DAVID SMITS
TOWARDS A
GENERIC
DISTRIBUTED
ADAPTIVE
HYPERMEDIA ENVIRONMENT
Towards a Generic Distributed Adaptive Hypermedia Environment

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Smits, David
Towards a Generic Distributed Adaptive Hypermedia Environment

A catalogue record is available from the Eindhoven University of Technology Library
NUR 988

Keywords: hypertext / hypermedia / databases / adaptive hypermedia

SIKS Dissertation Series No. 2012-10
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Cover Design: Tom Meurs.
Printed by University Press Facilities, Eindhoven, the Netherlands.

Copyright © 2012 by David Smits.

All rights reserved. No part of this dissertation publication may be reproduced, stored in retrieval systems, or transmitted in any form by any means, mechanical, photocopying, recording, or otherwise, without written consent of the author.

Towards a Generic Distributed Adaptive Hypermedia Environment

PROEFONTWERP

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de rector magnificus, prof.dr.ir. C.J. van Duijn, voor een commissie aangewezen door het College voor Promoties in het openbaar te verdedigen op maandag 2 april 2012 om 16.00 uur

door

David Smits

geboren te Brunssum
De documentatie van het proefontwerp is goedgekeurd door de promotor:

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

Copromotor:
dr. M. Pechenizkiy

Contents

Executive summary  7
Adaptive thesis  17
Summary  18
Samenvatting  19
Acknowledgements  21
Curriculum Vitae  23
Executive summary

Authors have used written text for centuries to convey ideas. They were constrained by the linear nature of books, papers, and the like. Hypertext has empowered authors, in that it allows interlinking fragments and pages of text and thus creating a document that is radically different from a linear text. Going from linear text to hypertext requires a change in mindset for both authors and readers. Only when authors come to convey ideas using this newfound power and when readers learn to cope with this new navigational freedom, will hypertext reach its full potential.

Though the idea of hypertext dates back to the 1940's, it was not until the technological context was available that authors started to use it en masse. However, the same technological context can be used to empower authors even further. Conveying ideas is still the driving force behind any authoring, linear text and hypertext. Taking your readers into account while writing is something every author should do. The technology described in this thesis will allow an author to create adaptive hypermedia, a text that will depend on the reader’s knowledge, interest, goals, etc.

When World Wide Web was first introduced it was intended as a means for sharing information. It uses information pages and links, and is therefore a form of hypermedia. Nowadays the Web is being used as a common user interface platform for all different kinds of applications, from communication (through discussion forums), shopping, banking and other global types of services down to controlling your heating, lighting and other types of devices in the home, and using interface devices ranging from mobile phones to huge computer monitors and wall-size displays. This thesis does not deal with the possibilities for adaptive technology in all these aspects of the Web, but concentrates on the Web as the hypermedia platform it is mainly used for: we look at applications in which users interact with information (pages), like in online courses, news-services, museum sites, (shopping) catalogs, encyclopedia, etc. We look at how these applications can adapt themselves to their individual users or user groups, through the interaction of the users with the applications.
The main focus of the work described here was to create the technological context necessary to support authors in creating these adaptive hypermedia applications. As such, we call the development author-driven. Several considerations have played an important role in the design of the developed languages, models, and systems:

- Reading (accessing) an adaptive hypertext should not be slower than reading a traditional hypertext;
- Authors should be able to express any adaptivity based on the user;
- The full complexity of adaptive behavior should be hidden from authors who don’t need to see or change it;
- Setting up the required adaptive hypermedia system should be easy.

Designing a system that takes these considerations into account has led to several questions.

What does a usable generic adaptive hypermedia system look like?

The term ‘generic adaptive hypermedia system’ can be applied to many frameworks. Changing the output of an HTTP server based on some characteristics of the client (possibly after logging in) has been possible since an early stage of the World Wide Web. Specialized server-side code written by an author (or programmer) for his specific adaptive application is one of the first methods used to add adaptive features to hypertext. Key in this research question is the term ‘usable’. How do we improve the authoring process? How can we make it accessible to authors who have no skill in programming, while still maintaining the expressive power of a programming language?

