For example, the pre-study reports 40 ms between requests as the highest successfully coped-with load for request complexity of 20 resources; in the main experiment we expect to see 80 ms between requests as the highest successfully tolerated load.

Out of 3 tested adaptation techniques – PERSEUS social, PERSEUS topic-based, and PERSEUS and NavEx concept-based – the least computationally intensive we think is the social navigation support. The most computationally intensive is most likely topic-based navigation support, due to the fact that it operates on larger numbers of resources: 50 to 200 versus from 5 to 40 in the case of social and concept-based. The computational complexity of concept-based navigation support implemented in NavEx or PERSEUS is expected to be in between.

Hypothesis 2. We expect to see social navigation support (as the least computationally intensive technique) to be able to cope with higher loads. Concept-based navigation is expected to successfully tolerate lower loads. Topic-based navigation support is assumed to

successfully withstand even lower loads. This partial order is likely to manifest itself at all request complexities.

In the two prior hypotheses we are making comparative claims about performance at different request-generating frequencies (inter-request delays) and for different request complexities. By the design of our experiment, the lowest values of request complexities and request-generating frequencies (higher values of inter-request delay) are likely to be comparably easy to achieve for all adaptation techniques. Similarly, the highest values of these parameters would be comparably challenging. We envision that the relative differences mentioned in Hypothesis 1 and Hypothesis 2 are going to be more pronounced in the mid-ranges of load parameters.

Hypothesis 3. We are likely to see less performance differentiation at the extreme values of request complexity (5 and 40 requests per request) and inter-request delay (10 ms and 160 ms between requests) and more likely to see stratification in the mid-ranges of the factors.

4.4.2 Baseline Comparison Sequential Processing Benchmark Tests

In general, when comparing a general-purpose system like PERSEUS in our case and a specialized system like NavEx, the advantage is more likely to be with the specialized one. The main reason is that generality is often achieved by making compromises.

Table 8. Differences in processing time of NavEx and PERSEUS concept-based adaptation

Operation †	Difference	Description
File Input/Output	No	NavEx and PERSEUS have a comparable number of file I/O operations. Detecting difference here is unlikely. *
Data input and initialization (w/o File I/O)	Yes	PERSEUS relies on RDF as a major format for data input-output. Necessity to parse RDF XML puts it into a disadvantaged position. At this type of activity PERSEUS is likely to lose to NavEx in terms of processing time.
Querying user model	Excluded	Querying user model for resource progress and concept knowledge is related to user modeling server performance and network; hence, it is excluded from the comparisons.
Uploading user model values	No	Here, both systems merge internal formats of acquired user model data with internal formats of adaptation models. We expect to see no difference in processing time.
Production of the adaptation decision and the result	Yes	PERSEUS is expected to be trailing due to necessity to perform a time-consuming RDF XML serialization. NavEx has to just print out concise HTML code.

†The enumeration and descriptions of the principal operations in NavEx and PERSEUS implementations of the concept-based navigation support are provided in Section 4.1

*NavEx's File I/O is different from PERSEUS in the part where the structure of the hyperspace is acquired. PERSEUS reconstructs it from local RDF files, NavEx from the database. Although database access is slower than reading a file, necessity to parse RDF is more likely to counterbalance that.

We are comparing concept-based navigation support implemented almost identically in NavEx and PERSEUS. The difference is that NavEx is an adaptive intermediary intended to provide adaptation for a set collection of educational resources and PERSEUS is a general-purpose adaptation provider targeted for a virtually unlimited number of domains and contexts of use. Table 8 presents a comparative summary of principal operations of the concept-based adaptation in NavEx and PERSEUS and the likelihood of detecting a difference between them in a baseline comparison benchmark test.

As we can see from Table 8, PERSEUS is likely to be disadvantaged as compared to NavEx in 2 out of 4 principal computational elements of the concept-based adaptation (the 5th being excluded from the comparison).

Hypothesis 4. Concept-based navigation support implemented in PERSEUS (a general purpose adaptation provider) is going to be more computationally demanding than the implementation in NavEx (specialized adaptation provider). This disadvantage is going to be detected in terms of increased processing time for 2 out of 4 processing stages of the adaptation during baseline comparison benchmark tests (second part of the experiment). In addition we are likely to see PERSEUS successfully tolerate lower loads during capacity-planning-soak tests (first part of the experiments) in terms of request complexity and/or request frequency.

Despite expected differences in computational time, we believe that the highest tolerated load will not be drastically lower for PERSEUS as compared to NavEx.

Hypothesis 5. Differences between highest tolerated loads of PERSEUS and NavEx implementations of concept-based adaptive navigation support are not going to be larger than one level of the load-characterizing values – request complexity and inter-request delay.

4.4.3 Size of the Supported User Population

Our pre-study (Yudelson & Brusilovsky, 2008) concluded that a desktop-grade machine running PERSEUS's social navigation support service could successfully provide social navigation support for up to 8000 actively working students. The pre-study was based on more relaxed load duration settings, and PERSEUS and CUMULATE are no longer co-located on one machine. This is why we expect to see a smaller effectively supported user population.

Table 6 already has mentioned the maximum number of supported users for the top request frequencies (10 ms between requests). For social navigation support this number is 1077 users, for topic-based navigation support it is 4662 users, and for concept-based navigation support it is 768 users. Since neither of the adaptation techniques is probably going to be able to tolerate the maximum request frequency, we expect smaller user populations to be supported. Since the size of the supported user population is tightly connected to the tolerated request frequency, these two measures are practically interchangeable.

Hypothesis 6. Our expectations regarding the number of users effectively supported by each adaptation technique are the following. PERSEUS's social navigation support: 270 users (40 ms between requests, 1077/4 = 270). PERSEUS's concept-based navigation support: 96 users (80 ms between requests, 768/8=96). NavEx's concept-based navigation support: 192 users (40 ms between requests, 768/4=192). PERSEUS's topic-based navigation support: 292 users (160 ms between requests, 4662/16=293). All of the estimates are given for typical/"working point" request complexities: 10-20 resources for social and concept-based and 10 resource folders for topic-based navigation support.

Given the lengths of typical user sessions and intensity of their work (see Section 4.2.2), the hardware settings for our machines (see Section 4.3), and the computational challenges that the adaptation methods pose, confirming these expectations would be a desirable result.

4.4.4 Number of Requests in the System

We expect that heuristic features of the requests-in-the-system curve – size and extent of the ramp-up bell-shaped portion and time between last request sent and last request completed – will mirror the results obtained during capacity-planning tests. In other words, "good" curves would correspond to successfully tolerated loads and "bad" curves would match

Hypothesis 7. We expect the empirical characteristics of the requests-in-the-systems curve to confirm our conclusions about load tolerance for most of the load configurations and adaptation techniques tested. Namely, the maximum tolerated load would correspond to the maximum load with acceptable requests-in-the-systems curve characteristics.

5.0 RESULTS

This chapter presents the analysis of the collected experimental data. First, the result of the parallel processing capacity-planning experiments is covered. Sequential processing benchmark tests are discussed next. Sizes of the supported user populations for each of the tested systems and test loads are presented third. Fourth, the number of requests in the system metric is considered. Finally, a general discussion of the dissertation results, implications, and limitations is covered.

5.1 PARALLEL PROCESSING CAPACITY-PLANNING/SOAK TESTS

The results of the parallel processing tests are shown inFigure 29, Figure 30, Figure 31, and Figure 32. They correspond to performance characteristics of PERSEUS's social navigation support, PERSEUS's concept-based navigation support, NavEx's concept-based navigation support, and PERSEUS's topic-based navigation support, respectively.

Each figure has 5 (4 in the case of Figure 32 for topic-based navigation support) graphs each corresponding to different request complexities (AKA request sizes). These graphs are percentile plots where the horizontal axis denotes time taken to complete requests and the vertical axis denotes percent of the requests complete. On each graph there are several curves, each corresponding to a different load condition (measured in milliseconds between requests).

These graphs offer a limited view on the timeline of the request completion. The horizontal axis is capped at 2 seconds – the maximum response delay – according to (Nielsen, 1993) – that would not interrupt the user's flow of thought when working with a system. Although (Shneiderman, 1984) gives a smaller delay of only 1 second, 2 seconds seems like an acceptable compromise.

The general rule for interpreting percentile graph is to look at the point of the 95th percentile and at the general shape of the curve in the graph. A percentile curve that corresponds to an acceptable performance is expected to be oriented approximately vertically. As soon as the time values abruptly grow, the performance degrades. To conclude that the system copes with the load successfully (sometimes the term scalable is used here), the curve is expected to be approximately vertical until the 95th percentile.

In our case, we are adding a 2-second cap on the percentile values. We are not interested in the orientation of the curve that much. Our criterion is that the curve stays within the 2-second boundary as much as possible. For this reason we decided to lower the percentile criterion from 95 to 90 percent. Thus, to conclude that a certain implementation of an adaptation technique successfully copes with a particular load, we require 90 percent of the requests to complete in less than 2 seconds, i.e., the 90th percentile point to be on the graph. Because we are most interested in whether 90th percentile point is within 2 second boundaries, we are only displaying percentiles starting with 80th in Figure 29, Figure 30, Figure 31, and Figure 32.

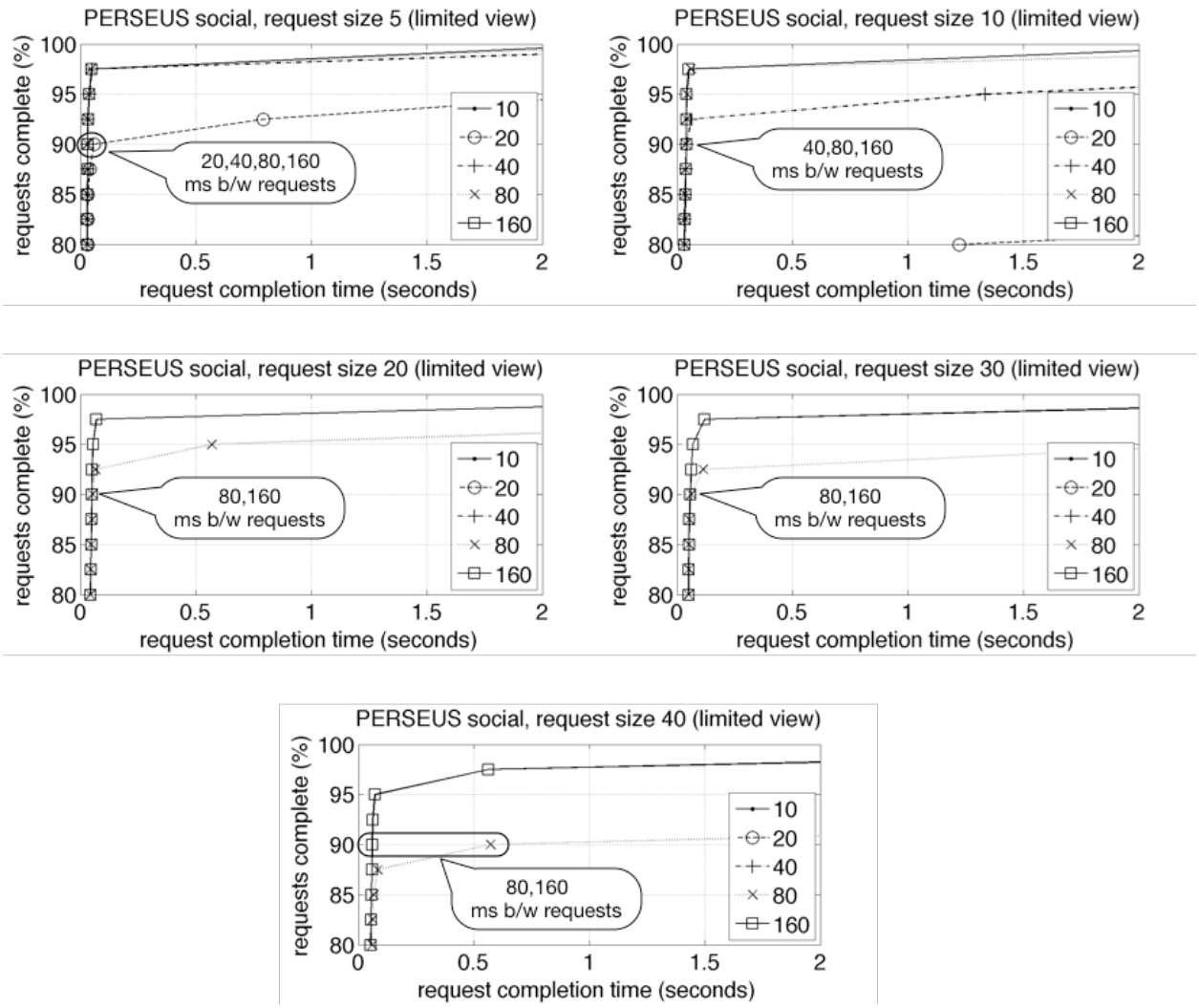


Figure 29. Percentile plots for PERSEUS's social navigation support service capacity-planning/soak tests. Loads meeting the cap criteria are marked by call-outs

