UNIVERSITY OF OSLO
Department of informatics

# The Use of Open Source and Open Standards in Web Content Management Systems

Master thesis
60 credits

Thomas Ferris Nicolaisen

2. May 2006

# Preface

The Use of Open Source and Open Standards in Web Content Management Systems

Keywords: web, content management, open source, standards

## Abstract

The World Wide Web is drowning with too much content. Stagnant web-sites, dead hyperlinks, inconsistent web-design and chaotic site-maps are all symptoms of a polluted Web where valuable content is hard to find. Web content management (WCM) systems have become an increasingly popular solution to these problems. In fact, these systems are so high in demand that competitive vendors seek to lock their users to their proprietary solutions and standards. An anti-reaction to this trend is the range of open source solutions appearing to relieve the web content pressure, as well as an emerging suite of open standards specifying how web content can be transported and stored.

By developing WCM systems, both inside a commercial company, and by participating in an open source project, we have disclosed the relations between web content management, open standards and open source software.

The results include how certain requirements of WCM systems are influenced by open source environments and the use of open standards, as well as the implications such environments have for developers.

## Acknowledgments

The series of events which have lead me to delivering this thesis are not entirely of my own orchestration. The composition of this document has been largely influenced by my friends, colleagues and family. I wish to emphasize the contribution from these individuals in particular.

I wish to thank Lars Brede Grøndahl for giving me the chance to work with Primetime, and Ole Christian Rynning, the breadth of his technical expertise saved the project countless times.

Without the Magnolia community this thesis would probably be half as thick as it is now. Thanks for providing me with an excellent research subject, and I especially want to thank Boris Kraft for his input and feedback.

I also want to thank my mother and my brother, for filling in on my lacking lingual skills.

To the inhabitants of Lekestue, especially Thommy Bommen, thanks for our countless discussions and your valuable opinions.

And finally I'd like to thank Aurelie Aurilla Arntzen for tolerating my endless irresolution and fluctuating ideas, for her confidence and advice, and for believing in me when I hardly did so myself.

## How to read this document

*Chapter 1* is an introduction to the concepts and context of this thesis. It briefly covers the context, question and motivation and background for the thesis, and sums up the main results of the research.

*Chapter 2* presents the methodology which has been used. It explains the research question and how it has been answered, elaborating on what approach has been taken, which frameworks have been used and which methods have been followed.

*Chapter 3* aims to explain the full domain of the context. It retraces the history and concepts of information systems related to web content management, describing the concepts of data, information and content, and how the management of these units have evolved. The context is narrowed down to how content management can be integrated the World Wide Web, and the definition of a web content management system (WCMS). The concepts of open source and open standards are explained. A brief overview of the state of art today is provided with a selection of which vendors, products, open standards and open source environments exist.

The second part of the third chapter presents a set of functional requirements of web content management systems, as well as the two non-functional requirements, *costs* and *extensibility*.

*Chapter 4* presents two possible solutions to the web content management challenges. The two implementations are compared step by step as they are run through the requirements.

Differences in performance on each requirement are explained and subsequently discussed to in *Chapter 5* to find the relation between the implementations, open source and open standards.

The *final chapter* repeats the main conclusions and discoveries. Suggestions for future research and improvement are made.

# Table of Contents

## Tables

## Figures

# 1 Introduction

Information systems are developed and adapted to fit the way humans manage and use information. As the focus on information oriented business increases, so does the number of variations on computer based information systems.

This increase has been made possible and pushed forward by a series of technological revolutions during the last few decades. These revolutions include the rise of the Internet (Hanseth, 2001), the success of the World Wide Web (Berners-Lee, 1999), the availability of personal computers and server performance, more recently the circulation of mobile devices and the distribution of broadband (De Argaez, 2003).

As storage space has grown, and network band-width has widened, the mass of digital information has exploded, both internally on intranets, and on the Internet. Users of the Internet have been most significantly affected by the increase in e-mail traffic and the amount of documents and pages available on the World Wide Web.

Websites have grown out of proportion, and it is not enough to simply deliver information any more. Websites must be easy to navigate and search. Users want personalized results, adapted, translated or shaped into their information reading device of choice, be it a personal computer, mobile phone or PDA. Content managers want more usable editors and workflow systems. To keep web-sites from stagnating, online documents and web-pages should be easy to create, update and archive.

These demands have resulted in a new member of the information system family, the web content management system (WCMS).

Ten years ago, few web-sites shared the same WCMS. This was due to the tendency of developing web content management solutions in-house. As such development is expensive the reaction to this trend was a supply and demand for pre-built WCM systems. Many commercial shelf-ware products, as well as a range of open source alternatives appeared.

Eventually these were followed by suggestions of open standards which specified how these systems could integrate with each other and the Web as a whole.

***Research Question***

This thesis asks what relations exist between WCMS development, open source software and open standards. We want to identify what implications open source development has for a WCMS, and what implications exist for a WCMS using open standards.

These questions are answered through an exploration of the field and a selection of literature reviews regarding the still limited WCM theory. The exploration shows that the academic research surrounding the field is either tied up in developing new solutions or reviewing existing large-scale proprietary systems. The theoretical fields have so far ignored the emergence of open source products into the WCMS software industry.

With the goal of gaining insight into WCM systems, I performed two experimental projects with two different WCMS providers. The first project's goal was to create a web-shop module for *Primetime Portal*. The second project was was to create an equal module for *Magnolia*.

Primetime is a Norwegian company that has been developing and creating web solutions since 1998. They have developed a WCMS by the name of Primetime Portal. It is currently in use at several medium sized Norwegian companies, powering several thousand web-pages.

Magnolia is a competing product of Primetime Portal. It shares some of the technological and architectural workings with its adversary, but the similarities end there. Magnolia is an open source project, developed by a collaborative community surrounding the product which every developer in the world is free to join and use.

These two products undergo an extensive comparison to disclose the relations between WCM systems, open source and open standards. This is opposed the closed proprietary Primetime Portal and its bypass of open standards. On the grounds of this comparison, lines are drawn as to which requirements are met by the suggested solutions, and whether the use of open source and open standards were of any importance in this. It will be suggested that open source solutions are ahead of proprietary products in most areas from a developer's perspective, but there are also some caveats. These are discussed and conclusions are made on which requirements are more satisfied by being implemented with open standards in open environments, and why this is the case.

# 2 Methodology

Too many WCMS evaluations only superficial reviews of how the systems actually perform when put to use (Raible, 2005), (Smith, 2005), (Shreves, 2006). A company auditing different vendors will have a tendency to go for the product which can show the visually most impressive performance in a ten minute demonstration. While the usability of the product is of course important, this does not disclose how it performs throughout the entire software life-cycle of acquisition, deployment, extensive use and extension.

When the methodology for this thesis was selected, it was essential that the research question was answered, and that research produced an advantage to the partnering company Primetime. The latter goal was liberated by the fact that Primetime planned to explore the concepts of open source software, either by developing their own line of products through this business model, or by contributing to, and making use of existing open source products.

Both goals were achieved by implementing one adaptation of each system, at the same time using the insight gained to produce knowledge which can be beneficial to software developers and web content management theorists.

## 2.1 Approach

This thesis has not undergone an empirical study of what WCM systems exist today. It is not a quantitative exploration of which web content management systems are open source software, nor is it a review of which open standards exist for such systems, although resources to find such reviews are provided within this thesis.

### Action Research

The results have been produced by a combination of Action Research (Dick, 2000) and a framework of WCMS requirements I have developed for this purpose.

While larger works of research are inclined towards performing quantitative research and extensive information acquisition on user feedback, this process has focused on experimenting and extensive development with the compared solutions. The comparison has been made mainly from my perspective as a software developer.

The reason for doing so is the practical nature tied to the research question, and the approach which has been made in experimentation by development. It is also evident that many of the discovered WCMS requirements are indeed part of the developer's concerns.

Conventional research produces objective results by studying cases without interfering. Action research on the other hand, is based on producing change. It is suitable to use this approach when researchers acknowledge that the research will have an effect on the case. The researchers are so involved that it is evident that their participation will influence the politics and implementations of the experiment.

I used the Action Research approach in an effort to lower the barrier between software development and computer scientific research. This gave me a chance to participate in the actual development of the case, combining theory, practice and research into the same thesis.

Action research is an adequately rigorous approach for performing research, but it reduces replicability in gaining responsiveness. It also sacrifices global relevance for local relevance, but I hope to be able to draw some general conclusions into the field of web content management nonetheless.