What performance considerations affect its design?

One of the goals mentioned on the introduction page was that ‘reading an adaptive hypertext should not be slower than reading a traditional hypertext’. However, an adaptive web server has more work to do than a traditional web server. There are user profiles to maintain and update. These updates need to occur within the time spent on the request, because they might affect the adaptation. And the adaptation of the document itself takes time when compared to only serving static content. How can we build a system that can manage all the extra load and still produce a result in the time span that would be experienced as instantaneous (roughly 100 ms)?

How can the design be improved to support the global scale of the World Wide Web?

Building a system that can handle the requests of a few users concurrently is one thing. Maintaining performance while it scales to handle hundreds, thousands, or more, is another. How do we construct an architecture that allows adaptive hypermedia to be as fast and accessible as the current (Web-based) hypertext is?

Authoring Adaptive Hypermedia

Authoring any hypermedia application involves writing content. The de facto standard for writing hypermedia is (X)HTML. Several tools exist to aid an author in writing (X)HTML. While some of them hide the source code completely and only show a graphical user interface, most will allow the author to manipulate the (X)HTML code directly. This allows beginner and expert authors to create (X)HTML.

A similar approach seems logical when creating an adaptive hypermedia application. It will still be a hypermedia application. Hence, writing (X)HTML is still a vital part. From that perspective, the content is a good place for the author to write instructions that are useful for an adaptation engine, e.g. “skip this section for users with ‘this’ in their profile”. As most tools to author hypermedia allow the editing of the actual (X)HTML, no effort was made to improve on this part of authoring an adaptive hypermedia application.

Additionally, to perform adaptation, the adaptive webserver requires a domain and adaptation model. This is a different type of authoring task when compared to writing the content. It is a task that is to some extent unique to adaptive hypermedia and as a result tools to make authoring easy have been developed over time. This thesis presents two adaptive webservers for which such tools have been developed, namely AHA! and GALE.

At first, adaptive behavior was authored for every concept individually using an XML format and later a graphical tool to edit the
format. Anyone authoring an application this way will quickly recognize patterns in the rules created. A common attribute is the number of times a concept has been requested called a 'visited counter'. Some notion of "user’s knowledge of a concept" is used in many applications. If the concepts form a hierarchical structure, it is common to propagate knowledge through the structure. Prerequisites might depend on knowledge values of other concepts. Using the early XML format and accompanying tool proved tedious when authoring this type of repetitive behavior.

Recognizing these patterns in adaptation rules prompted the development of a higher level authoring tool. The tool was called the 'Graph Author' (developed by a student at the TU/e as part of an internship). It allows an author to draw a graph where concepts are nodes and edges are relations that impose template adaptive behavior to the associated concepts. The tool made authoring an adaptive application easier, improving the usability of the adaptive system from an author's perspective.

In GALE the expressive power increased as the language used to author adaptive behavior moved from simple if-then-else rules to allowing rules written in Java. New formats and languages were developed to support authoring these rules, including the GAM language to author the domain and adaptation models. The language has built-in support to facilitate templates, a function used in external authoring tools like the graphical tool GAT. Some authors will prefer the use of such a graphical tool, while other authors prefer to edit the textual format directly (an observation that can be made about hypertext in general). Concept inheritance was introduced to provide the textually oriented author with the means to easily author GAM (i.e. without repeating the description for similar adaptive behavior in multiple places).

**Generic Adaptation Language and Engine (GALE)**

GALE as a project comprises a set of languages and a runtime engine to support adaptive hypermedia. The languages are intended to support the authoring process and are used as input for the adaptation engine and its various components. GALE as a software package refers to this entire architecture that can optionally be distributed over multiple computers. The image below shows the global structure.