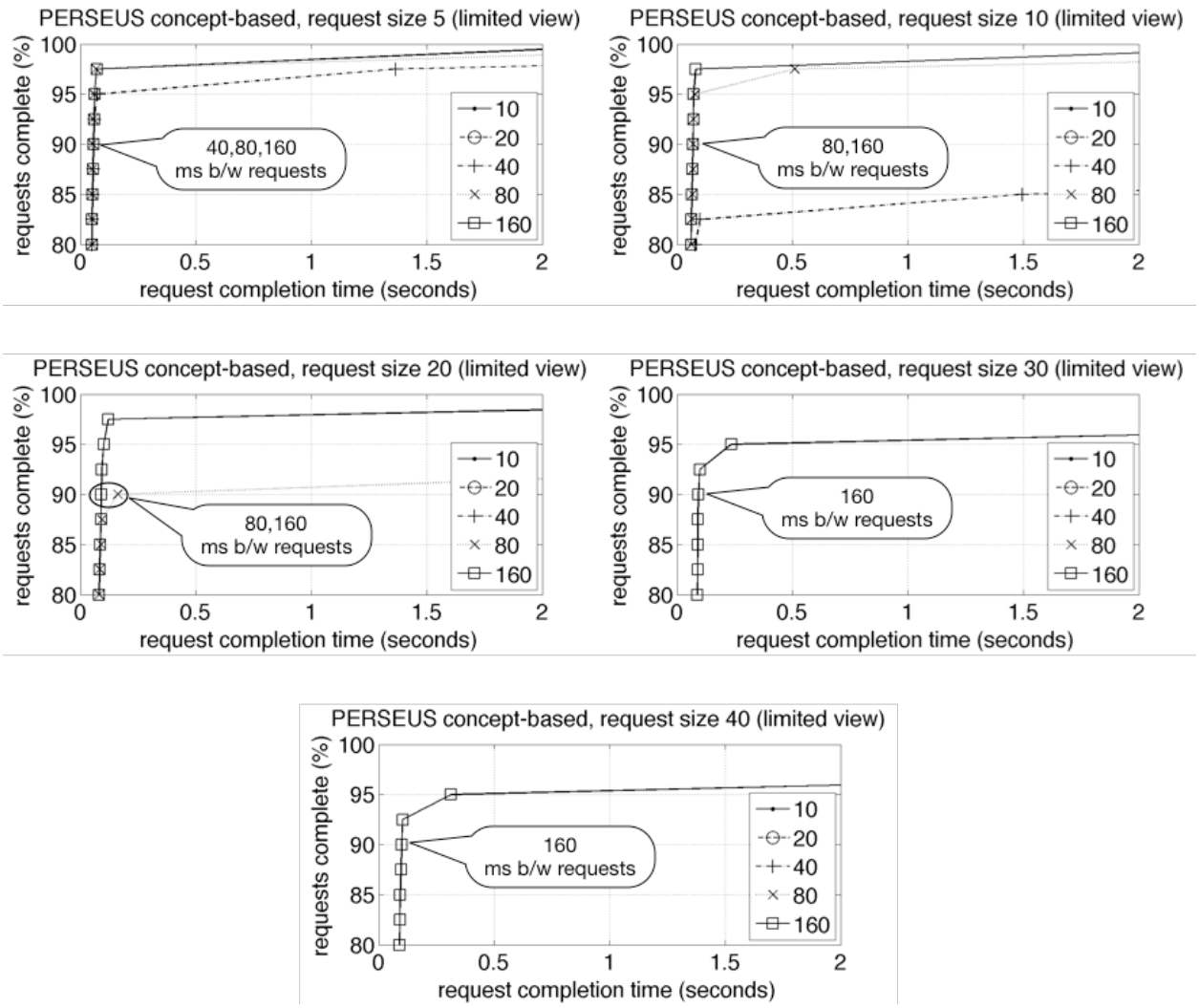


Figure 30. Percentile plots for PERSEUS's concept-based navigation support service capacity-planning/soak tests.

Loads meeting the cap criteria are marked by call-outs

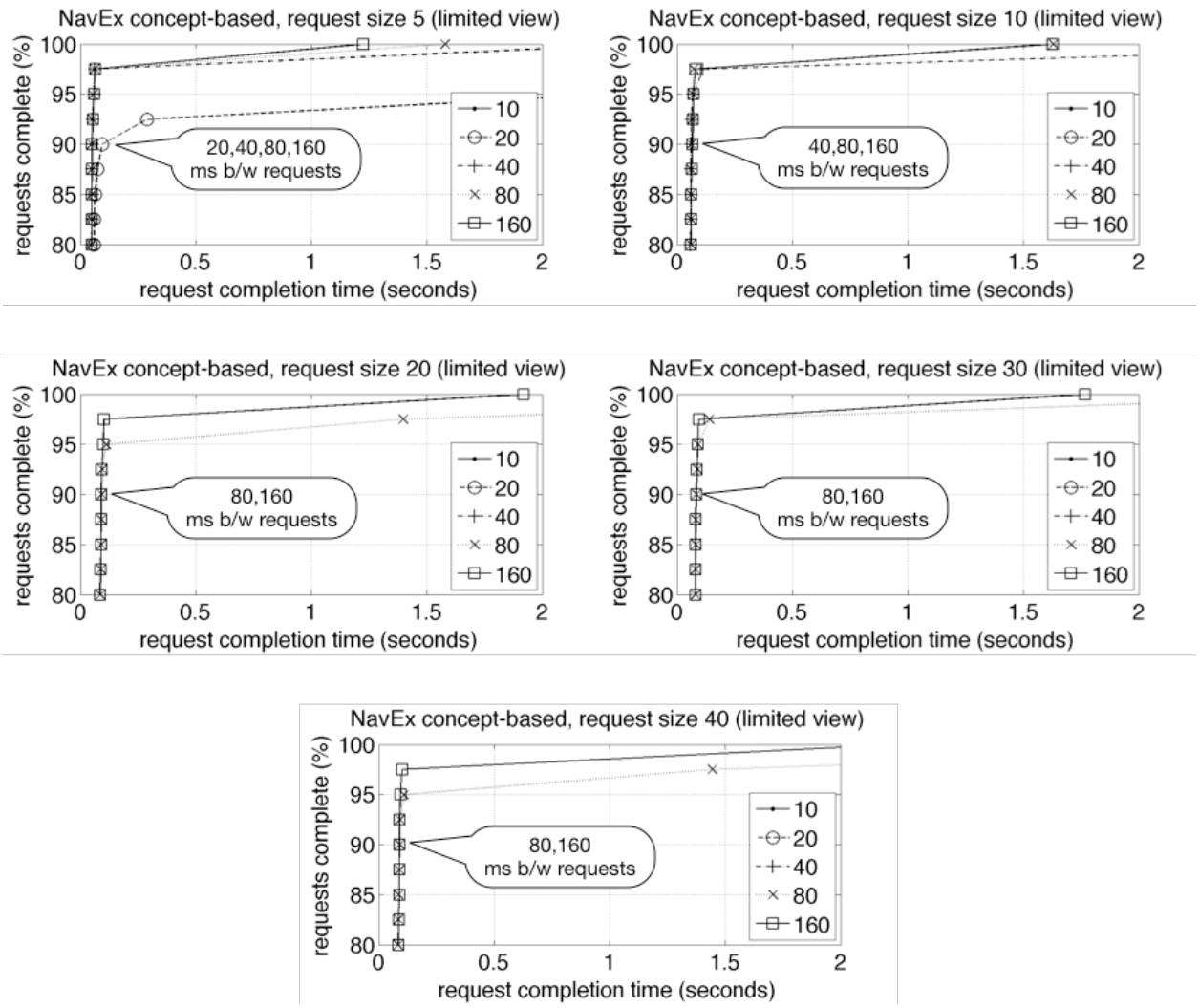


Figure 31. Percentile plots for NavEx's concept-based navigation support service capacity-planning/soak tests.

Loads meeting the cap criteria are marked by call-outs

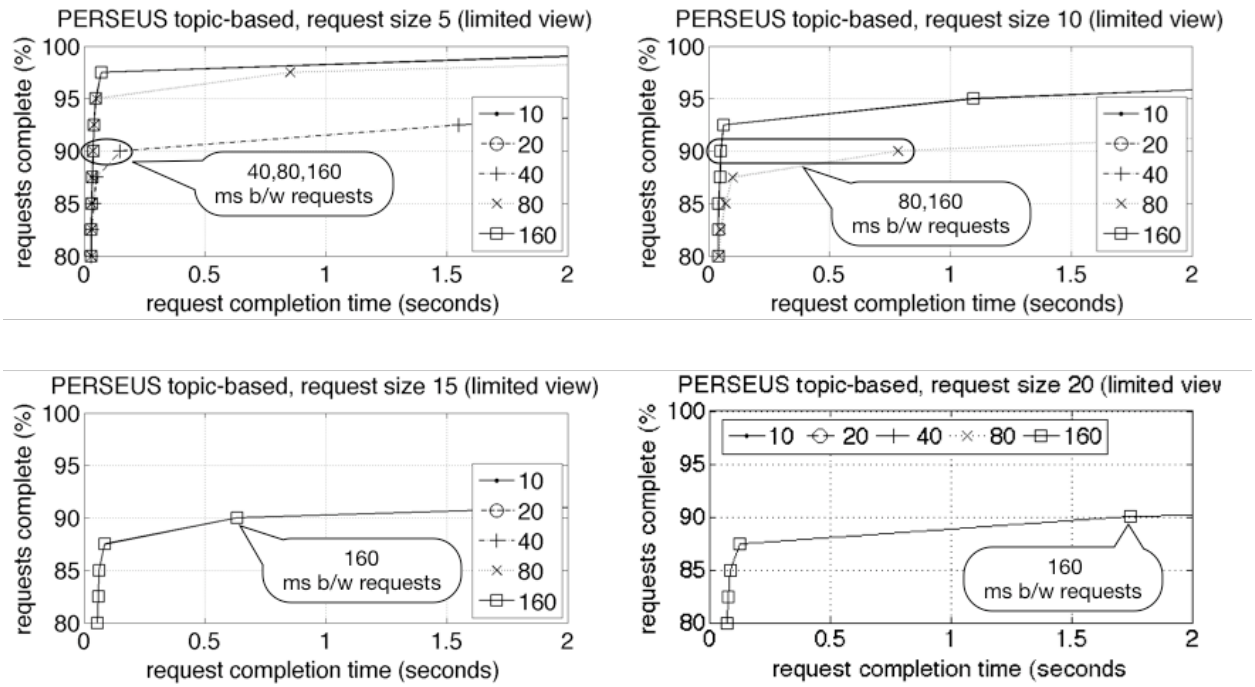


Figure 32. Percentile plots for PERSEUS's topic-based navigation support service capacity-planning/soak tests. Loads meeting the cap criteria are marked by call-outs

Table 9 provides a summary for in Figure 29 through Figure 32. Here for each of the tested adaptation technique implementations and across all request complexities, the highest load that is successfully tolerated is given. Shaded cells correspond to request complexities that are more probable to occur (judging from our experience).