### *Dialectics and Soft Systems*

Most Action Research methods include iterations of planning, acting, and reflection. The methodology used here comes quite close to the method of Soft Systems (Patching, 1990), applying dialectics. I first present the the ideal solution as a set of requirements. This is the first dialectic. I then describe two iterations, each appending a new dialectic, suggesting a real solution to meet the requirements of the first.

The Soft Systems method urges researcher to ally with actors in the research domain, be it research subject or researching colleagues. These allies can be called *clients*. I first partnered with the web technology company Primetime. The partnership included me working as a developer three days each week, assisting in several aspects of the daily Primetime tasks of hosting, development and support.

The second client was the Magnolia project. I "acquired" this client by literally marching into the project as it is open to any developer who wishes to participate. Informant collection was done by subscribing to the mailing lists, contributing to the Magnolia Wiki, and telephone calls to the Magnolia project leads in Basel, Switzerland.

## 2.2 Timeline

It can contrary that a study or research project is a chronological process, evolving and changing as it proceeds. It was not without friction that a two year long research project was compressed into this thesis. To give the reader an understanding of how this thesis was created  over last two years, I briefly retrace the process of events as illustrated in Figure 1.

Two years ago I started looking for a field of research. What began as an interest in quality assurance systems developed into an interest in knowledge management systems, which again was replaced with a fascination for knowledge portals, the most well-known brand of such systems at that time.

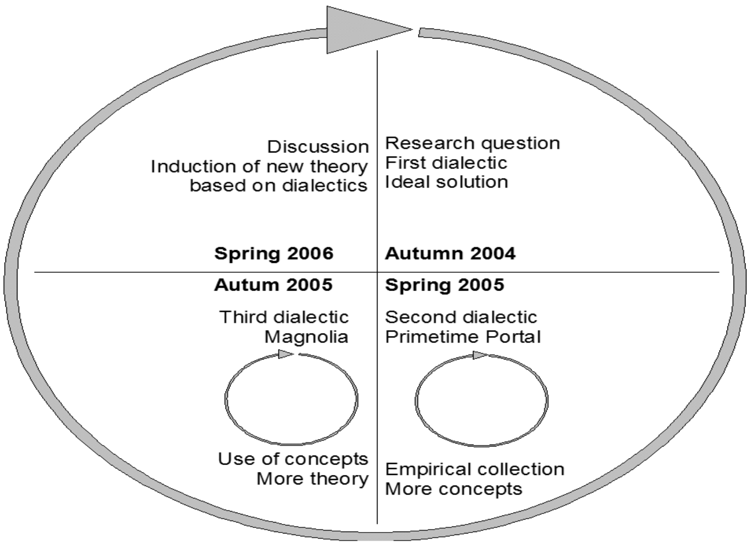


*Figure 1: Timeline*

Around the same period, I was hired by Primetime to assist in the development of the newest version of their WCMS, Primetime Portal. I am a practitioner by nature, and I found a way to combine my development effort at Primetime with the research done as part of my thesis. The reason for doing so was that it allowed me to do something useful, at the same time finding the answers to my research question in a very effective manner.

Together with two other Primetime developers, I developed the model for a new way of storing content, only to discover that a similar model had already been specified in the Java Specification Request (JSR) 170, the Java Content Repository (JCR). We considered doing our own implementation of the JCR but realized that there was already an existing implementation which performance surpassed any functionality our implementation had hopes of achieving within the limited time scope. The implementation was done by an open source project called Apache Jackrabbit. The open source license of this project allowed us to freely re-use the implementation in our own solution.

We then proceeded to plan how we could implement our new WCMS based on the JCR. Again, we discovered that there were existing open source implementations of the systems we were planning to build. This time it was a WCMS called Magnolia. We did consider other projects as well, but none were as compliant, standard abiding and developer friendly as Magnolia.

Today, two years after the thesis was initiated, I have amassed a large collection of WCM theory, which in turn I have forged into a set of WCMS requirements. I have developed one adaption of Primetime Portal, and a similar adaption of Magnolia. Based on this I have put the two solutions through the requirement framework to measure their performance against each other, discovering the advantages of open source and open standards.

On a side note I would like to point out that I used a weblog as an online research tool (Mortensen, 2002). While it lacks structure and rigorousness of this thesis, the blog is still chronological through time, and in a way, it represents the research in a more honest way. It also performs the role of a dynamic research tool, as updated resources are available through my blogroll and linkroll[1].

[1] http://tfnico.blogspot.com

# 3 Web Content Management

This chapter is an exploration of context around the topics treated in this thesis. It studies the need for web content management, what has been done to satisfy this need, and what requirements still remain to be met by WCM systems.

## 3.1 State of the Art

Web content management has been well hyped since the beginning of this millennium (Yankee, 2001), (Forrester, 2001). Like in any hype, the original business idea or re-invention has been flooded by a wave of evangelizing consultancy services, followed by a wide range of implementations to satisfy the sudden demand. According to a rough survey, there are close to 2000 different products that claim to provide content management today (Doyle, 2005). The flurry of products and confusion surrounding the content management hype is then sought by interest organizations to be stabilized to minimize investment risks.

### 3.1.1 Web Content Management Definitions

Before one can properly define the particular kind of information system referred to as the web content management system, one needs to define *content* itself, and separate it from *data* and *information*.

The four ambiguous concepts which are regularly applied in the theory and practice of information systems are *data*, *information, content* and *knowledge.* The context of this thesis is constrained to concepts which can be concretely handled by an information system. The definition of *knowledge* is left out to focus on the other three technical terms.

These terms have various meanings, and are potential candidates for extensive ontological discussion. Note that these terms are to be used in the context of software, not philosophy. To avoid confusion, the meanings of these terms as used in this paper are defined in the following paragraphs.

### *Data*

This is the basic unit of digital representation which can be used to construct information and content. Data is a raw and granular value. It does not inherently have any meaning as its meta-data is not self-contained.

Data is a set of symbols, ranging from a numeral value to a string of words, or a large series of encoded symbols that compose a binary value representing sound or picture. Data processing consists of feeding data as input to a program or an algorithm, the output being new data, information or content. For example, calculating the mean of a hundred numerical values into one number is an operation where data is processed, but no meaning is added. If it was given that this figure is the average temperature for the last three months, it could have been considered information. The data would have had context and meaning, and thereby have become information.

### *Information*

One definition of information is *one or more - well formed units of data with meaning* (Floridi, 2005). The same information can be conveyed with different sets of data. One example is to consider two identical images where one is a Bitmap and the other is a JPEG. They consist of widely different data, but they are still the same information.

Pieces of data combined with meta data form a package of meaning that can be conveyed from one object to another. In the first chapter of his Content Management Bible, Bob Boiko (Boiko, 2005) includes *all the common forms of recorded communication*, as well as presenting Liz Orna's attempt at describing information as *knowledge transformed into a transportable format, visible or audible*. It appears information can be a primitive form of knowledge, or a more advanced composition of raw data. The definition includes all kinds of raw media, video, audio and text alike.

Information can be valued by measuring how much it is used. Information which is stored but never used is worthless.

### *Content*

This is perhaps the term with the vaguest definition. Suggestions include *information put to use* or *information with meaning and context*, but those are quite equal the information definition. The earlier mentioned Bob Boiko mentions that the now disbanded ContextWatch organization defined it as *information shaped for an intended consumer* and *information with a purpose*, and supplies his own definition which is moderately adapted to the one used here.

The definition of *content* used in this thesis is streamlined for how content can be handled by a WCMS. Content is defined as *a collection or subset of information intended for a given audience or non-human consumer with a context of location, period and situation*.

To put it another way, content is *an information composite; ordered, built and delivered*.

### *Content Management*

Content management systems do not stem from academic research and development. They have appeared as a solution to the challenge of handling the massive amount of online content (Gilbane, 2000).

Content management can mean different things depending on what sort of content is to be managed. The most basic life cycle of content is production and consumption. For the producer, the processes of content management include creation, formatting, structuring and integration of content (Burner, 2002). For the consumer, it includes search, export, and display, but can also assist in content creation by providing content feedback, discussion and comments. The sum of these processes are content management. A content management system (CMS) is a suite of tools designed to assist and support these processes (Lin, 2004), (Ashley, 2003). Data, information and content are the building blocks of content management.

A conventional and practical perspective is to say that a content management system (CMS) is a piece of software responsible for taking care of all the digital documents and files in an organization. The functionality of such a system includes document repository control, the company's digital library.

Such a system is a complex implementation depending on whether it includes features like access control, product management, content versioning, import/export, workflow and search functionality. There is even a markup-language under construction for describing and classifying actors in the market (Gilbane, 2003).