Clients (readers using a browser to access webpages) are shown on the left and connect to the Adaptation Engine component by requesting concepts. Various smaller modules are used to service the request:
- The Login Manager handles everything related to logging in and registering users.
- The Concept Manager translates the HTTP request to a request for a concept.
- The Processor Stack (and subcomponents) updates some user profile information based on the concept access and loads and adapts resources based on information in the user profile.
- Various caches are loosely coupled with the outside world and provide a virtual view of the domain, adaptation and user models as provided by (possibly external) services.

The Event Bus uses the publisher/subscriber paradigm to route events between the various registered components.
Domain Model services provide details on domain and adaptation models. Various DM services are included in GALE. For instance, the GAM service reads DM/AM descriptions contained in .gam files that can be located anywhere on the Internet. The Adaptation Engine and other components that require access to the global domain model are unaware of the origin of these concept descriptions.

The User Model service (typically one for every GALE instance) provides access to the user model and might use information in the adaptation model that contains rules associated with user model values. The service views any update to the user model as a possible event that might lead to further updates. Any resulting change to the user model that might be relevant to the other components is communicated over the Event Bus to allow possible caches to stay valid.

The GRAPPLE project (http://www.grapple-project.org) provided tools to integrate an adaptive hypermedia system with Learning Management Systems (LMS). The components in the bottom right of the overview image form a bridge between GALE’s Event Bus and components developed in GRAPPLE, like the GRAPPLE Event Bus (GEB), the GRAPPLE User Model Framework (GUMF) and the GRAPPLE Authoring Tools (GAT).

Performance

Being able to respond to user requests quickly (preferably within 100 ms, but no longer than one second) is an important design requirement that affected GALE’s overall architecture. The ability to distribute the various components over multiple machines allows the use of more processing power when needed. To minimize the communication required between various components, caches are used throughout GALE.

Most components are aware that information they provide might be cached. They are built in such a way that whenever they provide information, they are also responsible for keeping the information up to date. (They thus not only invalidate but also update the cache.) This method tends to work well because most information is read often but updated rarely. For instance, whenever the User Model service provides information on particular attributes, it will resume responsibility over maintaining those attributes. Whenever their value changes, interested components are notified through the Event Bus.

GALE’s caches allow most of the information required to perform adaptation (the domain, adaptation and user model) to be instantly available in the Adaptation Engine component, eliminating much of the communication that would otherwise be necessary.

Performance considerations played an important role in the decision on how to implement adaptation rules. Though most parts of GALE are unaware of the actual language used to describe these rules, the default is a language that mixes Java with GEL (GALE Expression Language). GEL allows easy access to the domain and adaptation model and is translated to Java on the fly. The result is Java source code that can be compiled to byte code. Since most adaptive documents tend to use similar rules often, this allows GALE to cache and reuse the compiled code, eliminating much of the interpretation of adaptation rules.

These two factors have boosted GALE’s performance to a level that is experienced as instantly by most users. After the initial warming up as GALE’s caches fill with information, random pages of the adaptive thesis are served in under 20 ms.

Though this is a great figure when used by a single user, the question remains how GALE will perform under stress. To answer that question we setup a test suite using Apache JMeter (http://jmeter.apache.org). A number of concurrent threads were used to request random pages of the thesis and different user ids were used to simulate multiple users. The results are displayed in the table and accompanying graph:

1 All tests were executed using a HP Compaq 6710b notebook running GALE version 1.2. The notebook has an Intel Core 2 CPU T7500 at 2.20 GHz, 2 GB of RAM and runs Windows 7 32-bit.
<table>
<thead>
<tr>
<th>Threads</th>
<th>Average (ms)</th>
<th>Deviation (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>201</td>
<td>51</td>
</tr>
<tr>
<td>20</td>
<td>262</td>
<td>151</td>
</tr>
<tr>
<td>30</td>
<td>420</td>
<td>282</td>
</tr>
<tr>
<td>40</td>
<td>826</td>
<td>777</td>
</tr>
<tr>
<td>50</td>
<td>1293</td>
<td>1552</td>
</tr>
<tr>
<td>60</td>
<td>1623</td>
<td>1821</td>
</tr>
<tr>
<td>70</td>
<td>2116</td>
<td>2599</td>
</tr>
<tr>
<td>80</td>
<td>2546</td>
<td>2389</td>
</tr>
<tr>
<td>90</td>
<td>3278</td>
<td>3322</td>
</tr>
<tr>
<td>100</td>
<td>3832</td>
<td>3516</td>
</tr>
</tbody>
</table>