We can see that social navigation support, as expected, copes with performance challenges the best among other PERSEUS-based adaptation methods. It successfully tolerates loads of 40 and 80 ms between requests for 10 and 20 resources per request, respectively. PERSEUS's concept-based navigation support is slightly less scalable. It successfully copes with a load of 80 ms between requests for both 10 and 20 resources per request, while for higher request complexities its performance is visibly worse.

Table 9. Maximal successfully tolerated loads for tested adaptation techniques. Shaded cells mark expected request complexities.

Adaptation technique		Maximal tolerated load, ms between requests				
	Request complexity	5	10	20	30	40
PERSEUS's social		20	40	80	80	80
PERSEUS's concept-based		40	80	80	160	160
NavEx's concept-based		20	40	80	80	80
	Request complexity	5 (50)*	10 (100)	15 (150)	20 (200)	
PERSEUS's topic-based		40	80	160	160	

* Here the first-order object is not a resource, but a folder. The total number of resources in folders is given in brackets.

At first glance, PERSEUS's topic-based navigation support is the least scalable. As request complexity grows, its maximal successfully tolerated load quickly decreases. However, we should bear in mind that, while the nominal request complexities are from 5 to 20 folders, the number of underlying resources is from 50 to 200 and for an expected request complexity of 10 folders (100 resources), maximal successfully coped with load of 80 ms between requests is a high mark to score. Also, since topic-based navigation support is intended for index (folder) nodes of the hyperspace that are accessed less frequently, the numbers of supported users are actually the highest among all tested adaptation techniques.

NavEx's implementation of concept-based navigation support, as expected, wins by one step of the used load scale over PERSEUS's implementation of the same technique and is

identical to the characteristics of PERSEUS's social navigation support technique. The reasons for this not very large advantage are discussed in Section 5.2.

An important component of our evaluation is not only quick responses to requests for adaptation, but also the validity of the responses. Figure 33 provides a summary of the number of erroneous responses to adaptation requests. Four sub-graphs correspond to four different techniques tested. Each sub-graph visualizes the number of errors across load intensities (delays between requests) and request sizes. Shaded areas correspond to the most expected request sizes and the load intensities that were successfully tolerated (as summarized in Table 9).

PERSEUS's social navigation support

Delay b/w requests (ms)	Request Size (resources) 5	10	20	30	40
10	0	0	0	76	0
20	0	0	0	0	0
40	0	0	0	0	0
80	0	0	0	0	0
160	0	0	0	0	0

PERSEUS's concep-based navigation support

Delay b/w requests (ms)	Request Size (resources) 5	10	20	30	40
10	10	0	1	140	414
20	0	0	87	45	112
40	0	0	0	2	21
80	0	0	0	0	0
160	0	0	0	0	0

PERSEUS's topic-based navigation support

Delay b/w requests (ms)	Request Size (resources) 5	10	15	20
10	0	0	0	0
20	0	0	0	0
40	0	0	0	0
80	0	0	0	0
160	0	0	0	0

NavEx's concep-based navigation support

Delay b/w requests (ms)	Request Size (resources) 5	10	20	30	40
10	0	0	0	0	0
20	0	0	0	0	0
40	0	0	0	0	0
80	0	0	0	0	0
160	0	0	0	0	0

Figure 33. Summary of the errors for all adaptation techniques across request sizes and loads (delays between requests). Shaded regions correspond to expected request complexities and load characteristics up to maximal successfully tolerated ones. Only errors for requests with below 90th perentile delay are considered.

As it can be seen in Figure 33, PERSEUS's social navigation support and topic-based support and NavEx's concept-based adaptation support are predominantly error-free until the 90th percentile of the request delays – our cap. PERSEUS's concept-based adaptation support

has a fair amount of errors, especially for tougher loads and more complex requests. However, all of the targeted request complexities and successfully tolerated loads are error-free (until the 90th percentile of the request delays).

5.2 SEQUENTIAL PROCESSING BENCHMARK TESTS

One of the goals of this work is to compare encapsulated adaptation to one of the traditional forms of providing adaptation. For that purpose we chose one of the most popular and at the same time one of the most computationally challenging adaptation techniques – concept-based navigation support. In this section we compare the concept-based adaptation in PERSEUS (encapsulated approach) to the concept-based adaptation in NavEx (intermediary approach).

In the previous section, among other things, we saw that performance of PERSEUS's implementation of concept-based adaptation is worse than NavEx's implementation of the same technique. Figure 34 and Figure 35 show the results of sequential processing benchmark tests performed to determine where the difference is. These tests were done with precise logging features turned on both in PERSEUS and NavEx. The principal stages of data processing in concept-based adaptation procedures were: file input-output (FIO), data input and initialization (DAT), uploading user model values into adaptation model (UPL), and production of the adaptation decision and the result (PRO).

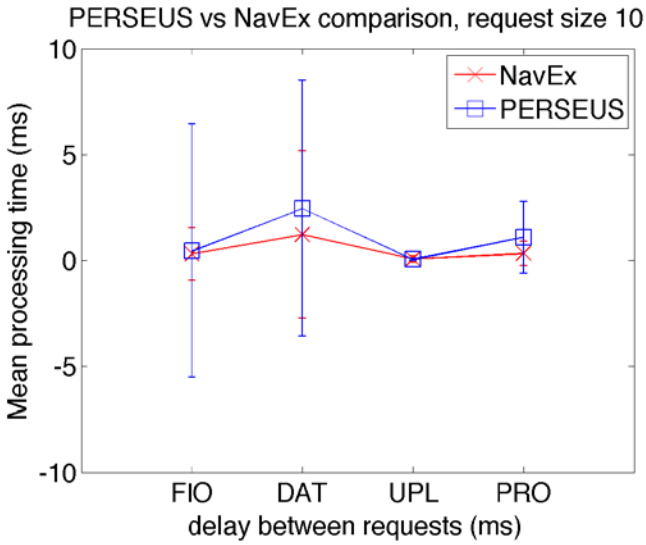


Figure 34. Comparison of concept-based navigation support realization in PERSEUS and NavEx (request size 10)

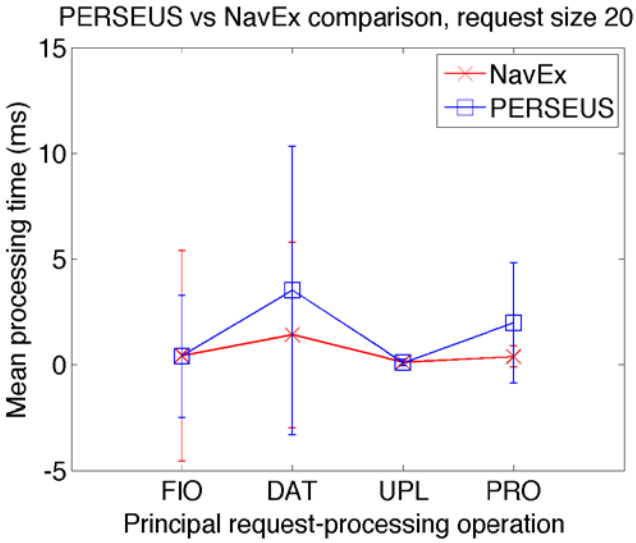


Figure 35. Comparison of concept-based navigation support realization in PERSEUS and NavEx (request size 20)

We can see in Figure 34 and Figure 35 that in tests for both request complexities – 10 and 20 resources per request, respectively – results are quite similar. As expected, file input-output and aggregating user model data into the adaptation model show practically no difference in the mean times. Differences are detected for the data input and adaptation decision and result production stages. Although error bars for these stages overlap in both graphs, the differences of

means are quite large. For example, mean processing time for the data entry stage is 3.97±1.23 (10 resources per request) and 4.38±1.41 (20 resources per request) in the case of NavEx and 6.03±2.46 (10 resources per request) and 6.84±3.52 (20 resources per request) in the case of PERSEUS. The mean values in this case are approximately 50 percent higher. In general, standard errors for per-stage mean processing times are larger in the case of PERSEUS.

5.3 SIZE OF THE SUPPORTED USER POPULATION

In Table 6 the last column lists the maximal number of concurrently working users for each of the tested adaptation techniques. That estimate is based on the respective observed user access patterns and the assumption that each technique would successfully tolerate a load of 10 ms between requests. For the social navigation support, the maximum is 1077 users working simultaneously; for topic-based navigation support, 4662 users; and for concept-based navigation support (usage patterns merged across PERSEUS and NavEx) the estimate is 768 users.

Since in all cases the highest successfully tolerated load is lower (milliseconds between request value is higher), the number of simultaneously working users supported is lower and is obtained by simple division by a factor of 2 (as our request frequencies double at each level of the load). Table 10 is a summary of the supported user population values across all techniques and request complexities. Shaded cells denote results corresponding to the expected values of the request complexities.

As per Table 10, the social navigation support technique in PERSEUS is capable of supporting 135 to 270 users working concurrently, depending on the request complexity. PERSEUS's concept-based navigation support technique can effectively support 96 users, while

the NavEx version of the same techniques can support 96 to 192 users. The topic-based navigation support technique in PERSEUS can effectively support up to 583 users working at the same time.

Table 10. Number of **simultaneously working** users that are effectively supported by tested adaptation technique and request complexity. Shaded cells mark expected request complexities.

Adaptation technique		Size of supporter user population, users				
	Request complexity	5	10	20	30	40
PERSEUS's social		540	270	135	135	135
PERSEUS's concept-based		192	96	96	48	48
NavEx's concept-based		384	192	96	96	96
	Request complexity	5 (50)*	10 (100)	15 (150)	20 (200)	
PERSEUS's topic-based		1166	583	291	291	

* Here the first-order object is not a resource, but a folder. The total number of resources in folders is given in brackets.

Although the sizes of effectively supported user populations reported in Table 10 seem quite small, we should bear in mind that these are only for the users that work with adaptive systems simultaneously. If we factor in the data of user session co-occurrence (reported in Table 7), the total number of supported users is much larger. We are basing our estimates of the total number of effectively supported users for each technique on the 95th percentile of the session co-occurrence data taken from our logs. While obtaining these estimates, only non-zero values were considered for the computation of the 95th percentile. Hence, the estimates are inflated and are more conservative.

For social navigation support the 95th percentile of session co-occurrence is 5, for topic-based navigation support – 3, and for concept-based navigation support (both PERSEUS and NavEx) – 1. To go from the number of concurrently working users to the number of total users supported, we multiplied the former by a factor. The numerator of the factor is the number of concurrently working users, and the denominator is the 95th percentile of concurrent sessions. For example, the number of users simultaneously working with a social navigation support and issuing 10-resource requests is 270. This transfers into 270*270/5 = 14580

Table 11. Size of the supported user population (observing the number of concurrent sessions) for tested adaptation techniques. Shaded cells mark expected request complexities.

Adaptation technique		Size of supporter user population, users				
	Resource complexity	5	10	20	30	40
PERSEUS's social		58104	14580	3645	3645	3645
PERSEUS's concept-based		36864	9216	9216	2304	2304
NavEx's concept-based		147456	36864	9216	9126	9126
	Resource complexity	5 (50)*	10 (100)	15 (150)	20 (200)	
PERSEUS's topic-based		453185	113296	28227	28227	

* Here the first-order object is not a resource, but a folder. The total number of resources in folders is given in brackets.

As a result, the social navigation support technique in PERSEUS is capable of supporting a total of 3645 to 14580, depending on the request complexity. PERSEUS's concept-based navigation support technique can effectively support 9216 users, while the NavEx version of the

same techniques can support 9216 to 36864 users. The topic-based navigation support technique in PERSEUS can effectively support 113296 users in total.