***Web Content Management***

As pointed out earlier, the explosion of digital information has been most significant on the World Wide Web. To manage this mass of online content and use, a new breed of information systems has evolved; the Web Content Management System. The responsibility of such a system is similar to that of the CMS, but it is limited to content which consumption is done on the Web.

### 3.1.2 Web Content Challenges

The concept of content in itself seeks to solve the challenges by delivering the *right* content. This goal is not easily reached due to the following conditions.

***Content is not maneuverable***

The main problem with information is that there is too much of it (Goodwin, 2002). There are too many web-pages with too many attached documents (McGovern, 2006b). A company can invest resources into sustaining a site map and a navigation tree menu, but if these are constructed manually, and not generated from the content structure automatically, these navigational methods will stagnate and become more of a nuisance than helpful tools (McGovern, 2006a). Navigating by search is a great shortcut to make all content available, but searching the right way is easier said than done (Belam, 2006) and a search-engine can not substitute conventional site navigation.

***Content is useless***

Stagnated web-sites quickly grow dead links which are references to other web-pages that have been moved or deleted. There might be many pages and documents in existence which are not hyper-linked at all, and thereby will never be accessed. As defined earlier on, content which is not accessed and used has no value. Maintaining value-less content takes up resources which the content managers could have spent on more useful parts of the web-site. It also confuses the visitor by polluting the web-site, making it harder to find the useful content.

### *Content is not automatically accessible*

Two elements by which one can interpret a language are syntax (grammar) and semantics (meaning). A computer interpreting the content of a web-page first checks the syntax by parsing the page and checking whether the markup language is valid. If the syntax is incorrect, the parsing is likely to break depending on the fault-tolerance of the parser. Although incorrect use of markup causes annoyance among web developers, the main issue accessing and reusing web content is lack of semantics. A computer can automatically access a web-page and read it, but it can not decide which paragraph is the title of an embedded article, which is the abstract text and which is the main text of the article unless the semantic standard is enabled in both the web-page and in the program reading it.

Mixing content and design also reduces accessibility. A computer can not decide whether a table is used to control the layout of a page, or if the table has semantic value.

### *Content is not structured*

This grievance is tightly connected to the one above, though it is more apparent in traditional content management. Web content has the advantage of dealing mostly with HTML, which despite its criticism is still a transparent text-based standard based on the more reliable XML. This transparency is lacking in binary files, such as multimedia assets and proprietary formats such as Microsoft Office documents and PDF-files (Martins, 2004).

### *Content has no meta information*

There has a been a noteworthy increase in the ability to tag or label various data objects with meta data. Meta tags can be included in the header of a HTML-page, or in the properties of a Word-document. Forcing users into actually using these features manually can prove to be difficult. If the title of a document is "Content Management", it is quite tedious to label the document with meta-data that states that topic is “content management” and similar keywords. A possible solution to the meta-problem lies in automatically tagging content (Staelin, 2004).

#### *Content is not connected*

There is bound to be digital content within the organization which could have been enabled on its web-site. Databases, memos, product catalogs and other documents, which do not violate corporate confidentiality by being made available online, are typical resources which are held back by their isolation from other content. Information systems are too often designed with a single purpose in mind, and it proves difficult to integrate them as services into the web-site. The worst scenario is when the organization has grown dependent on some specific proprietary software or platform which has restrictions on how the content can be accessed.

#### *Design is not consistent*

A company will normally have one graphic profile, or one different profile for each division of the company. The profile includes names, slogans, logos, a color-scheme, text styles, document headings, footers and layout. Periodically, the profile of a company will be changed, and typically all content produced up and until then will be stuck with the old graphical profile. It is expensive to have a clerk go through each HTML-document and change each document manually. As the profile perpetually changes, the company web-site will grow into a confusing mongrel of pages using various outlooks designed throughout the lifetime of the site. As a result, the visitor of the web-site gains little image of the company's identity, and is left with the impression that the company is badly organized.

### 3.1.3 The Evolution of Web Content Management

It is challenging to make a clear distinction that separates WCM systems from similar information systems. To explore this one must understand the possible ways to do web content management. Various architectures of implementation exist. One possible categorization is presented here.

These four levels are a way to divide the physical management of content. In general one can say that the higher use of web content in a company, the higher level its WCMS implementation should be. The separation is historical and drawn from my personal experience with web development through the last decade, therefore the evolutionary approach.

### *Static files on a web-server*

The most basic strategy is to compose static HTML files and transfer these to a web server capable of serving such files to clients connecting to the web-site as illustrated in Figure 2. It is possible to apply styles to the pages, for example with the help of cascading style-sheets (CSS).

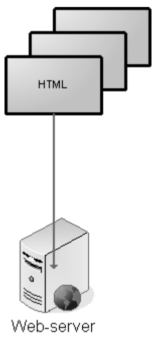


*Figure 2: Static files on a web-server*

### *Content wrapped in templates*

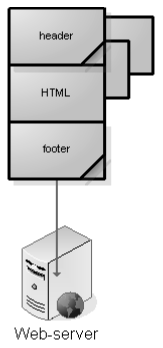


*Figure 3: Content wrapped in templates*

The next level of content management is attained when the editor wishes to re-use the design of the web-site by dynamically including content into a frame of finished design, or a template. The content is typically contained in a text file the dynamic page engine can read, illustrated in Figure 3. Examples of technology capable of this are Server-Side Includes (SSI), Simple Common Gateway Interfaces (CGI) (Dudek, 2003) and XML-documents using XLST transformations[2] (Weitzman, 2002). The HTML standard also has a command called *frames* to include nested web-pages, although professional web designers and developers frown upon the use of this deprecated function (Nielsen, 1996).

### *Dynamically generated content*

More complexity arrives as the re-use of templates is pushed further, having a template dynamically selecting content source based on a dynamic parameter. This is not possible with SSI as you have to provide each separate content page with its own physical HTML file. This

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

means two files for any page on the web-site, one with content, another with design. Many find this to be too cumbersome and end up putting both files inside one, thereby mixing content and design. If a dynamic parameter is possible, as is the case with scripting languages such as PHP (a recursive acronym), Active Server Pages (ASP) or JavaServer Pages (JSP), one can have the template select and read the content file conditionally, thereby removing the need for its own HTML file (Challenger, 2005). This is illustrated in Figure 4.

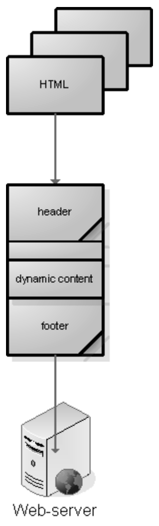


*Figure 4: Dynamically generated content*

***Content stored in a repository***

The next step is to remove the content files to replace them with something more scalable. Native files have many disadvantages: they are not versionable, backup-routines require mirrored copies, search is not easy, binary files like picture and video can not be wrapped with meta data, there is no fitting access control and the possibilities for collaboration is limited. Instead the content is put inside some kind of repository, most likely a database, illustrated in Figure 5. Management of the content is subsequently handled by middle-ware that replace the programming interface of the file system.

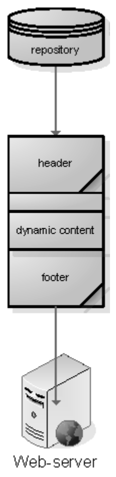


*Figure 5: Content from repository*

A system developer will recognize this three-level architecture of the Model-View-Control (MVC) pattern (Reenskaug, 1978). The model consists of the content in the database, the view-layer is provided by templates, and control is implemented in the middle-ware. The MVC is a pattern that offers a separation of concerns in the WCMS.

***The next level***

It is possible to invent further levels of content management, but any present form of WCMS will most likely apply some variation of the last level. Future levels might include technologies focusing on content integration and service orientation with the use of web services and mash-up principles (First Author, 2006). Another direction in improving performance is distributed CMS networks (Voras, 2005), (Canfora, 2002).

### 3.1.4 Stand-Alone Web Content Management System

Many organizations have intranets on which they perform their content management duties. It is natural to propose that the WCMS integrates with the CMS. Parts of the content which should be exposed on the Web already exists somewhere in the CMS, perhaps on the intranet or on a central file server.

It is natural to believe that the best solution is to invest in a total solution where a CMS includes the WCMS by displaying the content with a Web interface. The case for choosing an isolated or singular standalone WCMS is explained below.

When selecting a system to control their web-site, decision makers are tempted to invest in enterprise solutions. These solutions promise to solve many of the corporate IT-problems with a single centralized silver bullet system. However, the projects where these solutions are selected, implemented and deployed often fail miserably, taking too long to complete. If they ever achieve nominal use, the requirements have changed and the system no longer satisfies the expectations of corporate presence on the World Wide Web (Robertson, 2006).