GALE's primary use is to serve adaptive hypertext where the reader is assumed to spend some time studying the material presented. Assuming the average time spent reading a page is 20 seconds and the number of concurrent threads that can be handled within 1 second is around 45 (see table), this would allow 900 users that are actively reading the thesis to do so comfortably.

These figures can be improved by allocating more memory to GALE's caches. By default the User Model service stores 1000 entries and the Adaptation Engine 5000. Changing both these numbers to 32768 results in the following statistics:

<table>
<thead>
<tr>
<th>Threads</th>
<th>Average (ms)</th>
<th>Deviation (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>27</td>
<td>8</td>
</tr>
<tr>
<td>20</td>
<td>99</td>
<td>66</td>
</tr>
<tr>
<td>30</td>
<td>174</td>
<td>140</td>
</tr>
<tr>
<td>40</td>
<td>326</td>
<td>218</td>
</tr>
<tr>
<td>50</td>
<td>377</td>
<td>233</td>
</tr>
<tr>
<td>60</td>
<td>549</td>
<td>396</td>
</tr>
<tr>
<td>70</td>
<td>570</td>
<td>398</td>
</tr>
<tr>
<td>80</td>
<td>701</td>
<td>477</td>
</tr>
<tr>
<td>90</td>
<td>789</td>
<td>601</td>
</tr>
<tr>
<td>100</td>
<td>966</td>
<td>738</td>
</tr>
</tbody>
</table>

These settings allow 2000 users to actively view content through a single GALE server running on a notebook from 2007.

**Conclusions**

Adaptive hypermedia have great potential. There seems to be no end to the amount of information available on the World Wide Web today. When authors, publishing this huge amount, would have the ability to write things differently or write different things, based on their knowledge of the reader, would they use that power? One can never
answer that question without having realized the technical context to support it.

Many adaptive hypermedia systems have been created. However, the added complexity of authoring adaptive hypermedia and the limited functionality and performance issues of adaptive hypermedia systems have limited the amount of material produced. This thesis, and accompanying prototype GALE, is based on 15 years of experience of authors writing adaptive hypermedia and engineers developing systems to support them. It answers the questions raised in the introduction.

GALE is a 'generic' AHS. It can perform most adaptation using rules that govern the adaptive behavior. It is easily extended to add new functionality. GALE is 'usable' in the sense that authoring adaptive behavior is not needlessly complex. Authoring can be complex, because adaptive behavior can be complex. However, GALE allows authors to inherit adaptive behavior defined elsewhere. Not only can the content be distributed over the Internet, the domain and adaptation model can be distributed as well, allowing for 'open model adaptation'.

Performance matters. This is not something that should be easily disregarded by engineers of adaptive hypermedia systems. Readers tend to avoid slow web sites. The acceptance of adaptive sites may be hampered by excessive latency caused by slow adaptation engines. Furthermore, the Internet allows potentially many concurrent readers. Poor performance will likely get worse when more people are reading. GALE has great performance, as you can experience from reading this thesis on a server that not only simultaneously delivers several adaptive applications to many different users but also serves the Information Systems Group's website, Learning Management System and other applications.

The existence of a flexible generic adaptive hypermedia system that performs well offers a technological basis for further research into the nature and usefulness of adaptive hypermedia.

Adaptive thesis

The body of this thesis is presented as an adaptive hypertext. It can be found online at this address:

http://gale.win.tue.nl/thesis

Though reading the online thesis requires no setup for the reader and is by far the easiest method, we provide the same material on a CD included with this document. This ensures that you always have access to an unaltered version of the thesis. Setting up an environment to read the version on the CD requires several steps that depend on the operating system used. The CD contains a document detailing these steps called setup.html.