5.4 NUMBER OF REQUESTS IN THE SYSTEM

Number of requests in the system (RIS) is a measure that shows at any given moment how many requests are pending reply. It is our candidate of proxy for the instantaneous measure of the load. Figure 37, Figure 38, Figure 39, and Figure 40 present percent RIS plots for the four tested adaptation technique implementations. The horizontal axis corresponds to time. A vertical red line marks the moment when the last request is sent. The vertical axis is the percentage of the total requests set that currently are pending reply.

When analyzing percent RIS curves, we are going to use two heuristic criteria. First is the time from the moment the last request is sent (which is constant across all experimental series for a particular adaptation technique) to the moment the last request is completed (not necessarily the request that was sent last). The smaller this time, the better the adaptation technique copes with the load. Second is the shape of the curve. As the experiment starts, the RIS number starts to grow. If this RIS number (or in our case percent RIS) stabilizes or drops to a small value (e.g., less than 5 percent), the performance of the adaptation technique is said to be acceptable. If percent RIS starts to decline abruptly at the time the last request is sent (whether it was declining before or not), the performance is rendered as not acceptable. We will call the load corresponding to an acceptable performance a *recoverable load*; the one that created an initial boost that an adaptation technique safely recovered from.

In Figure 36 there are 5 percent RIS curve sub-graphs for PERSEUS's social navigation support adaptation technique, one for each request complexity of 5, 10, 20, 30, and 40 resources per request. In the case of the first two complexities, all of the loads are recoverable. The system successfully copes even with high frequencies of 10 milliseconds between requests. In the case of 20 and 30 resources per request, only loads up to 20 milliseconds between requests are recoverable. Notice that in the graph for 30 resources per request the percent RIS curve corresponding to 10 milliseconds between requests starts to visibly decline only when no more new requests are sent. In the case of 40 resources per request, 40 milliseconds is the highest recoverable load. Although the percent RIS curve for 20 milliseconds between requests starts its decline before the last request is sent, it does not decrease to a value of 5 percent or less until after the last request is sent.

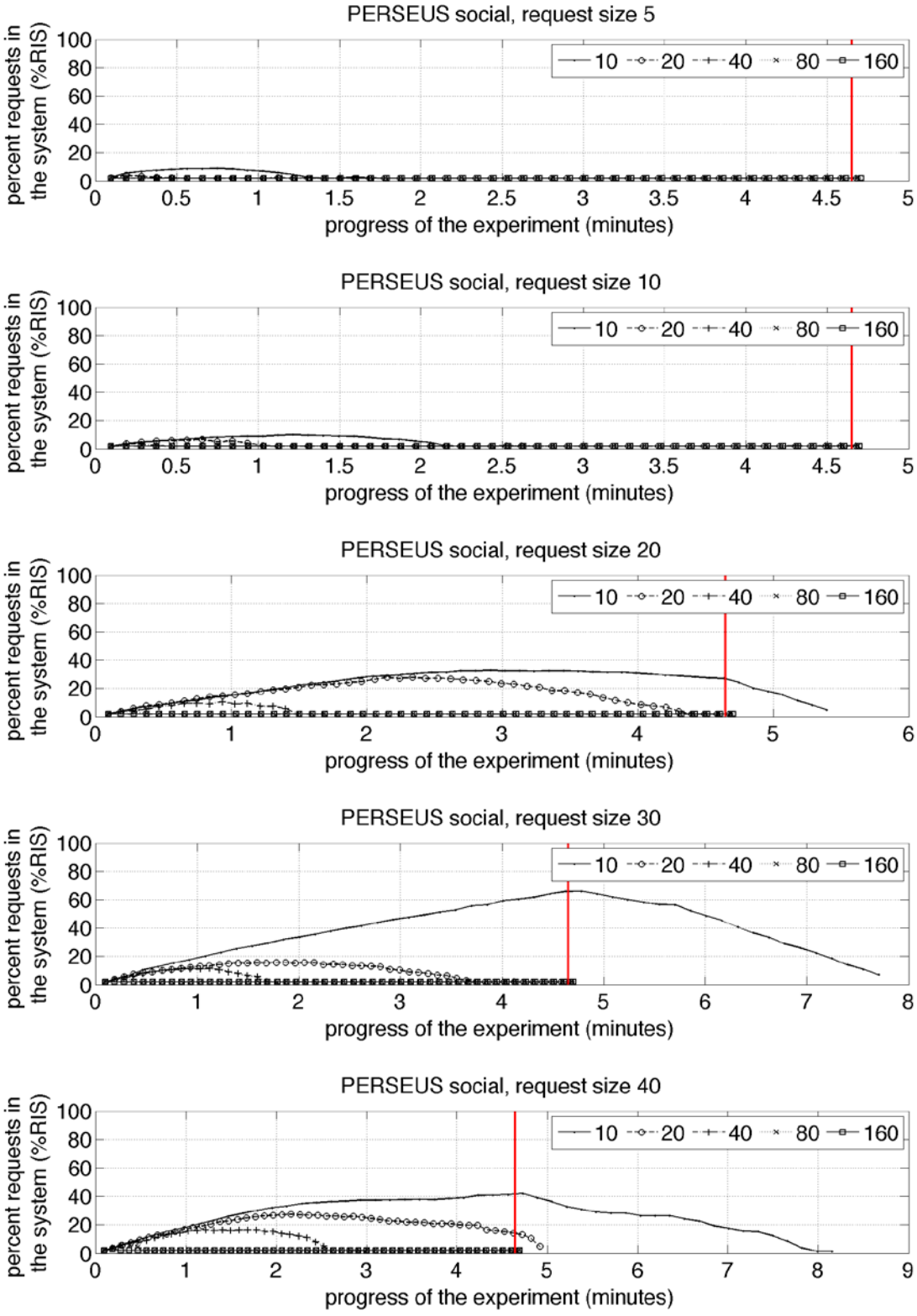


Figure 36. Percent of requests in the system over the time of the experiment for PERSEUS's social navigation support service capacity-planning/soak tests

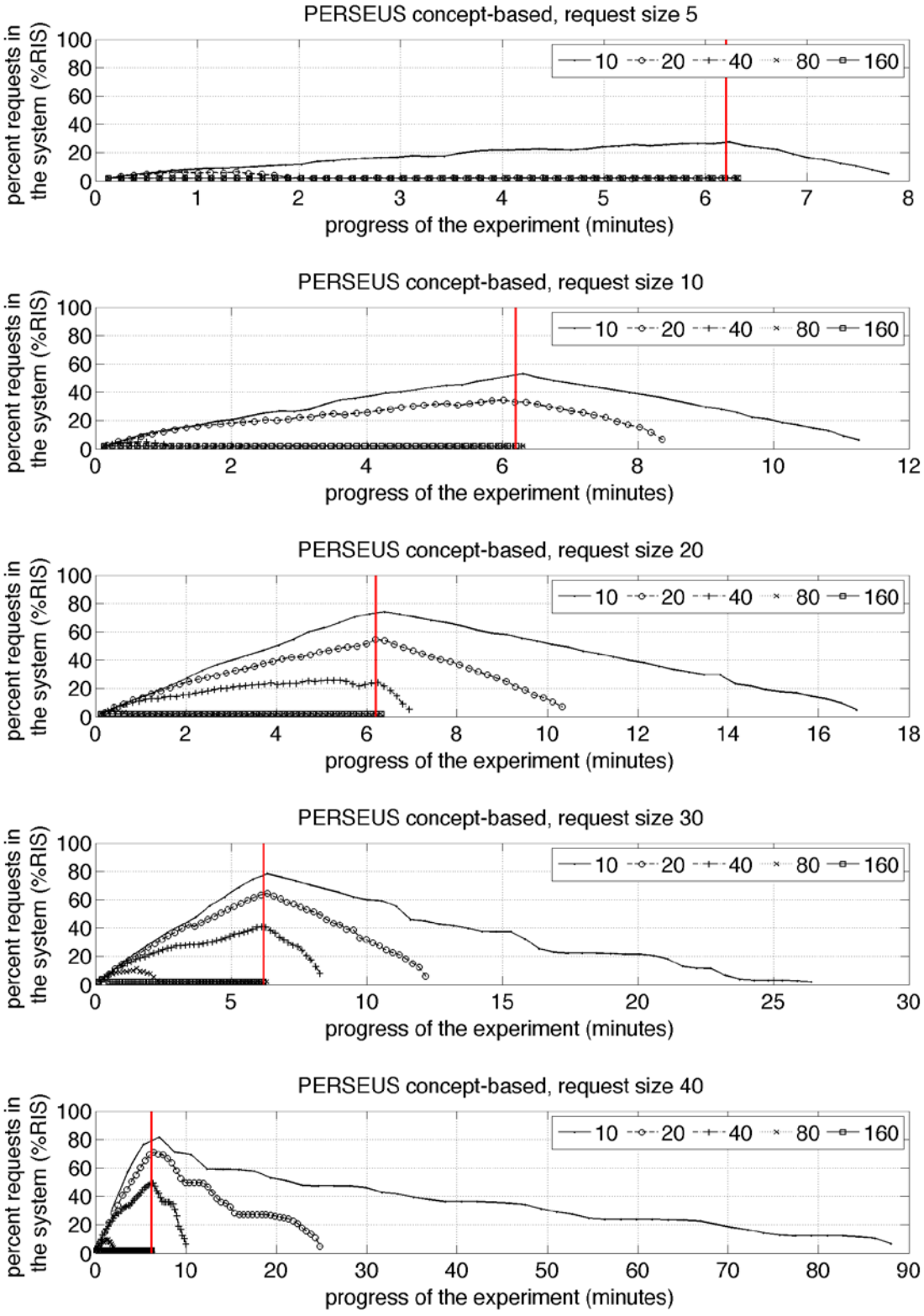


Figure 37. Percent of requests in the system over the time of the experiment for PERSEUS's concept-based navigation support service capacity-planning/soak tests

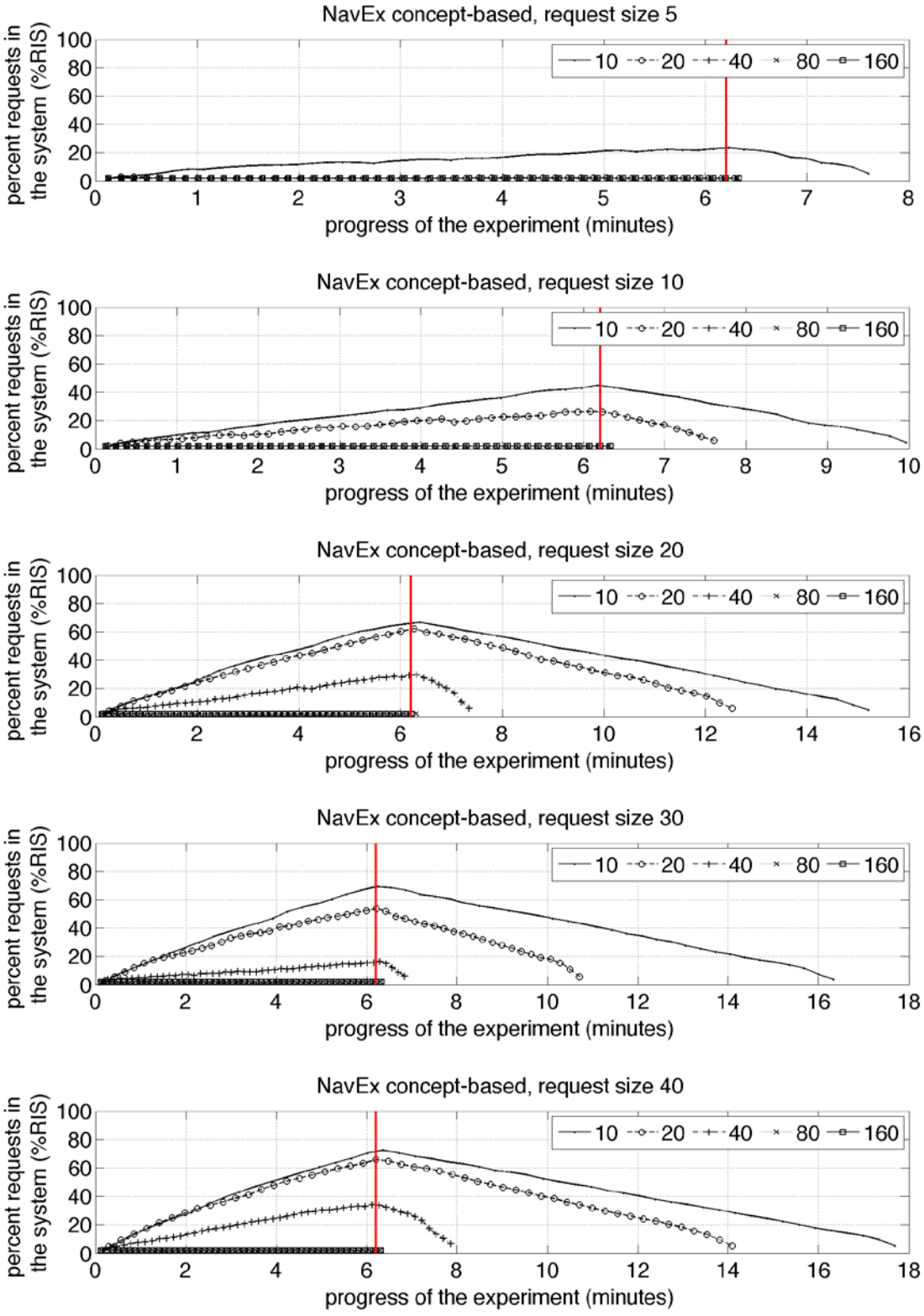


Figure 38. Percent of requests in the system over the time of the experiment for NavEx's concept-based navigation support service capacity-planning/soak tests