One way to avoid this pitfall is to build an internal lightweight WCMS, or to invest in an off-the-shelf product. There is still an understood need for such enterprise solutions in large corporations, but such systems are outside the scope of this thesis.

For smaller organizations it is a viable option to leave web content management to a standalone system which is streamlined and specialized for the task.

### 3.1.5 The Differences between a CMS and a WCMS

A CMS and a WCMS have some traits in common. They contain some of the same content, like company and product information, and they might have similar content delivery methods. A CMS can be used to control the web-site. The company can make the knowledge base in the

Intranet available online for allowing customers to troubleshoot problems themselves (Pelz-Sharp, 2006).

A WCMS can either be implemented as a front-end to the company's CMS, or as a stand-alone application. Since many companies have no suitable CMS in place, or their CMS lack a proper web front-end, the latter solution is likely the case.

If the web-site has a user name/password sign-on for employees, there is technically an “intranet” on the WCMS. This access control creates many possibilities for the system. As soon as the identity of an employee or member can be verified online, several normal content management processes can be performed inside the WCMS. The key advantage of doing content management online is portability. The users can access and modify content from anywhere in the world, as long as they have an Internet connection.

### 3.1.6 Alternatives to Web Content Management Systems

To further explain web content management, one can consider what other web content tools and management systems are used today, and what separates these from full WCM systems (Byrne, 2001), (Junco, 2004).

The definitions in use are not clear, and some vendors flag functionality which goes beyond their product. To avoid confusion, these are some of the product families which most often are mixed with the WCMS.

#### *File system*

There are various servers or directory services that can be set up to store digital documents and expose them to the Web with the use of a web-server. Even though many of them store content and perform similar tasks to the WCMS, these systems are not complete content management systems. However, file systems form an architectural basis for physical storage in several WCMS implementations.

### *Weblog*

Perhaps the fastest growing channel for content creation is the weblog, more commonly referred to as 'blog'. Weblog systems make it possible for authors in lack of technical skills to publish online content. Recent years have seen an explosion of 'bloggers' appearing (Blood, 2000), and some believe that this form of publishing will continue to grow at such a rate that it eventually will replace communication lines like e-mail and online forums. In spite of its success, the weblog is still a far too simple protocol to be considered anything more than a possible part of a WCMS.

### *Wiki*

Not nearly as widely known as the weblog, the wiki stems from similar communities of developers using the Web for asynchronous communication and collaboration (Cunningham, 2001). The wiki is a decade old tool allowing developers to create documentation on web-page format, making the documentation easily accessible for viewing and editing. The most famous wiki today is by no doubt Wikipedia (Wikipedia, 2006). Like the weblog, the wiki is too simple a tool to be considered a WCMS. Some have explored the so-called xanalogical potential of wikis (Di Iorio, 2005), so this may very well change in the future.

### *Web editing tools*

Most web-sites are made manually with the use of HTML-editors. While HTML documents can be made with simple text-editors, many users turn to larger web design tools like Macromedia Dreamweaver, Microsoft Frontpage and Adobe GoLive. These products usually feature WYSIWYG-editing[3], web-page previews and even synchronization processes for updating web-pages. Strictly speaking, these tools are mere design-tools. They can be used for creating content, but their main purpose is to control the look and feel of the web-design. This does not constitute content management.

[3] What You See Is What You Get – A term for editing content as it will be displayed, for example editing a Word document as opposed to editing a markup language in its raw format, like HTML or Latex.

### *Enterprise Content Management*

Systems performing enterprise content management (ECM) are typical large scale systems meant for corporations with content throughput of higher magnitude. Some systems like these incorporate their own WCM systems, while other vendors have separated their WCM product from their ECM system (Pelz-Sharp, 2006).

In the industry of content management, the use of this term is largely undetermined. ECM is used for products that do simple content management.

Some WCMS vendors claim their services feature ECM. On the other side of the scale, many lightweight web applications claim to deliver *content management* when they actually are providing what is by most perceived as *web* content management, or perhaps merely weblog or wiki functionality. Regardless, in the terms of this thesis, ECM remains something larger than the WCMS, a system able to process the entire digital content flow of an organization.

### *Digital Asset Management*

These systems are developed to handle advanced kinds of media information, like video and images. The market for this kind of software is expected to grow during the next years due to a larger amount of Internet subscribers capable of streaming multimedia due to wider bandwidth. Many WCMS support media types, especially digital images to some extent, but proper digital asset management systems are stand-alone systems (Porter, 2003).

### *Records Management*

Records management (RM) is also referred to as data warehousing. Large quantities of situational and transactional information require special software developed to store information snippets where the number of articles is counted by the million. Some ECM vendors include RM systems in their enterprise solutions, but a WCMS alone is not necessarily linked with an RM solution.

***Document Management System***

Systems that allow version-management, workflow control, collaboration on documents, digital library and information repositories lie at the core of several content management systems. Some will regard document management systems as software managing scanned digital copies of paper documents. Traditionally these systems were built in-house or proprietary systems, but recently some open source alternatives have started to appear (Gottlieb, 2006). Like RM solutions, these are not essential for web content management.

***Knowledge Management Systems***

Foremost, the principles behind knowledge management (KM) take on a more human approach than traditional software engineering (Davenport, 1998). Even though a knowledge management process will at some point include digital content management, the process as a whole has a nobler end. While the goal of a WCMS is to make content delivery smarter, the knowledge management goal is to make people smarter. Most would agree that a KMS is a suite of processes and tools that includes a variety of computer systems like groupware and generally every kind of management and communication system, including the WCMS.

***Web Portal***

This is perhaps the most difficult category to separate from the WCMS. The term portal is subject to many interpretations. Some considered it to be a personalized start-point on the Web, displaying bookmarks, news and other select content. The Java Community Process' Portlet definition describes portal (or the compilation of Portlets) as a tool for integrating different content sources into one single page (JCP, 2003).

Regardless of its content, a portal is most easily recognized from its panel-like display, including several windows of various content types. It is both possible to say that a portal is part of the WCMS since it can be used for handling online content. On the other hand one can say that the WCMS is one of the many windows in one portal, one WCMS being simply one of the many data sources integrated in the portal.

CMSWatch defines the difference between a WCMS and a portal as the latter being intended for content delivery, while the former is mainly used for content creation. Still it admits that the tasks of the systems overlap, and that open source WCM systems bear portal similarities (Boye, 2006).

***The content landscape***

The landscape of alternatives is summarized in Figure 6. Note that this is just one simple way to consider the range of content management software in the market today. The horizontal axis represents the goal ranging from delivery to the Web to storage. The vertical axis indicates the size or complexity of the system. This is not accurate overview, and many variations of these systems could have been placed differently.

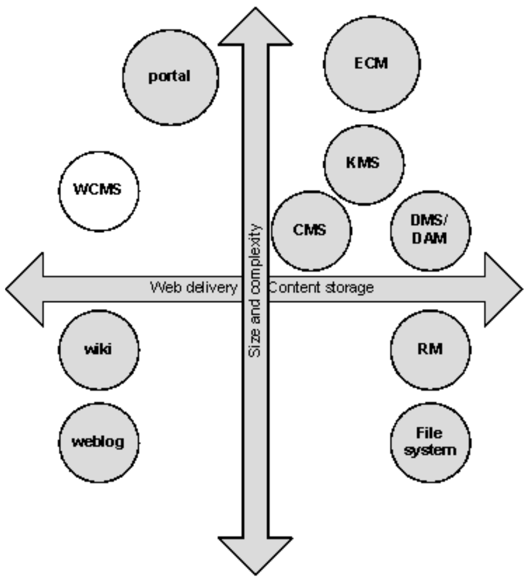


*Figure 6: The Content Landscape*

### 3.1.7 Communities

The WCMS market is so large that it is nearly impossible to get a complete overview of solutions. Attempts to explore this market have already been made by some online communities, and in my opinion the best way to experience the market is by following the lead of these communities. There are also a number of annual conferences specifically intended for content management system vendors, consultants and users.

CMProfessionals[4] is a membership-based community of practice for content management practitioners. Their members are largely responsible for the CMS Forum[5], conferences and the CMS Meta Language, among other resource for CMS evaluation.

[4] http://www.cmprofessionals.org
[5] http://cms-forum.org

The ContentWatch organization has been disbanded, as has the CMS Mailing List[6]. Attempts have been made to revive these, but they have either failed or been absorbed into other communities.