The CD also contains binary and source versions of the software produced within the context of this thesis, namely GALE. Instructions on building GALE from source can be found in build.html.

When accessing the adaptive thesis (on-line or off-line) it provides an option to indicate you are an ‘expert’. When you activate this option all content adaptation and adaptive navigation support is turned off, resulting in a complete and completely static presentation as required by the Doctorate Board (College voor Promoties) of the Eindhoven University of Technology.
Summary

Hypertext has proven its value in many applications, including Technology-Enhanced Learning. Traditional paper textbooks present material in a single order, which is supposedly the best order for studying. In a hypertext, cross-references or links allow more freedom to navigate the material. An adaptive hypertext takes information about the reader into account when presenting the material including the cross-references. Actions taken by the reader (e.g. following a link) can update the information on the reader, hence change the adaptation.

Over the years many different adaptive hypermedia systems have been developed. These systems vary in the way the user is modeled and how that can affect the actual presentation. They are, for the most part, able to perform particular types of adaptation well, but are not compatible with each other. Furthermore, when compared to a traditional web server, an adaptive web server has more work to do. This has often led to performance issues, making the adoption of adaptive hypertext less likely.

GALE (Generic Adaptation Language and Engine) is the result of the ongoing research effort to create an efficient generic adaptive hypermedia system (AHS) that is simple to use yet powerful in expressing adaptation. It is able to emulate most of the adaptation found in other AHS. The performance of GALE allows for a browsing experience similar to that of a normal web server. Adaptive behavior can be inherited allowing different types of authors, both beginner and advanced, to use adaptivity. The description that constitutes the adaptive behavior can be distributed over multiple computers and maintained by multiple authors. The modular structure of GALE allows extensions, integration with other systems and the various components can be distributed to run on different computers.

This thesis describes the historical background of GALE and motivates its design decisions. It shows how lessons learned from its predecessor AHA! have been implemented. The current state of GALE is described to demonstrate its expressive power. As proof of the pudding this thesis is served adaptively by GALE.

Samenvatting

Hypertekst is van toenemende waarde gebleken bij veel toepassingen, waaronder Technology-Enhanced Learning. In een papieren leerboek wordt het materiaal in één volgorde gepresenteerd. De auteur bepaalt een volgorde die geschikt is voor studie. In een hypertext leiden kruisverwijzingen of links tot meer navigatie-vrijheid. Een adaptieve hypertekst houdt rekening met informatie over de lezer wanneer het materiaal en de links worden gepresenteerd. Acties door de lezer (bv. het volgen van een link) kunnen invloed hebben op de informatie die het systeem over de lezer heeft, met als gevolg een verandering in de adaptatie.

Door de jaren heen zijn veel verschillende adaptieve hypermedia systemen ontwikkeld. Deze systemen verschillen in de manier waarop de gebruiker wordt gemodelleerd en hoe dit de presentatie van materiaal beïnvloedt. De meesten zijn goed in staat gebleken bepaalde typen adaptatie uit te voeren, maar ze zijn niet met elkaar te verenigen. Bovendien heeft een adaptieve web server meer werk te doen dan een normale web server. Dit heeft vaak geleid tot prestatie-problemen, waardoor minder vaak wordt gekozen voor adaptieve hypertekst.

GALE (Generic Adaptation Language and Engine) is het resultaat van onderzoek dat gericht is op het maken van een efficiënt generiek systeem voor adaptieve hypermedia dat simpel is om te gebruiken, maar krachtig in het uitdrukken van adaptatie. Het kan vrijwel alle adaptatie die aangetroffen wordt in andere adaptieve systemen emuleren. GALE's prestaties maken het mogelijk om adaptieve documenten met eenzelfde snelheid te bekijken als standaard hypertekst. Om adaptiviteit voor zoveel mogelijk auteurs binnen handbereik te brengen, zijn er mogelijkheden om het adaptieve gedrag van andere documenten te 'erver'. De beschrijving die het adaptieve gedrag van een document bepaalt kan hierdoor over meerdere computers worden verspreid en door meerdere auteurs worden onderhouden. GALE bestaat uit verschillende modules. Hierdoor is het eenvoudig uit te breiden, te integreren in andere systemen, en de verschillende modules kunnen
verspreid worden over meerdere computers en als één geheel samenwerken.