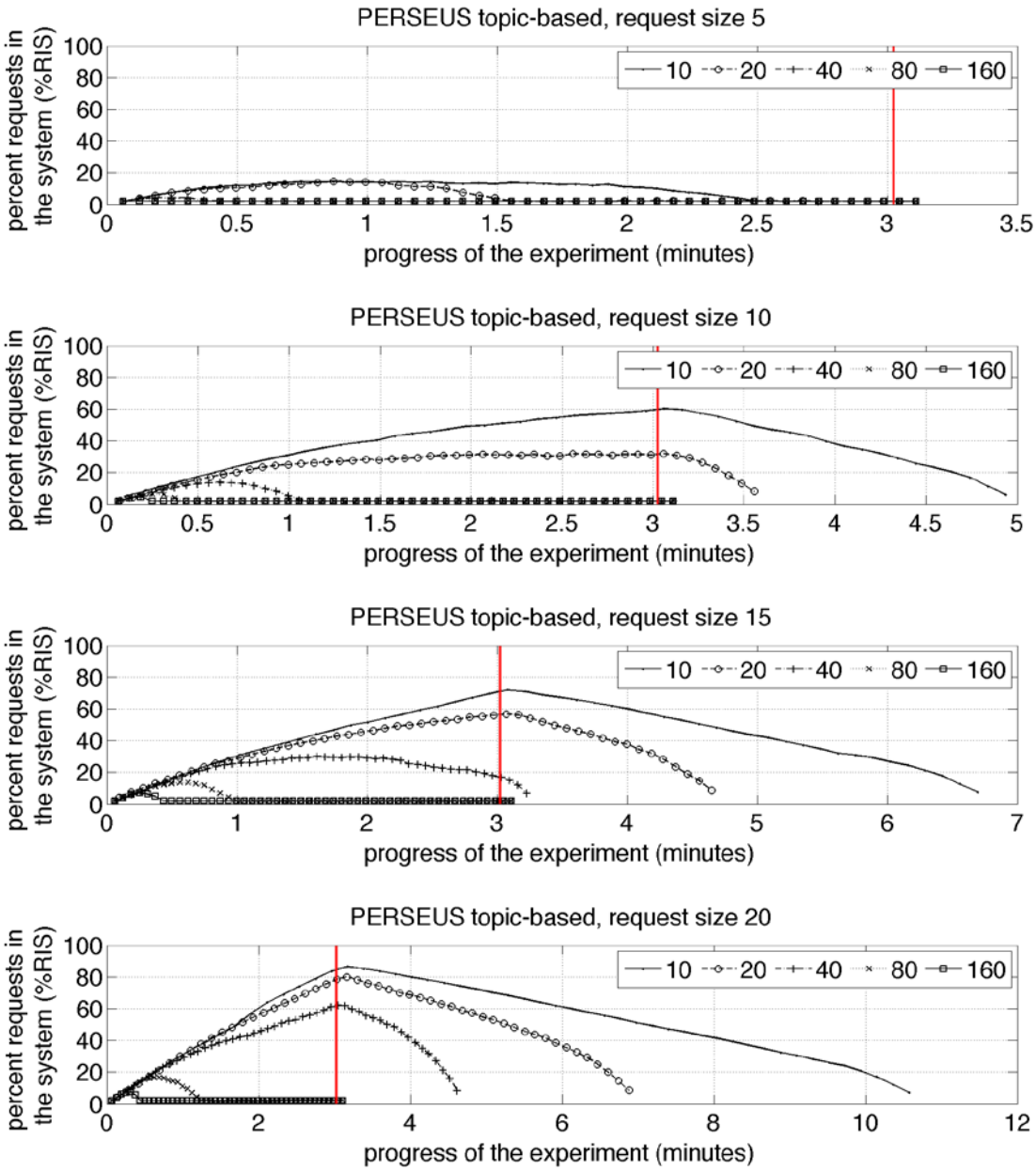


Figure 39. Percent of requests in the system over the time of the experiment for PERSEUS's topic-based navigation support service capacity-planning/soak tests

A summary of the maximal recoverable loads is given in Table 12. This is a version of Table 9 where highest tolerated loads are given. In Table 12, the highest successfully tolerated load is given alongside the highest recoverable.

In many of the cases, the highest recoverable loads confirm the highest tolerable loads. However, in some cases, recoverable loads are higher. This means that, even if the load is not well tolerated and the 90th percentile of request delay does not meet our criteria, the adaptation technique is able to recover from the initial performance challenge successfully.

Table 12. Maximal successfully tolerated loads for tested adaptation techniques vs. maximal recoverable loads. Shaded cells mark expected request complexities.

Adaptation technique		Maximal tolerated load : maximal recoverable load, ms between requests				
	Resource complexity	5	10	20	30	40
PERSEUS's social		20 : 10	40 : 10	80 : 20	80 : 20	80 : 40
PERSEUS's concept-based		40 : 20	80 : 40	80 : 80	160 : 80	160 : 80
NavEx's concept-based		20 : 20	40 : 40	80 : 80	80 : 80	80 : 80
	Resource complexity	5 (50)*	10 (100)	15 (150)	20 (200)	
PERSEUS's topic-based		40 : 10	80 : 40	160 : 80	160 : 80	

* Here the first-order object is not a resource, but a folder. The total number of resources in folders is given in brackets.

Judging from the maximal recoverable loads, we can see that the most scalable technique is PERSEUS's social navigation support. All recoverable loads are higher than tolerated loads. Next is NavEx's concept-based navigation support, where all recoverable loads confirm the tolerated load values. PERSEUS's concept-based navigation support comes after that. It is less scalable in terms of tolerable loads, but recoverable loads follow the pattern of NavEx's

implementation of the technique. PERSEUS's topic-based navigation support is hard to compare to the rest, since it deals with a radically higher number of resources.

5.5 SUMMARY

From the results reported above, we could conclude that PERSEUS has met the experiment goals we set and supported practically all of our prior assumptions. We empirically proved that it is capable of supporting a representative set of adaptation technologies and reasoning engines that are widely used in contemporary AHS and range in computational complexity. The obtained performance estimates demonstrated that adaptation functionalities developed on the basis of a standalone adaptation server could be successfully deployed for a practical use and exhibit acceptable performance characteristics.

The size of the effectively supported user population, in fact, was in some cases larger than we expected. This is another testament to the viability of PERSEUS's design and the quality of its implementation.

The advantage that a traditional intermediary approach to adaptation (special-purpose system) has over PERSEUS's encapsulated approach (general purpose system) is visible yet acceptable. The reasons for this advantage descend from the choice of the data exchange format used by PERSEUS (RDF XML). This format is a de facto interoperability and information-sharing standard for the field of AHS today. Adhering to this widely accepted standard is the major reason for PERSEUS's relative loss in terms of scalability. Changing the data exchange format to a more scalability-friendly one would harm PERSEUS's value as a general-purpose provider of adaptation functionality.

6.0 CONCLUSIONS

In this section we offer closing remarks about the results of the experiments in particular and this dissertation work in general. We start from contributions and significance of this work. Then, we outline limitations. Finally, future work is discussed.

6.1 CONTRIBUTIONS AND SIGNIFICANCE

In this work we are suggesting a solution that would help achieve a wider dissemination of adaptive hypermedia technologies to a larger user population and help offer adaptive access to a larger volume of content. We target AHS designers and aim at making the development of new systems faster. We propose to extract the adaptation functionality and encapsulate it in a standalone system – a server of adaptation functionalities that offers adaptation as a service transparently to non-adaptive client applications.

We designed and built a server of adaptation functionalities called PERSEUS that implemented the idea of the encapsulated adaptation. With PERSEUS, adaptation can be provided like a service to a client content management hypermedia system. The compliance and configuration overhead is relatively smaller compared to traditional approaches to implementing adaptation – built-in adaptation and intermediary solutions.

Conceptual evaluation of PERSEUS's design showed that it can serve as a basis for implementing virtually all known adaptation techniques (Brusilovsky, 2001). PERSEUS has been deployed as a primary adaptation provider for a number of undergraduate and graduate courses since 2008 at the University of Pittsburgh (USA), Dublin City University (Ireland), and Universidad Autónoma de Madrid (Spain).

We performed an extensive evaluation of PERSEUS covering several widely used adaptation engines of various computational complexities (social, topic-based, and concept-based) and addressing one of the most widely used adaptation methods – adaptive link annotation. A total of nearly one hundred experiments were conducted.

The setup of the experiments was based on the analysis of system usage logs collected for a period of 4 to 5 years. Obtained results confirmed our hypotheses about PERSEUS's ability to answer performance challenges and successfully support a few to several hundred users working simultaneously and from thousands to tens of thousands of users in total; all of that was achieved on a rather modest hardware.

The results we report are grounded in nearly a decade of research on building AHS and offering adaptively accessible educational content online. The work reported in this dissertation is the first work to evaluate the idea of extracted adaptation both conceptually and from the viewpoint of performance. This is also the first large-scale performance evaluation of AHS in general and component-based AHS in particular. It is the first effort to compare a novel architectural approach to offering adaptation to the traditional solutions.

6.2 LIMITATIONS

There are several properties to the performed experiments that can potentially limit our ability to generalize the conditions set up for the evaluation performed. These limitations are listed below.

First, we only tested a small set of adaptation techniques, namely those that would help us visualize changes in performance that originate in computational complexity. One might easily come up with techniques that are computationally less demanding or more demanding, or even techniques that in terms of performance lie between the ones we chose. Concrete implementations of conceptually the same techniques can also vary. Thus, the performance estimates that we obtained only correspond to points on a continuous curve of adaptation techniques' computational complexities.

Second, all three tested techniques – social, topic-based, and concept-based – assume that adaptation decision is made (and fresh user data is requested) at every call to PERSEUS. This, of course, should not always be true. For example, an earlier version of the CUMULATE user-modeling server (Brusilovsky et al., 2005) could be set to honor the *freshness* of assumptions about the user parameter, where the user model is only updated if at least a certain amount of time has passed since the last update. An adaptation procedure aware of this feature could significantly limit the amount of network traffic by limiting the number of queries to the user model. One could also imagine an adaptation procedure that uses local cache in order to prevent frequent polling of user model data or any other relevant data that needs to be queried.