Neighboring communities are less structured and scattered around the Internet. Some camps focus on the relevant theory and practices of intranets, knowledge management and web technologies, and thus provide occasional input to the web content management field.

### 3.1.8 Implementations

Profiling the WCMS as an isolated product has resulted in quite a number of WCMS-products available, some of which are based on an open source business model.

It has been claimed that the birth of the WCMS can be dated back to early summer 1995 (Doyle, 2004). As stated before, this thesis does not aim to review the available alternatives as far better resources are available elsewhere. One starting point is the CMS Community Wiki [7], a knowledge base for Content Management Professionals. It covers many topics of content management as well as several product directories. Another umbrella site for several CMS resources is CMS Review[8].

The consultancy company CMS Works has done a division of WCMS products into six categories (Byrne, 2006). These are (1) Major Enterprise Web Content Management Systems, (2) Upper Tier Companies, (3) Mid-Market Mainstream CMS Packages, (4) Mid-Market Challengers, (5) Hosted Services, (6) Low-Priced Products and finally (7) Open Source Alternatives.

A simplified interpretation of the divisions is presented below.

#### *Large*

The most known vendors in this class include Vignette, Interwoven and Stellent. These systems are for large sized companies, possibly running web-sites across continents,

[6] http://cms-list.org/
[7] http://www.cmswiki.com
[8] http://www.cmsreview.com

generating a large need for dealing with globalization and extreme masses of content. Installation, development and maintenance can usually be measured in hundred thousands or perhaps millions of dollars on an annual basis. It is most unlikely that such companies will run their WCMS totally isolated from their other content systems, rather it will be part of an ECM effort. These systems profile on high level of integration, both between their own proprietary services, as well as across open protocols.

#### *Medium*

Fatwire, Day, Microsoft and IBM's products are members of this class. These vendors supply content management systems to medium sized business. The products suffice to store large masses of content administered by 10-100 content administrators. The software is not shelf-ware, and the WCMS typically requires application servers to contain it. These systems are seldom treated in isolation, and might be incorporated in an ECM solution. The rest of the content process interacts with the online content.

#### *Small*

The market for smaller WCM systems is usually dominated by local and regional vendors. Most Norwegian companies turn to local vendors for implementation since WCM is mostly done in one single language. Small companies have no globalization issues and require an administration interface in their local language. Small WCMS can be sold as shelf-ware, deployable on smaller servers or even desktop machines. These small systems are less likely to interconnect with other information systems in the company's infrastructure. Most will rely on manual file transfer when such interaction is necessary, although some systems have support for protocols which can transfer content from the WCMS to other systems, or the other way around.

#### *Hosted services*

Users who want to entirely outsource the maintenance of their WCMS have several hosted options to choose from. These systems offer low risk as the WCMS costs will result in a static monthly fee plus support expenses. The downside is that these hosted systems are the hardest to customize, as the host will have total control of the system. Also, this WCMS service results in heavy lock-in to the hosting vendor as content and functionality lies here. There is very little chance that the vendor will make an effort to help migrate away from the system, nor give away source code of the functionality with which the content has been enabled.

### *Open Source WCM systems*

The open source WCMS also come in different shapes, and can in a similar fashion spread over several tiers of company sizes (Gottlieb, 2005).

Technical approaches remain much the same for open source and proprietary systems. Although this is gradually changing, the situation is that there is little use of open source in the uppermost tiers of the market (Chawner, 2005). The common feel of open source WCMS projects is that there is great potential, but also reluctance among buyers as such systems come without warranty, and therefore represent risk.

Open source software attracts two kinds of users. The first are small companies with small WCM budgets but skilled in-house developers. There is little wish to invest larger sums in trying out shelf-ware, and management is convinced that the developers can handle the configuration of an open source product. The other kind is companies who wish to comply with open standards, typically governmental offices regulated to do so, or non-profit organizations who do so for principal reasons.

There are many sources for exploring the landscape of open source WCM systems. OSCOM[9] is the international association for Open Source Content Management. It maintains the CMS Matrix for comparing open source products. The matrix is somewhat outdated and only features the most renowned projects. There is OpenSourceCMS[10] that reviews mostly lightweight WCM systems, most of them based on PHP and other scripting languages, and finally Java-Source.net[11], a directory of open source content management systems based on Java.

[9] http://www.oscom.org
[10] http://www.opensourcecms.com/
[11] http://java-source.net/open-source/content-managment-systems

### 3.1.9 Open Source

Having given some indicators to open source WCM systems, the concept should be properly explained. Open source software refers to programs whose source code is made available for use or modification. This means that open source software is in fact free to acquire (Walli, 2005) and change.

A lot of people find this hard to believe, and many presume that such software is produced on a volunteer basis, and therefore lacks quality, security and consistency (Economist, 2006). This is true for a lot of smaller open source projects, but many projects show signs of the opposite (Raymond, 2000), the most famous of these being the operating system GNU/Linux. There is a prominent case for the use of open source (Wheeler, 2005), and larger companies do in fact develop open source software on an economically feasible business model (OSI, 2005).

The revenue can be generated by offering support, customization and plug-ins. Large software companies like IBM and Sun have for the last years been funding, as well as founding, open source projects to ensure that their ideas and standards are established throughout the open software community (IBM, 2005), (Sun, 2006). This thesis will not delve further into the principles and ideas of the open source movement. The interests of WCMS users lie in the risks versus the benefits of the system. It is important to remember that most open source material comes without guarantees and warranty unless support is bought from the vendor or developer, and this is where the cost of "free" software lies.

Open source projects have a tendency to prefer re-use and compatibility over developing their own formats and protocols. Whenever possible they embrace open standards in an effort to receive further adoption from the community. Open standards are of course also adopted by proprietary software developers, but not to the same extent as with the open source alternatives.

The Free Software Foundation (FSF) is persistent in bordering itself from the Open Source community (GNU, 2006). A short summary of the debate is that the methods of the two communities are the same, but the ideals are different. The FSF support the practice of open source of ethical reasons, while the Open Source movement does so for practical reasons.

For the purpose of this thesis it is not the ideal freedom of the software which has implications for developers, but the availability of the source code, the option to modify or extend it and the presence of open standards. The term used within this thesis when talking about open source is compliant to that of the Open Source Definition (OSI, 2001).

### 3.1.10 Open Standards

The relation between open standards and web content management is easy to find, as the Internet itself is based on open standards. The open source relation is similar. The most well known connection between open source software and the Web is by no doubt the Apache web-server. This open source project has been powering the majority of the world's web-sites for many years (Netcraft, 2006).

The openness of the Web attracts open standards and open source projects. A WCMS is a complex piece of software which leaves single developers with much fatigue if they should ever attempt to implement such a system on their own. The culture of the World Wide Web has naturally led such developers together in numerous open source implementations which will be further explored in the next chapters.

A standard is an agreement of two or more parties regarding a product, specification or other. Standards used by web applications are mostly guarded by the Internet Engineering Taskforce (IETF), the World Wide Web Consortium (W3C), Institute of Electrical and Electronics Engineers (IEEE) and International Telecommunications Union (ITU). Examples of successful standards are hypertext markup language (HTML), hypertext transfer protocol (HTTP) and resource description framework (RDF).

System developers can choose either to use existing standards or invent their own. Sometimes not having to follow a standard is easier and quicker than having to fulfill a specification's every need for details, but along the network externalities in the system where other systems interconnect, open standards must be followed (Ciborra, 2000). This applies to the technology used for *transport* or *storage*.

A typical transport technology standard is HTTP, through which all web applications are made accessible.

Storage technology standards are the format in which content is stored or presented. A web-page must output format in HTML, pure text or a standardized binary format like Bitmap pictures or Macromedia's Flash.

Proprietary standards can be open like Adobe's PDF format and Macromedia's Flash file format, or closed like Microsoft Office Word documents and Powerpoint presentations. A proprietary standard can only be changed by its owner. You can make software that reads both open and closed standards, but discovering how the closed standard is built up internally can be difficult, and under certain certain condition, so-called reverse-engineering is considered illegal (LII, 2005).

Microsoft uses a multitude of proprietary standards to enable other vendors to produce software for the Windows platform. Examples are DirectX for graphics and MFC for desktop applications.

Note that even though Microsoft and their Office products are frequently used as examples of proprietary software, they are not the “big bad wolf” regarding use of open standards. Such advanced software can not always suffice for the bureaucratic democracy and slow development of open standards. Microsoft is more and more embracing the use of open standards like WebDAV and SOAP (W3C, 2003) in their newest software. In fact the next version of the Office suite will use zipped XML-files for storage, like OpenOffice has been doing for several years (Microsoft, 2006), (Spangler, 2006).