Deze documentatie bij het proefontwerp bespreekt de historische achtergrond van GALE en motiveert enkele ontwerp beslissingen. De voorganger van GALE, AHA!, heeft een grote invloed gehad op het ontwerp. Verder wordt de huidige werking en het gebruik van GALE uitgelegd om duidelijk te maken welke adaptiviteit mogelijk is. Om de waarde van adaptieve hypertekst en de prestaties van GALE als generiek platform te onderstrepen, wordt deze documentatie als een adaptieve hypertekst door GALE aangeboden.

Acknowledgements

During my Computer Science study I came into contact with the group of people conducting adaptive hypermedia research at the Eindhoven University of Technology. They were looking for someone to implement a new generic system for adaptive hypermedia. This appeared to be a suitable topic for an internship and I have been interested in adaptive hypermedia ever since.

First and foremost I would like to thank my promoter Paul De Bra for being a great supervisor and mentor, for our regular discussions on adaptive hypermedia (as many ideas described in this thesis originate from those) and for allowing me to discover the merits of writing documentation (as opposed to only designing and implementing software) at my own pace. This thesis and underlying software would not exist without him.

I would like to express my gratitude to the members of the reading committee: Hans Cuypers, Nicola Henze, Mykola Pechenizkiy and Vincent Wade, for their many comments that helped improve the quality of this thesis. In particular, I thank Vincent for not only commenting on the contents but also on the usability of the adaptive hypertext.

The work presented in this thesis took place within the scope of many projects, specifically the AHA! project and GRAPPLE. I would like to thank Stichting NLnet and the European Commission for their financial support, the many partners in the GRAPPLE project for their contributions that helped shape GALE and the Eindhoven University of Technology for being a flexible organization where I was able to perform most of my research part-time.

In the course of the past ten years I met many people working in the area of adaptive hypermedia who have affected my work. Many thanks go to Natasha Stash for her programming on AHA! and her enthusiasm for adaptive hypermedia, to Alexandra Cristea for her ideas on languages to describe adaptive behavior and to Alexandros Paramythis for his great help and expertise in programming with Java, which motivated me to use software libraries whenever possible. I
would like to thank Riet van Buul and Ine van der Ligt for their great administrative skills and support.

I owe a debt of gratitude to my parents, Martin and Sibilla Smits, who have always supported me. I would not have finished my Computer Science study if it wasn’t for my father’s persistent effort to motivate me. Finally I would like to give special thanks to my wife Jenny who had to miss me more than usual these past few months.

David Smits
Landgraaf, February 2012

---

Curriculum Vitae

David Smits was born on July 15, 1980, in Brunssum (The Netherlands). After graduating from the Romboutscollege in Brunssum he started his study in Computer Science at the Eindhoven University of Technology in 1998 where he obtained an M.Sc. degree in 2004.

He spent the next six years working part-time at the Eindhoven University of Technology in various research projects on building a generic adaptive hypermedia system. The ALS and GRAPPLE projects required cooperation with various partners, both commercial companies as well as other universities. In 2010 he started work on his PhD thesis ‘Towards a Generic Distributed Adaptive Hypermedia Environment’ under supervision of Paul De Bra and Mykola Pechenizkiy.

Writing computer programs has been his hobby ever since a classmate explained BASIC when he was 10 years old. In 1996 he won the Dutch Olympiad in Informatics (second round), in 1998 he received a scholarship from the Eindhoven Technical University and in 1999 he was awarded the ‘Civi Aanmoedigingsprijs’ for excellent results on the propaedeutic exam in the field of Computer Science.