Third, the number of experimental factors involved in the experiments was quite large. This is why we chose to control some of them (e.g., size of the metadata vocabulary, number of metadata concepts per resource, etc.) and limit the range of and discretize the values of the others (e.g., number of resources to adapt per request, delay between requests, etc.). Several of these

decisions were made after looking back at our own experience with deploying educational tools in class for several years (e.g., the range of number of resources per request approximated the values occurring in the actual courses that we administered with or without adaptation support from PERSEUS). Others were dictated by the outcomes of our prior work on AHS performance evaluation (Yudelson & Brusilovsky, 2008; Yudelson et al., 2007; Zadorozhny et al., 2008).

Both the controlled and discretized values for the experiment factors do limit our ability to reason about performance of the subject techniques. For example, the value of the size of supported user population for each technique would change if the number of metadata items per educational resource would change (either as a constant value or as a distribution of values). Nevertheless, this does not prevent us from making principal claims about performance characteristics of PERSEUS and adaptation techniques implemented on its basis. This work is the first one that addresses the question of adaptation functionality performance across several adaptation techniques. The subjectivism of the experiments setup is fully justified by the fact that there was literally no related work on the subject except for our own.

Fourth, the hardware we used for performance evaluation is quite modest (single dedicated 2.26 GHz CPU core and 2 GB of RAM). Additionally, it was run via virtualization software. If a more powerful machine is available (more RAM, faster than virtual non-shared HDD), performance characteristics are very likely to improve visibly.

6.3 DISCUSSION

In this work we are primarily focusing on using a standalone adaptation engine (such as PERSEUS) in an educational context. We are assuming that PERSEUS is a client of an

educational portal and provides adaptation and personalization for links to educational resources. However, the conceptual design of PERSEUS allows it to be used in any context, where personalization is needed. In Figure 14, where a typical personalization scenario of PERSEUS's operation is shown, client portal (or content-management system) could be serving content from virtually any domain, and the back-end user model could work not only with knowledge models, but also preferences, interests, etc.

Examples of alternative domains, where PERSEUS functionality could be consumed, might include healthcare. A health-related portal, where resources are articles intended for patients with various conditions, could benefit from a link sorting and annotation hybrid service that would help people access relevant information. PERSEUS would need to be able to access targeted patient conditions, either from patient data (external data store) or from a query string. Both types of information here can be treated as substitutes for the student model in the case of educational domain. Of course, additional precautions should be made to adhere to HIPAA[10] standards on patient data privacy.

Another field, where PERSEUS could potentially find its use is on-demand adaptation and personalization in an industrial context. Google Site Search[11] and Amazon's A9 search engine[12] are well-known examples of outsourcing offering custom search functionality on a large-scale basis.However, a closer analog of PERSEUS would be the internal Amazon's product recommendation API or Google's internal similarity algorithms (that they recently made available to a limited population of beta testers under the name of Google Prediction API[13]).

[10] HIPAA – Health Insurance Portability and Accountability Act, http://www.hhs.gov/ocr/privacy/

[11] http://www.google.com/sitesearch/

[12] http://a9.com

[13] http://code.google.com/apis/predict/

PERSEUS, in contrast to the solutions mentioned above, is not self-sufficient. It heavily relies on external data about users that it utilizes for purposes of adaptation. For PERSEUS to conceptually approach the utility of an industry-level personalization solution it should be accompanied by a user modeling server supporting a range of user profiling and user modeling services. Additionally/alternatively, support of a set of standardized or custom user data repositories could be sufficient. Also, to be usable on an industrial scale both PERSEUS and the external user data/model storage (user model server or otherwise) should be highly available, for example deployed on a cluster of machines.

6.4 FUTURE WORK

The work on deploying PERSEUS – a standalone adaptation functionality server – is not limited to providing adaptation to relatively small groups of students accessing several hundred interactive resources in the context of a university class. It is currently making a big step from the lab shelf intoreal life. We are participating in an Ensemble[14] - NSF NSDL Pathways project working to establish a national, distributed digital library for computing education.

This project is building a distributed portal providing access to a broad range of existing educational resources for computing. It is targeted at encouraging contribution, use, reuse, review, and evaluation of educational materials at multiple levels of granularity and seeks to support the full range of computing education communities including computer science, computer engineering, software engineering, information science, information systems, and information technology, as well as other areas. PERSEUS is currently a primary adaptation

[14] Ensemble Project http://nsdl.org/about/?pager=pathways&subpager=ENSEMBLE

provider for Ensemble's portal[15] responsible for offering social navigation support to the multiple communities of course designers, content authors, and educators.

In addition to the PERSEUS adaptation techniques that are discussed in this work, we are developing a number of hybrid approaches that combine link annotation and link recommendation techniques. In several of these approaches PERSEUS serves as a wrapper to an external recommendation engine and aggregates its output with the results of the concept-based navigation support method.

[15] Ensemble Computing portal http://www.computingportal.org/

BIBLIOGRAPHY

Aleahmad, T., Aleven, V., and Kraut, R. (2008). Open Community Authoring of Targeted Worked Example Problems. In B. P. Woolf, E. Aïmeur, R. Nkambou, and S. P. Lajoie (Eds.), 9th International Conference On Intelligent Tutoring Systems (ITS 2008), (pp. 216-227).

Aleahmad, T., Aleven, V., and Kraut, R. (2009). Creating a Corpus of Targeted Learning Resources with a Web-Based Open Authoring Tool. IEEE Transactions on Learning Technologies, 2(1), 3-9.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., and Pelletier, R. (1995). Cognitive Tutors: Lessons Learned. Journal of the Learning Sciences,, 4(2), 167-207.

Ardissono, L., Gena, C., Torasso, P., Bellifemine, F., Chiarotto, A., Difino, A., and Negro, B. (2003). Personalized Recommendation of TV Programs. In A. Cappelli and F. Turini (Eds.), Advances in Artificial Intelligence (AI*IA 2003), 8th Congress of the Italian Association for Artificial Intelligence, (pp. 474-486).

Bailey, C., Hall, W., Millard, D. E., and Weal, M. J. (2007). Adaptive hypermedia through contextualized open hypermedia structures. ACM Transactions on Information Systems, 25(4),© 2007 ACM, Inc. http://doi.acm.org/10.1145/1281485.1281487

Barrett, R. and Maglio, P. P. (1998). Intermediaries: New Places for Producing and Manipulating Web Content. Computer Networks, 30(1-7), 509-518.

Benhamou, E., Eisenberg, J., and Katz, R. H. (2010). Assessing the changing U.S. IT R\&D ecosystem. Communications of the ACM, 53(2), 76-83.

Bollen, J. and Heylighen, F. (1998). A system to restructure hypertext networks into valid user models. The New Review of Hypermedia and Multimedia, 4, 189-214.

Boyle, C. D. B. and Encarnacion, A. O. (1994). Metadoc: An Adaptive Hypertext Reading System. User Modeling and User-Adapted Interaction, 4(1), 1-19.

Bra, P. D. and Calvi, L. (1998). AHA! An open Adaptive Hypermedia Architecture. The New Review of Hypermedia and Multimedia, 4, 115-140.

Bra, P. D., Smits, D., and Stash, N. (2006). The design of AHA!. In U. K. Wiil, P. J. Nürnberg, and J. Rubart (Eds.), 17th ACM Conference on Hypertext and Hypermedia (Hypertext 2006), (pp. 171-195).

Brusilovsky, P. (1996). Methods and Techniques of Adaptive Hypermedia. User Modeling and User-Adapted Interaction, 6(2-3), 87-129.

Brusilovsky, P. (2001). Adaptive Hypermedia. User Modeling and User-Adapted Interaction, 11(1-2), 87-110.

Brusilovsky, P. (2003). Adaptive navigation support in educational hypermedia: The role of student knowledge level and the case for meta-adaptation. British Journal of Educational Technology, 34(4), 487-497.

Brusilovsky, P. (2004). KnowledgeTree: a distributed architecture for adaptive e-learning. In S. I. Feldman, M. Uretsky, M. Najork, and C. E. Wills (Eds.), 13th international conference on World Wide Web (WWW2004) Alternate Track Papers And Posters, (pp. 104-113).

Brusilovsky, P. (2001). WebEx: Learning from Examples in a Programming Course. In W. A. Lawrence-Fowler and J. Hasebrook (Eds.), World Conference on the WWW and Internet (WebNet 2001), (pp. 124-129).

Brusilovsky, P., Chavan, G., and Farzan, R. (2004). Social Adaptive Navigation Support for Open Corpus Electronic Textbooks. In P. D. Bra and W. Nejdl (Eds.), 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2004), (pp. 24-33).

Brusilovsky, P. and Eklund, J. (1998). A Study of User Model Based Link Annotation in Educational Hypermedia. Journal of Universal Computer Science, 4(4), 429-448.

Brusilovsky, P., Eklund, J., and Schwarz, E. W. (1998). Web-Based Education for All: A Tool for Development Adaptive Courseware. Computer Networks, 30(1-7), 291-300.

Brusilovsky, P., Karagiannidis, C., and Sampson, D. (2004). Layered evaluation of adaptive learning systems. International Journal of Continuing Engineering Education and Lifelong Learning, 14(4/5), 402-421.

Brusilovsky, P., Karagiannidis, C., and Sampson, D. (2001). The benefits of layered evaluation of adaptive applications and services. In S. Weibelzahl, D. Chin, and G. Weber (Eds.), 1st Workshop on Empirical Evaluation of Adaptive Systems, (pp. 1-8).

Brusilovsky, P., Knapp, J., and Gamper, J. (2006). Supporting teachers as content authors in intelligent educational systems. International Journal of Knowledge and Learning, 2(3/4), 191-215.

Brusilovsky, P. and Millán, E. (2007). The Adaptive Web, Methods and Strategies of Web Personalization. In P. Brusilovsky, A. Kobsa, and W. Nejdl (Eds.), The Adaptive Web, Methods and Strategies of Web Personalization (3-53). Springer.

Brusilovsky, P. and Pesin, L. (1998). Adaptive navigation support in educational hypermedia: An evaluation of the ISIS-Tutor. Journal of Computing and Information Technology, 6(1), 27-38.

Brusilovsky, P., Schwarz, E. W., and Weber, G. (1996). A tool for developing adaptive electronic textbooks on WWW. In World Conference of the Web Society (WebNet'96).

Brusilovsky, P., Sosnovsky, S., and Shcherbinina, O. (2004). QuizGuide: Increasing the Educational Value of Individualized Self-Assessment Quizzes with Adaptive Navigation Support. In J. Nall and R. Robson (Eds.), World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (ELEARN 2004), (pp. 1806-1813).

Brusilovsky, P., Sosnovsky, S., and Yudelson, M. (2009). Addictive links: the motivational value of adaptive link annotation. New Review of Hypermedia and Multimedia, 15(1), 97-118.

Brusilovsky, P. and Sosnovsky, S. A. (2005). Engaging students to work with self-assessment questions: a study of two approaches. In J. C. Cunha, W. M. Fleischman, V. K. Proulx, and J. Lourenco (Eds.), 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2005), (pp. 251-255).

Brusilovsky, P. and Sosnovsky, S. A. (2005). Individualized exercises for self-assessment of programming knowledge: An evaluation of QuizPACK. ACM Journal of Educational Resources in Computing, 5(2), .

Brusilovsky, P., Sosnovsky, S. A., Lee, D. H., Yudelson, M., Zadorozhny, V., and Zhou, X. (2008). An open integrated exploratorium for database courses. In J. Amillo, C. Laxer, E. M. Ruiz, and A. Young (Eds.), 13th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2008), (pp. 22-26).

Brusilovsky, P., Sosnovsky, S. A., and Shcherbinina, O. (2005). User Modeling in a Distributed E-Learning Architecture. In L. Ardissono, P. Brna, and A. Mitrovic (Eds.), 10th International Conference on User Modeling (UM 2005), (pp. 387-391).