Research on open standards abounds in information infrastructure research, especially regarding the architecture of the Internet and the Open Systems Interconnection (OSI) effort (Hanseth, 1998), (Hanseth, 2002).

A WCMS will naturally output its content through HTML on a web-site. Internally, however, the implementation may store the content in a home-grown format, for example a relational database with a streamlined scheme following no standard (except the standard of SQL itself). As long as the company uses the WCMS the way it was built to be used, the inside workings of the content repository is not important. The problem arises when the company either wishes to change the output or use of the content, or to replace the WCMS all together. In most organization, this does eventually happen. Requirements change.

How will the content be exported from the old WCMS and imported into the new one? Manually copying the HTML code from each web-page will no doubt be a very tiresome effort. Another alternative is reading content directly from the relational database with an exporter-application. If the WCMS has not supplied one, developing this application could be a large task. And then an application would have to be developed for importing the content into the new WCMS.

The best solution would be if the storage of both WCMS-es utilized a standard content repository, so the content of the old system could simply be dragged-and-dropped into the new one. Unfortunately, today there exists almost as many different content repository implementations as there are content management system vendors.

## 3.2 Requirements

The following section is the core part of the theory which will be used in the next chapter to evaluate the proposed solutions. As mentioned in the methodology chapter, fulfilling these requirements can be considered the ideal solution of web content management to which the other dialectics will be measured against.

As the discussion of this thesis will show, the absolute requirements of a WCMS are impossible to predict. Consequently, *extensibility* is the final and most important requirement. The others are organized into the categories of *technical*, *management*, *globalization*, *content delivery* and *cost* requirements.

### 3.2.1 Technical

Technical requirements are the obligatory basic needs of the environment, hardware and software hosting and maintaining the WCMS.

The successful deployment of a WCMS depends on many information infrastructural circumstances and politics like management priority, user acceptance and technical feasibility. As declared in this chapter, the main requirement of a WCMS is extensibility, and the one who has to make use of this requirement is indeed the developer responsible for deploying and running the WCMS in-house of the intended organization or corporation.

Since Primetime has provided hosting and maintenance to both implementations of the case, the hardware requirements and costs have not been a main issue of the development projects. We have therefore disregarded the still very crucial requirements of *scalability*, *availability* and *security*. When professionally auditing WCMS solutions these requirements *must* be considered. They are only disregarded here because we believe they are related to open source development and open standards in a lesser fashion that these others.

The WCMS may rise or fall by the outcome of these developer tasks.

#### *Deployment*

Developers are responsible for installing the WCMS, not only the first time, but they are also the ones performing redeployment when upgrades are necessary or patches are released from the vendor. If the process is cumbersome, this will happen with a low frequency and lead to a compromised and outdated WCMS. If it is not easy to migrate old content from the old installation to a newer one, the developer will quickly tire of the process and opt to management for choice of a different WCMS,

#### *Integration*

Infrastructural services such as e-mail, user directories and existing services should often be interconnected into the WCMS, and this will perhaps be the largest task the WCMS developer is responsible for, depending on requirements and existing information systems within the organization. Larger ECM solutions often benefit from utilizing strategies of service-oriented architecture (SOA), making it easier to integrate new functionality as web-services into the system.

### *Templates*

Default templates and skins are bound for change after acquiring the system. Company logo and themes must be applied, and the CSS-styles applied by the WCMS may not be of the same patterns as the company's graphical profile.

This is not as much a feature as it is a necessity. A company is often judged by the outlook and consistency of its web-site. While the web designers no longer need to author the content of web-sites, they still need full control of the design. Templates allow designing once, and then applying the same design to whole parts of the site in one action.

Older web design tools have created an inclination towards not using mesh templates, where the template is separated into header, footer, left panel, main column, right column, and so on. More modern web design tools have support for working on such composite page design.

### *Backup*

A WCMS is a complex system, and since this type of software is a fairly modern family of information systems, it is prone to experience bugs and crashes where data loss is a risk. Many technicians would argue that the responsibility of making information backup lies outside the WCMS, but there is still a requirement for the content repository to be backup-able in an automated fashion. A home grown file system or smaller database repository may lack support for such tasks.

### *Monitoring*

Monitoring consists of automatically computing statistics and numbers on server usage and display them to the developer in a readable format. For a web-site this includes keeping track of incoming requests from visitors. If site traffic is not monitored it becomes harder to evaluate the returns of the WCMS, and it will quickly loose its favor from the management which weighs the cost of sustaining against these returns.

#### *Logging*

Logs are the server's output on relevant activities and processes. If logging is not done properly, it becomes hard to trace the source of errors and crashes.

Most web-servers have tools for monitoring the number of visitors. Traffic can be measured in number of “visits” or “hits”, although number of hits can give a very misleading understanding of how much traffic the web-site is experiencing. Number of visits and average visit length is the correct way to report traffic.

### 3.2.2 Management

The person or persons who will be spending the most time on the web-site are no doubt the ones responsible for managing the online content, be it a company clerk, a webmaster or a chief information/content/knowledge officer. If this user does not find the WCMS practical and usable, the content will quickly stagnate, and site traffic drop.

#### *Creation*

For the authors, the most important functionality of the WCMS is the composition of articles. This is where content is assembled. Advanced composition features a WYSIWIG-editor, spell checking, insertion of images and hyperlinks and the ability to create tables.

#### *Publishing*

Publishing is the process of taking the content from the author and making it available online. It should also be possible to later edit published pages, as well as taking them off line, hiding them from public view without deleting them. The last point is actually part of the workflow requirement presented below.

#### *Workflow*

This is a feature of WCMS featuring several authors and perhaps an editorial staff. A web-page or document has status which perhaps only certain individuals are authorized to change, for example the editor accepting an article for publishing. Time-limits are also part of the workflow. One page can be scheduled to go on- or off line at a given point in time.

#### *Administration*

When web-site structure grows complex, there appears a need to administer the larger amounts of content, and which users are privileged to do which actions. User, role or group access rights must be managed. The administration is generally what the content manager is doing besides creating content.

### 3.2.3 Globalization

International companies need multilingual web-sites (Huang, 2001) with **internationalization** and **localization** features.

#### *Internationalization*

This is the concept of having country and language-specific content, essentially having the main content of the web-site translated to one or more languages (Iverson, 2002).

Translation of a WCMS can be divided into two parts. The most important one is how the content itself can be translated by the content managers. The other aspect is the language of the WCMS itself regarding internal interfaces for administration and management.

#### *Localization*

This refers to visual effects based on the visitor's locale, like country specific temperature, time, date and currency formats, one example being how certain countries use the 12-hour AM/PM style to define time, while others use 24-hour notation.

### 3.2.4 Content Delivery

***Syndication***

To increase the availability of content, larger web-sites feature syndication, or off-site publishing. This can be approached by subscribing to receive new pages through e-mail (newsletters), or as the increasingly popular news-feed (RSS).

As an example, many news-sites have offered the option of subscribing via RSS-feeds. By subscribing to these feeds in RSS-readers or news-aggregators, the process of collecting news from these sites is turned from a pull-protocol, actively browsing for content, into a push-protocol where content is pushed to the reader.

This is related to the idea of the Semantic Web (Berners-Lee, 2001), a set of W3C standards created for enabling data sharing across the Web. One version of the RSS format (1.0) is actually a name-space within the Semantic Web's RDF specification.

***Accessibility***

Many developers associate accessibility with the extent on which disabled people can use computers. This could be because they lack motoric skills, or because their hearing or eyesight is impaired. For example, certain keyboard shortcuts would not be accessible for a one-handed person, and color-codes can be hard to read for the weak-sighted.

A more generic understanding of accessibility is the limitations readers have accessing content. These limitations can be lack of mouse or keyboard, small sized screen or lack of colors. Limited devices like mobile phones, PDAs and older computers lack the luxury of heavy graphical user interfaces.

***Search***

The importance of the this requirement is proportional with the size and maneuverability of the web-site. Although a very basic search-engine is sufficient for most sites, it is also possible to implement smarter searches that accord for miss-spelling, try different word ending(s), and use context specific dictionaries. A good search-engine also indexes your online binary files (PDF and Microsoft Office documents for instance).

The intelligence of a search engine increases by the work which is put into configuring it because there are a lot of context related parameters which must be sorted out. The engine must accord with language(s), location of where the searchable information is stored, possibilities for tracking content by URLs with spidering techniques and security. There are many issues which much be situationally decided, like whether hidden files should be search-accessible. Upon installation of the search engine, it will require hours of manual configuration to fit the context. It should be able to monitor the search patterns of the visitor to better tune the searches to yield usable results.