Brusilovsky, P., Sosnovsky, S. A., and Yudelson, M. (2006). Addictive Links: The Motivational Value of Adaptive Link Annotation in Educational Hypermedia. In V. P. Wade, H. Ashman, and B. Smyth (Eds.), 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2006), (pp. 51-60).

Brusilovsky, P., Sosnovsky, S. A., Yudelson, M., and Chavan, G. (2005). Interactive Authoring Support for Adaptive Educational Systems. In C.-K. Looi, G. I. McCalla, B. Bredeweg, and J. Breuker (Eds.), 12th International Conference on Artificial Intelligence in Education (AIED 2005), (pp. 96-103).

Brusilovsky, P., Yudelson, M., and Sosnovsky, S. (2004). An Adaptive E-Learning Service for Accessing Interactive Examples. In J. Nall and R. Robson (Eds.), World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (ELEARN 2004), (pp. 2556-2561).

Böcker, H.-D., Hohl, H., and Schwab, T. (1990). Hypadapter - Ein adaptives Hypertextsystem zur Präsentation von Lerninhalten. In P. A. Gloor and N. A. Streitz (Eds.), Hypertext und Hypermedia, (pp. 230-234).

Cafolla, R. (2006). Project MERLOT: Bringing Peer Review to Web-Based Educational Resources. Journal of Technology and Teacher Education, 14(2), 313-323.

Carbonell, J. R. (1970). AI in CAI: An Artificial-Intelligence Approach to Computer-Assisted Instruction. IEEE Transactions on Man-Machine Systems, 11(4), 190-202.

Carmichael, D. J., Kay, J., and Kummerfeld, B. (2005). Consistent Modelling of Users, Devices and Sensors in a Ubiquitous Computing Environment. User Modeling and User-Adapted Interaction, 15(3-4), 197-234.

Carr, L., Hall, W., Bechhofer, S., and Goble, C. A. (2001). Conceptual linking: ontology-based open hypermedia. In 10th international Conference on World Wide Web (WWW 2001), (pp. 334-342).

Cheng, R. and Vassileva, J. (2005). Adaptive Reward Mechanism for Sustainable Online Learning Community. In C.-K. Looi, G. I. McCalla, B. Bredeweg, and J. Breuker (Eds.), 12th International Conference on Artificial Intelligence in Education (AIED 2005), (pp. 152-159).

Chin, D. N. (2001). Empirical Evaluation of User Models and User-Adapted Systems. User Modeling and User-Adapted Interaction, 11(1-2), 181-194.

Claypool, M., Le, P., Waseda, M., and Brown, D. (2001). Implicit interest indicators. In C. Sidner and J. Moore (Eds.), 6th International Conference on Intelligent User Interfaces (IUI 2001), (pp. 33-40).

Conlan, O., Hockemeyer, C., Wade, V. P., Albert, D., and Gargan, M. (2002). An Architecture for integrating Adaptive Hypermedia Services with Open Learning Environments. In P. Barker and S. Rebelsky (Eds.), World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA 2002), (pp. 344-350).

Conlan, O. and Wade, V. P. (2004). Evaluation of APeLS - An Adaptive eLearning Service Based on the Multi-model, Metadata-Driven Approach. In P. D. Bra and W. Nejdl (Eds.), 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2004), (pp. 291-295).

Cristea, A., Smits, D., and De Bra, P. (2005). Writing MOT, Reading AHA! - converting between an authoring and a delivery system for adaptive educational hypermedia. In A. Cristea, R. Carro, and F. Garzotto (Eds.), 3rd Workshop of Authoring of Adaptive and Adaptable Educational Hypermedia (A3EH), (pp. 36-45).

Cristea, A. I., Ashman, H., Stewart, C. D., and Cristea, P. (2005). Evaluation of adaptive hypermedia systems' conversion. In S. Reich and M. Tzagarakis (Eds.), Hypertext, (pp. 129-131).

Debevc, M., Meyer, B., Donlagic, D., and Svecko, R. (1996). Design and Evaluation of an Adaptive Icon Toolbar. User Modeling and User-Adapted Interaction, 6(1), 1-21.

Debevc, M., Meyer, B., and Svecko, R. (1997). An Adaptive Short List for Documents on the World Wide Web. In 1997 International Conference on Intelligent User Interfaces (IUI'97), (pp. 209-211).

Denaux, R., Dimitrova, V., and Aroyo, L. (2005). Integrating Open User Modeling and Learning Content Management for the Semantic Web. In L. Ardissono, P. Brna, and A. Mitrovic (Eds.), 10th International Conference on User Modeling (UM 2005), (pp. 9-18).

Denny, P., Luxton-Reilly, A., and Hamer, J. (2008). Student use of the PeerWise system. In J. Amillo, C. Laxer, E. M. Ruiz, and A. Young (Eds.), 13th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2008), (pp. 73-77).

Dieberger, A. and Guzdial, M. (2003). CoWeb - experiences with collaborative Web spaces. In C. Lueg and D. Fisher (Eds.), From Usenet to CoWeb (155-166). New York, NY, USA: Springer-Verlag.

Dolog, P., Henze, N., and Nejdl, W. (2003). Logic-based open hypermedia for the semantic web. In 1st International Workshop on Hypermedia and the Semantic Web at Hypertext 2003.

Dolog, P., Henze, N., Nejdl, W., and Sintek, M. (2004). The Personal Reader: Personalizing and Enriching Learning Resources Using Semantic Web Technologies.. In P. D. Bra and W. Nejdl (Eds.), 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2004), (pp. 85-94).

Dron, J., Mitchell, R., Siviter, P., and Boyne, C. (2000). CoFIND - An experiment in N-dimensional collaborative filtering. Journal of Network and Computer Applications, 23(2), 131-142.

Eklund, J. and Brusilovsky, P. (1999). InterBook: An Adaptive Tutoring System. UniServe Science News, 12, 8-13.

Farzan, R. and Brusilovsky, P. (2005). Social Navigation Support Through Annotation-Based Group Modeling. In L. Ardissono, P. Brna, and A. Mitrovic (Eds.), User Modeling, (pp. 463-472).

Farzan, R. and Brusilovsky, P. (2008). AnnotatEd: A social navigation and annotation service for web-based educational resources. New Review of Hypermedia and Multimedia, 14(1), 3 - 32.

Finin, T. W. (1989). GUMS: A General User Modeling Shell. In A. Kobsa and W. Wahlster (Eds.), User Models in Dialog Systems (411-430). New York, USA: Springer-Verlag.

Finin, T. W. and Drager, D. (1986). GUMS1: A General User Modeling System. In Proceedings of the Workshop on Strategic Computing Natural Language, (pp. 224 - 230).

Gardner, H. (2009, May/June). The Next Big Thing: Personalized Education. Foreign Policy, .

Gehringer, E., Conger, S. G., and Wagle, P. (2007). Reusable Learning Objects Through Peer Review: The Expertiza Approach. Journal of Online Education, 3(5), .

Geldof, S. (1998). Con-textual navigation support. The New Review of Hypermedia and Multimedia, 4, 47-66.

Gena, C. and Torre, I. (2004). The importance of adaptivity to provide onboard services: A preliminary evaluation of an adaptive tourist information service onboard vehicles. Applied Artificial Intelligence, 18(6), 549-580.

Gertner, A. S. and VanLehn, K. (2000). Andes: A Coached Problem Solving Environment for Physics. In G. Gauthier, C. Frasson, and K. VanLehn (Eds.), 5th International Conference on Intelligent Tutoring Systems (ITS 2000), (pp. 133-142).

Hammond, N. (1989). Hypermedia and Learning: Who Guides Whom? (Invited Paper). In H. A. Maurer (Ed.), 2nd International Conference on Computer Assisted Learning (ICCAL '89), (pp. 167-181).

Henze, N. and Nejdl, W. (1999). Adaptivity in the KBS Hyperbook System. In P. Brusilovsky, P. De Bra, and A. Kobsa (Eds.), 2nd Workshop on Adaptive Systems and User Modeling on the World Wide Web, (pp. 67-74).

Henze, N. and Nejdl, W. (2001). Adaptation in Open Corpus Hypermedia. International Journal of Artificial Intelligence in Education, 12(4), 325-350.

Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. (1999). An Algorithmic Framework for Performing Collaborative Filtering. In 22nd Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR '99), (pp. 230-237).

Hirashima, T., Hachiya, K., Kashihara, A., and Toyoda, J. (1997). Information Filtering Using User's Context on Browsing in Hypertext. User Modeling and User-Adapted Interaction, 7(4), 239-256.

Hohl, H., Böcker, H.-D., and Gunzenhäuser, R. (1996). Hypadapter: An Adaptive Hypertext System for Exploratory Learning and Programming. User Modeling and User-Adapted Interaction, 6(2-3), 131-156.

Hsiao, I.-H., Brusilovsky, P., and Sosnovsky, S. (2008). Web-based Parameterized Questions for Object-Oriented Programming. In C. J. Bonk, M. M. Lee, and T. Reynolds (Eds.), World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (ELEARN 2008), (pp. 3728-3735).

Hsiao, S. and Brusilovsky, P. (2010). The Role of Community Feedback in the Student Example Authoring Process: an Evaluation of AnnotEx. British Journal of Educational Technology, in press, .

Höök, K. (1998). Evaluating the utility and usability of an adaptive hypermedia system. Knowledge-Based Systems, 10(5), 311-319.

Höök, K. (2000). Steps to take before intelligent user interfaces become real. Interacting with Computers, 12(4), 409-426.

Iglezakis, D. (2005). Is the ACT-value a Valid Estimate for Knowledge? An Empirical Evaluation of the Inference Mechanism of an Adaptive Help System. In S. Weibelzahl, A. Paramythis, and J. Masthoff (Eds.), 4th Workshop on the Evaluation of Adaptive Systems, in conjunction with 10th International Conference on User Modeling (UM'05), (pp. 19-26).

Jameson, A. (2003). The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications. In J. A. Jacko and A. Sears (Eds.), (305-330). Hillsdale, NJ, USA: L. Erlbaum Associates Inc.

Kaplan, C. A., Fenwick, J., and Chen, J. (1993). Adaptive Hypertext Navigation Based On User Goals and Context. User Modeling and User-Adapted Interaction, 3(3), 193-220.

Kaptelinin, V. (1993). Item recognition in menu selection: the effect of practice. In S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, and T. N. White (Eds.), IFIP TC13 International Conference on Human-Computer Interaction (INTERACT'93) joint with ACM Conference on Human Aspects in Computing Systems (CHI'93), (pp. 183-184).

Karagiannidis, C. and Sampson, D. G. (2000). Layered Evaluation of Adaptive Applications and Services. In P. Brusilovsky, O. Stock, and C. Strapparava (Eds.), International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2000), (pp. 343-346).

Kavcic, A. (2004). Fuzzy user modeling for adaptation in educational hypermedia. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 34(4), 439 - 449.

Kay, J. (1995). The um Toolkit for Coopertive User Modeling. User Modeling and User-Adapted Interaction, 4(3), 149-196.

Kegel, D. (2010). The C10K Problem. Retrieved from http://www.kegel.com/c10k.html.

Kelly, D. and Tangney, B. (2005). Matching and Mismatching Learning Characteristics with Multiple Intelligence Based Content. In C.-K. Looi, G. I. McCalla, B. Bredeweg, and J. Breuker (Eds.), 12th International Conference on Artificial Intelligence in Education (AIED 2005), (pp. 354-361).

Knutov, E., Bra, P. D., and Pechenizkiy, M. (2009). AH 12 years later: a comprehensive survey of adaptive hypermedia methods and techniques. New Review of Hypermedia and Multimedia, 15(1), 5 - 38.

Kobsa, A. (1990). Modeling the user's conceptual knowledge in BGP-MS, a user modeling shell system. Computational Intelligence, 6(4), 193-208.

Kobsa, A. and Fink, J. (2006). An LDAP-based User Modeling Server and its Evaluation. User Modeling and User-Adapted Interaction, 16(2), 129-169.

Kobsa, A. and Pohl, W. (1995). The User Modeling Shell System BGP-MS. User Modeling and User-Adapted Interaction, 4(2), 59-106.

Koidl, K., Conlan, O., and Wade, V. (2009). Non-Invasive Adaptation Service for Web-based Content Management Systems. In P. De Bra and M. Pechenizkiy (Eds.), International Workshop on Dynamic and Adaptive Hypertext.

Krogsäter, M., Oppermann, R., and Thomas, C. (1994). A User Interface: Integrating Adaptability and Adaptivity. In R. Oppermann (Ed.), Adaptive User Support: Ergonomic Design of Manually and Automatically Adaptable Software (97-125). Hillsdale, N.J: Lawrence Erlbaum Associates.

Lundgren-Cayrol, K., Paquette, G., Miara, A., Bergéron, F., and Rivard, J. (2001). Explor@ Advisory Agent: Tracing the Student's Trail. In W. A. Lawrence-Fowler and J. Hasebrook (Eds.), World Conference on the WWW and Internet (WebNet 2001), (pp. 802-808).

Lutkenhouse, T., Nelson, M. L., and Bollen, J. (2005). Distributed, real-time computation of community preferences. In S. Reich and M. Tzagarakis (Eds.), 16th ACM Conference on Hypertext and Hypermedia (Hypertext 2005), (pp. 88-97).

Maccaux, M. (2008). Approaches to Performance Testing. Retrieved from http://weblogic.sys-con.com/node/185298.

Magnini, B. and Strapparava, C. (2001). Improving User Modelling with Content-Based Techniques. In M. Bauer, P. J. Gmytrasiewicz, and J. Vassileva (Eds.), 8th International Conference on User Modeling (UM 2001), (pp. 74-83).

Masters, J., Madhyastha, T., and Shakouri, A. (2008). ExplaNet: A Collaborative Learning Tool and Hybrid Recommender System for Student-Authored Explanations. Journal of Interactive Learning Research, 19(1), 51-74.

Masthoff, J. (2003). The evaluation of adaptive systems. In N. V. Patel (Ed.), Adaptive evolutionary information systems (329-347). Hershey, PA, USA: IGI Publishing.

Masthoff, J. and Gatt, A. (2006). In pursuit of satisfaction and the prevention of embarrassment: affective state in group recommender systems. User Modeling and User-Adapted Interaction, 16(3-4), 281-319.

Meyer, B. (1994). Adaptive performance support: User acceptance of a self-adapting system. In B. Goodman, A. Kobsa, and D. Litman (Eds.), 4th International Conference on User Modeling (UM 1994), (pp. 65-70).

Millard, D. E., Moreau, L., Davis, H. C., and Reich, S. (2000). FOHM: a fundamental open hypertext model for investigating interoperability between hypertext domains. In 11th ACM Conference on Hypertext and Hypermedia (Hypertext 2000), (pp. 93-102).

Mitrovic, A. (2003). An Intelligent SQL Tutor on the Web. Iinternational Journal of Artificial Intelligence in Education, 13(2-4), 173-197.

Murray, T. (1999). Authoring Intelligent Tutoring Systems: Analysis of the state of the art. International Journal of Artificial Intelligence in Education, 10(1), 98-129.

Nielsen, J. (1993). Usability Engineering. San Francisco, California, USA: Morgan Kaufmann Publishers.

Nielsen, J. and Molich, R. (1990). Heuristic evaluation of user interfaces. In J. C. Chew and J. Whiteside (Eds.), SIGCHI conference on Human factors in computing systems (CHI'90), (pp. 249-256).

Oppermann, R. (1994). Adaptively supported adaptability. International Journal of Human Computer Studies, 40(3), 455-472.

Orwant, J. (1995). Heterogeneous Learning in the Doppelgänger User Modeling System. User Modeling and User-Adapted Interaction, 4(2), 107-130.

Paiva, A. and Self, J. A. (1995). TAGUS - A User and Leamer Modeling Workbench. User Modeling and User-Adapted Interaction, 4(3), 197-226.

Paramythis, A., Totter, A., and Stephanidis, C. (2001). A modular approach to the evaluation of Adaptive User Interfaces. In S. Weibelzahl, D. Chin, and G. Weber (Eds.), 1st Workshop on Empirical Evaluation of Adaptive Systems, (pp. 9-24).

Paramythis, A. and Weibelzahl, S. (2005). A Decomposition Model for the Layered Evaluation of Interactive Adaptive Systems. In L. Ardissono, P. Brna, and A. Mitrovic (Eds.), 10th International Conference on User Modeling (UM 2005), (pp. 438-442).

Pazzani, M. J. (1999). A Framework for Collaborative, Content-Based and Demographic Filtering. Artificial Intelligenve Review, 13(5-6), 393-408.

Pazzani, M. J., Muramatsu, J., and Billsus, D. (1996). Syskill and Webert: Identifying Interesting Web Sites. In 13th National Conference on Artificial Intelligence and 8th Conference on Innovative Applications of Artificial Intelligence (AAAI'96), (pp. 54-61).

Pohl, W. (1997). LaboUr - Machine Learning for User Modeling. In M. J. Smith, G. Salvendy, and R. J. Koubek (Eds.), Design of Computing Systems: Social and Ergonomic Considerations, 7th International Conference on Human-Computer Interaction, (HCI International '97), (pp. 27-30).

Polson, P. G., Lewis, C., Rieman, J., and Wharton, C. (1992). Cognitive Walkthroughs: A Method for Theory-Based Evaluation of User Interfaces. International Journal of Man-Machine Studies, 36(5), 741-773.

Ramp, E., Bra, P. D., and Brusilovsky, P. (2005). High-level translation of adaptive hypermedia applications. In S. Reich and M. Tzagarakis (Eds.), 16th ACM Conference on Hypertext and Hypermedia (Hypertext 2005), (pp. 126-128).

Ritter, S., Anderson, J., Cytrynowicz, M., and Medvedeva, O. (1998). Authoring Content in the PAT Algebra Tutor. Journal of Interactive Media in Education, 98(9), .

Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In WWW, (pp. 285-295).

Schwab, I., Pohl, W., and Koychev, I. (2000). Learning to recommend from positive evidence. In 2000 International Conference on Intelligent User Interfaces (IUI'2000), (pp. 241-247).

Sears, A. and Shneiderman, B. (1994). Split Menus: Effectively Using Selection Frequency to Organize Menus. ACM Transactions on Computer-Human Interaction, 1(1), 27-51.

Shneiderman, B. (1984). Response Time and Display Rate in Human Performance with Computers. ACM Computing Surveys, 16(3), 265-285.

Smith, A. S. G. and Blandford, A. (2003). MLTutor: An Application of Machine Learning Algorithms for an Adaptive Web-based Information System. International Journal of Artificial Intelligence in Education, 13(2-4), 235-261.

Sosnovsky, S. and Brusilovsky, P. (2005). Layered Evaluation of Topic-Based Adaptation to Student Knowledge. In S. Weibelzahl, A. Paramythis, and J. Masthoff (Eds.), 4th Workshop on the Evaluation of Adaptive Systems, in conjunction with 10th International Conference on User Modeling (UM'05), (pp. 47-56).

Sosnovsky, S., Yudelson, M., and Brusilovsky, P. (2007). Community-oriented course authoring to support topic-based student modeling. In N. Capuano, D. Dicheva, A. Harrer, and R. Mizoguchi (Eds.), 5th International Workshop on Ontologies and Semantic Web for E-Learning (SWEL'2007), (pp. 91-100).

Sosnovsky, S. A., Brusilovsky, P., Lee, D. H., Zadorozhny, V., and Zhou, X. (2008). Re-assessing the Value of Adaptive Navigation Support in E-Learning Context. In W. Nejdl, J. Kay, P. Pu, and E. Herder (Eds.), 5th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2008), (pp. 193-203).

Totterdell, P. and Boyle, E. (1990). The Evaluation of Adaptive Systems. In D. Browne, P. Totterdell, and M. Norman (Eds.), Adaptive User Interfaces (161-194). London, UK: Academic Press Ltd.

Turner, T. E., Macasek, M. A., Nuzzo-Jones, G., Heffernan, N. T., and Koedinger, K. R. (2005). The Assistment Builder: A Rapid Development Tool for ITS. In C.-K. Looi, G. I.

McCalla, B. Bredeweg, and J. Breuker (Eds.), 12th International Conference on Artificial Intelligence in Education (AIED 2005), (pp. 929-931).

Vygotsky, L. S. (1978). Mind and society: The development of higher mental processes. Cambridge, MA: Harvard University Press.

Wang, Y. and Kobsa, A. (2009). Performance Evaluation of a Dynamic Privacy-Enhancing Framework for Personalized Websites. In G.-J. Houben, G. I. McCalla, F. Pianesi, and M. Zancanaro (Eds.), 17th International Conference on User Modeling, Adaptation, and Personalization (UMAP 2009), (pp. 78-89).

Weber, G. and Brusilovsky, P. (2001). ELM-ART: An adaptive versatile system for Web-based instruction. International Journal of Artificial Intelligence in Education, 12(4), 351-384.

Weber, G. and Specht, M. (1997). User modeling and adaptive navigation support in WWW-based tutoring systems. In A. Jameson, C. Paris, and C. Tasso (Eds.), 6th International Conference on User Modeling (UM 1997), (pp. 289-300).

Weibelzahl, S. (2005). Problems and pitfalls in the evaluation of adaptive sytems. In S. Y. Chen and G. D. Magoulas (Eds.), Adaptable and Adaptive Hypermedia Systems (285-299). Hershey, PA, USA: IRM Press.

Weibelzahl, S. and Lauer, C. U. (2001). Framework for the Evaluation of Adaptive CBR-Systems. In I. Vollrath, S. Schmitt, and U. Reimer (Eds.), Experience Management as Reuse of Knowledge. 9th German Workshop on Case Based Reasoning (GWCBR2001), (pp. 254-263).

Weibelzahl, S. and Weber, G. (2003). Evaluating the Inference Mechanism of Adaptive Learning Systems. In P. Brusilovsky, A. T. Corbett, and F. de Rosis (Eds.), 9th International Conference on User Modeling (UM 2003), (pp. 154-162).

Yan, T. W., Jacobsen, M., Garcia-Molina, H., and Dayal, U. (1996). From User Access Patterns to Dynamic Hypertext Linking. Computer Networks, 28(7-11), 1007-1014.

Yudelson, M. and Brusilovsky, P. (2005). NavEx: Providing Navigation Support for Adaptive Browsing of Annotated Code Examples. In C.-K. Looi, G. I. McCalla, B. Bredeweg, and J. Breuker (Eds.), 12th International Conference on Artificial Intelligence in Education (AIED 2005), (pp. 710-717).

Yudelson, M. and Brusilovsky, P. (2008). Adaptive Link Annotation in Distributed Hypermedia Systems: The Evaluation of a Service-Based Approach. In W. Nejdl, J. Kay, P. Pu, and E. Herder (Eds.), 5th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2008), (pp. 245-254).

Yudelson, M., Brusilovsky, P., and Zadorozhny, V. (2007). A User Modeling Server for Contemporary Adaptive Hypermedia: An Evaluation of the Push Approach to Evidence Propagation. In C. Conati, K. F. McCoy, and G. Paliouras (Eds.), 11th International Conference on User Modeling (UM 2007), (pp. 27-36).

Zadorozhny, V., Yudelson, M., and Brusilovsky, P. (2008). A framework for performance evaluation of user modeling servers for Web applications. Web Intelligence and Agent Systems, 6(2), 175-191.

Zellweger, P., Chang, B.-W., and Mackinlay, J. D. (1998). Fluid Links for Informed and Incremental Link Transitions. In 9th ACM Conference on Hypertext and Hypermedia (Hypertext '98), (pp. 50-57).

de La Passardiére, B. and Dufresne, A. (1992). Adaptive Navigational Tools for Educational Hupermedia. In I. Tomek (Ed.), 4th International Conference on Computer Assisted Learning (ICCAL'92), (pp. 555-567).