#### *Communication*

A powerful mean to further enable existing content is to give the consumer the opportunity to provide feedback to the web-site. This functionality can come in several shapes, including the ability to add comments to web-pages, participate in online surveys and discuss content in forums or chatting consoles.

If the goal is to make it easier for potential customers to contact the business, one could measure the number of visitors compared to the number of visitors who actually fill in some online contact form.

A way to generate income directly this way is to provide the visitor with the option to buy services through a web-shop. Having this channel makes it quite easy to measure how many sales are generated from the business' web front-end.

Feedback from visitors can collected to help improve the web-site, but some sort of incentive is normally required to tempt any visitors into actually completing such a form. If the web-site is of low value to the visitor, chances are slim that the visitor will aid improving the web-site.

### 3.2.5 Costs

This is perhaps the most important factor for WCMS buyers.

The total cost of an information system is easily displaced as buyers have a tendency to ignore the total lifecycle of the software. A CIO in a small company could explain that she spends zero on web content management since she does it all by herself, but the number of hours she spends updating the web content each week might amount to a significant expense relative to the size of the company.

A WCMS has costs upon acquisition. The software is bought, and additional modules or plug-ins will likely add to the price. It must be tested, deployed and tweaked by developers to fit the company's environment. A web design must be applied to the templates. Users must be instructed on how to use the system. Older content must be imported.

There are maintenance costs to be considered. Content managers receive wages. The WCMS is customized, extended and maintained by developers, adding cost to the investment.

The final step of the lifecycle is migrating away from the WCMS to a newer one, or perhaps the web content is to be absorbed into an enterprise content management system. The content has to exported from the old system and imported into the new system. Finally, the value of the previous investments are nulled as the intellectual capital put into the use of the legacy WCMS is no more.

Depending on the amount web content and the complexity of the software, all these tasks involve considerable costs.

Like in any form of company profiling, there is no immediate return on the investment (ROI). This can lead to an negative process where the WCM division of an organization gets low priority and receives low-funding, the division performs worse web content management, and the web-site returns less revenue. However, many of the WCM systems benefits, like in any IT-investment in general, are intangible and hard to find and measure (Weill & Broadbent, 1998). Intangible benefits of running an advanced WCMS can include a smaller need for WCM-staff. One less full-time employee could quickly make such a large WCMS investment worthwhile.

Measuring all investment down to an economical figure can prove to be an inaccurate measure of a WCM system's success. There may also be other infrastructural business values which should be investigated. A quality web-site is a crucial part of the identity of a large IT-company. All in all, finding the ROI is a complex task which is not the center focus of this research, but it has been explored in many others (Hallikainen, 2002), (Ward, 2003).

### 3.2.6 Extensibility

The final and most important requirement of the WCMS stems from a single principle. It is impossible to predefine all requirements for a WCMS. Each year new concepts, ideas and methods are introduced to the World Wide Web, and web-sites must change the way they deliver content, content managers must change the way they produce content, and developers must change the WCMS to allow the requisite changes.

We propose that extensibility is the most important requirement of WCMS, because they have no definite set of requirements. A WCMS is an abstract information system, and users will not properly realize potential functionality before the WCMS has been put to use. The requirements are indefinite, and more functionality will be demanded as time goes by, and this is why extensibility is such a crucial requirement.

To explore how the WCMS performed through development of the system, a special case was selected as a trial of customization and extensibility.

One of Primetime's customers was in need of a web-shop. It was judged to be beneficial for both customer and developer that the web-shop be developed as part of the WCMS. This special case was selected for reasons including that none of the WCM systems had this functionality beforehand. A web-shop is complex enough to test a wide aspect of the developer's requirements, and it is a natural extension to a WCMS.

The Mesterbrevnemd is an officially appointed interest organization for those qualified to the Norwegian Letter of Master Craftsman Certification. There are approximately 18.000 members (as of December, 2003). The organization distributes effects proving membership, like gold watches, cups and T-shirts, as well as the actual physical letters of members' certification.

The requirements were quite simple and abstract. The customer required a simple web-shop featuring the functionality of adding and removing items from a virtual shopping cart, then afterwards checking out, submitting personal information like name, address, e-mail address and telephone number.

Naturally it had to be possible to modify the actual items, prices, pictures and information in the web-shop from an administration point of view.

For simplicity, it was not a requirement to store transactional data. Each order would be sent to the web-shop managers by e-mail, who would then handle and store the transaction internally as needed.

As a result there was no need to store any information about the visiting purchaser. However, only master craftsmen or their affiliates were allowed to purchase items in the web-shop. The authentication of this was done externally by the *mesterbrev.no* web-site, and was consequently not a requirement to the web-shop.

Implementing extensions and custom functionality requires a different course of action depending on the architecture of the WCMS. If the WCMS is not flexible enough to allow plug-ins or modules that can satisfy the requirements, extending the functionality of the core software may be necessary.

## 3.3 Summary

Web content is constructed from information components for an intended consumer. The production and delivery of this content are performed by web content management systems. Many solutions exists, some of which are open source, others which are proprietary. Some try to comply with open standards for transport and storage, while others develop their own formats. The question this thesis raises is whether there is an interconnection between open standards, open source development and WCM systems. Already it is evident that such a relation exists. For example, the requirement of accessibility is achieved quite directly by presenting the content by open standards such as XHTML and CSS, then leaving it to the browser to display the content in a readable way (Kennedy, 2006). Other requirements are less tangible to treat in such a relation, extensibility and costs being the hardest ones to measure.

To provide an overview of the requirements, they have been retraced in Table 1: Summary of Requirements below.

| ***Requirement*** | ***Keywords*** |
|---|---|
| **Technical** | |
| Deployment | Installation, migration, environment |
| Integration | Infrastructure, architecture, connection |
| Templates | Consistency, graphical profile, re-use, customization |
| Backup | Exporting content, security |
| Monitoring | Site traffic, status, measure returns |
| Logging | Error handling, notification, security |
| **Management** | |
| Creation | Editing, authoring, WYSIWYG |
| Publishing | Public content, drafting |
| Workflow | Content process, roles, responsibilities |
| Administration | User administration, access rights, configuration |
| **Globalization** | |
| Internationalization | Translation, multi-language sites |
| Localization | Locale, format date, time, currency |
| **Content Delivery** | |
| Syndication | Export, XML, E-mail, news-feed |
| Accessibility | Disabled content readers, limited devices |
| Search | Search-engine, intelligent searches, tuning |
| Communication | Visitor feedback, forum, comments, chat |
| **Costs** | |
| **Extensibility** | |

*Table 1: Summary of Requirements*

The next chapter presents two alternative solutions. One of them is an open source project which enables heavy use of open standards. The other is a proprietary system with less dependency on standards. Each of the requirements will be investigated in order to find other relations between the concepts of open source, open standards and web content management.

# 4 Suggesting Implementations

This chapter describes two individual attempts at using Primetime Portal and Magnolia to meet the general requirements of a WCMS as prescribed in the previous chapter. After this requirement evaluation there is a discussion based on the research questions originally raised in the first chapter of this thesis.

The solutions meet the requirements at a varying degree and these are compared in the next chapter. This chapter aims to explain and create an understanding of the suggestions of implementations that have been made.

Each section aims to give the reader an understanding of the system. Their internal architectures are different, and this has implications on how they are to be developed and used. Acquiring this knowledge was done by reading all documentation, extensively exploring, using and changing the code base, as well as getting to know the developer and user communities. It was also done in sessions spent using the WCMS with the customer, letting the customer use it alone, thereafter receiving feedback through several meetings through which further improvements to the system were suggested.

## *4.1 Primetime Portal*

Primetime Portal is a proprietary WCMS which has been under constant development from 2000 till 2005 (Primetime, 2006). I participated in the most recent development as part of the research for this thesis. It is a framework of Java applications containing various modules which take the key roles of typical content management features.

Primetime Portal is at the lower end of the price scale compared to other WCMS-vendors. Depending on the number of modules, use of storage and bandwidth load, the costs can vary greatly between 100€ and 1000€ per month. Primetime Portal is hosted entirely in Primetime's server park, so the customer has no other technical expenses.

It is not really a portal in the conventional sense of the word. It does not feature personalization or interface components like Java Portlets or Microsoft SharePoint's web-parts. Rather it is more of a classical WCMS with services like straight-forward news publishing and mailing-lists.

### *Architecture*

Primetime Portal runs on a set of two servers. The first is a web-server that handles the requests and responses to visitors through Internet connection. The second is a database server behind a firewall. The web-server runs the application container, an Apache Tomcat (see http://tomcat.apache.org), which runs one instance of the Primetime Portal web application for each installation, meaning one for each customer. The database server runs an Oracle database with one database scheme for each Primetime Portal installation.

The web application is developed with Java technology. It features one client side package and a server side package. The client side is a Java Applet that communicates with the server side through Sockets (Sun, 2005). The request/response model used in this communication is developed entirely in-house and the stream-object method renders the flow of communication non-transparent to those without access to the Primetime Portal source code.

The final web-site is coated with HTML and rendered dynamically through the use of JavaServer Pages (JSP)[12]. These JSP templates have unfortunately become entangled with web design and cluttered with scriptlets and JavaScripts due to a lack of enforcing design-content-separation techniques such as style-sheets (CSS). CSS is used, but only for modifying the style of fonts. Layout is done with the use of tables and maps.

### *Use*

To invoke the client, initializing a web content management session, a web author will access a certain address (like www.mysite.com/publishing), enter username and password upon where the Applet is downloaded and run locally on the author's computer. Note that the client computer must have the Java Runtime Environment installed. Other than this there is no need to install any kind of software on the client computer.

[12] This technology is part of Java Enterprise Edition – http://java.sun.com/javaee/

Inside the Applet, the author has access to several panels, each panel offering functionality. Note that custom functionality is not part of the client software, but rather resides within Java library modules in the web application's server side.

Primetime's customers have been using Primetime Portal for 3-5 years, so using the Applet was no great challenge for them. They complain about the old issues with the Applet, which are problems with copying and pasting into the Applet window, as well as the generally aged outlook of the program.

### *Development*

Counting lines of code, Primetime Portal is a sizable project. The core libraries alone reach roughly 30.000 lines of code over 261 Java classes. The code base resides inside a source code management (SCM) repository. When improving or changing an existing module, the code is checked out from the SCM system, modified and committed back to the repository. This procedure is normal, but still important to the development process, allowing versioning the code base and structuring collaboration between developers.

Building the code involves running an Apache Ant[13] script. Each module has its own script, and running the script results in a Java archive (JAR) file being built.

Deploying or redeploying the module consists of moving its JAR-file into the classpath of the web application nesting the Primetime Portal instance for a given customer. Upon doing this the web application has to be restarted, resulting in the customer's web-site being unavailable for roughly three seconds.

There are some issues with this deployment process. There are dependencies between the modules, and changing one module might result in functionality being broken in another module. This can be checked by vigorous testing, but in reality testing is often skipped to save time. There is no suite of unit tests which could have done this step automatically for the developers, as is the procedure for those doing test-driven development (Beck, 2003).

[13] http://ant.apache.org

Another issue is that there is no record kept of which modules are deployed at what version in each Primetime Portal instance. When upgrading one module it is uncertain which other modules have to be updated to their latest version as well.

As described in the architecture of the system, data is stored on the database server. This does of course have implications for the development process. Adding new functionality will on occasion involve change to the database scheme. Databases are static structures, and the more content they contain, the harder they are to change. Usually the only way to add a database scheme is by adding additional tables to the design, not by modifying the existing ones.

Each customer has their own scheme stored in the database, and like with the modules the design of a scheme is not stored in a versioning system. For example, a customer running a very old instance of Primetime Portal might be using a very old scheme with 24 tables. A more modern installation of the WCMS might be using a database scheme with 29 tables. When upgrading the instance, the design of the new scheme might not be compatible with the new one.

In the following section, the requirements of WCM systems are applied to Primetime Portal. These are the same requirements as presented in Table 1: Summary of Requirements on page 39 of this thesis. For each requirement the performance of the WCMS will later on compared to the performance of the open source alternative, Magnolia.

### 4.1.1 Technical

#### *Deployment*

Primetime Portal's deployment server-side is a set of direct operations to the web-server and database. To create a new instance one must first define the database scheme on the Oracle server. This is usually done by copying the scheme of a neighboring instance and takes one minute if one is familiar with Oracle databases.

Afterwards one must add configuration records into a special instance scheme that controls which customers are using different domains and servers. When the database scheme is set up, the application context is manually created within the web-server, compiling and adding the core module JAR-files to the library of the context as one goes along. Finally the web application context can be started, this last step also being a manual one. All in all, setting up the most simple Primetime Portal installation will approximately one working day to set up and test.

#### *Integration*

As far as the integration with the existing IT-services of a customer are concerned, Primetime Portal is a mixed case. A varying amount of the customers' IT-systems are provided by Primetime, and as long as we were hosting the two services which were to be integrated, this was solved by developing custom-made integration module consisting of hacks and “spaghetti code” that patched the services together. Integrating Primetime Portal's services with other IT-systems is a different matter. Some simple XML data export services have been created, but nothing more advanced than this. There has never been an explicit need to integrate services from external providers.

Internally, the existing modules of Primetime Portal use a communication method of very domain specific Java object streams. These are useless when it comes to re-use in integration efforts, and each time integration is made it involves some tedious adaption of these internal data streams.

#### *Templates*

Most of Primetime Portal's templates were developed at a time where many of the newer JSP features were not available. As a result, the templates vary in how they use the custom tags, JavaScript and JSP scriptlets. Adding to the confusion, these scripts are intermingled into the same and singular template files. They are incomprehensible less a longer effort put into understanding them, and only minor modifications can be done without risk.

#### *Backup*

Primetime Portal is tightly coupled with the configured schemes in the database. Since these schemes are poorly documented, deployment of new installations is a complex procedure.

Making backup of the content in Primetime Portal is heavily tied to the backup-procedures of the Oracle database. These procedures will not be explained in detail here, but Oracle does feature a powerful backup-system, running incremental storage of the content with regular snapshots of the database being taken. Should the database crash or become corrupted, a technician can restore the latest snapshot and rebuild the latest changes by leveraging the incremental commits on top of this data.

#### *Monitoring*

Primetime Portal uses an extensive monitoring log system. Nightly processes running on a dedicated server automatically copy the HTTP-traffic logs from the web-server, parse through it and separate what traffic has taken place with a given customer. From this data readable reports are generated and published to the customer's web-site.

#### *Logging*

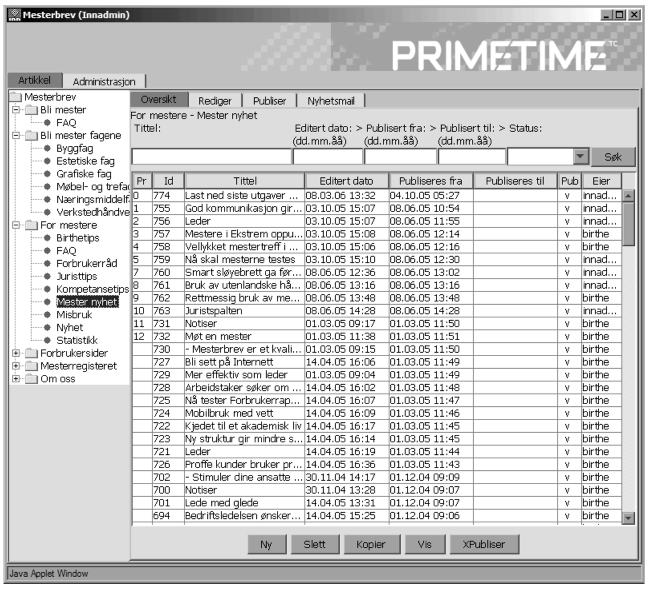
Primetime Portal does not have the same logging framework in order. All messages are output to the system default console, which in effect means that all instances of the WCMS flushes the one and same log file with its messages. To the occasional observer, this output log seems massive, cryptic and nonsensical, and it is in fact meaningless for most of the developer's tasks as well. This log is only periodically observed for monitoring web-server operations.

### 4.1.2 Management

#### *Creation*

When it comes to producing content, Primetime Portal has a more traditional way of creating articles. Upon initializing the client-side Applet the manager can view the entire archive of articles in a single list (see Figure 7: Article management in Primetime Portal). Filters can be applied to the list to make articles easier to find, and the list can be sorted on different criteria such as author and publication date. Articles are edited in a more or less simple WYSIWYG text-editor (see Figure 8: Creating content in Primetime Portal).

There is also a module that allows the content system to store and share documents and media files among its users. It also includes an image manager for using pictures inside published articles.



For mestere - Mester nyhet

Tittel: Editert dato: > (dd.mm.åå) Publisert fra: > (dd.mm.åå) Publisert til: > (dd.mm.åå) Status:

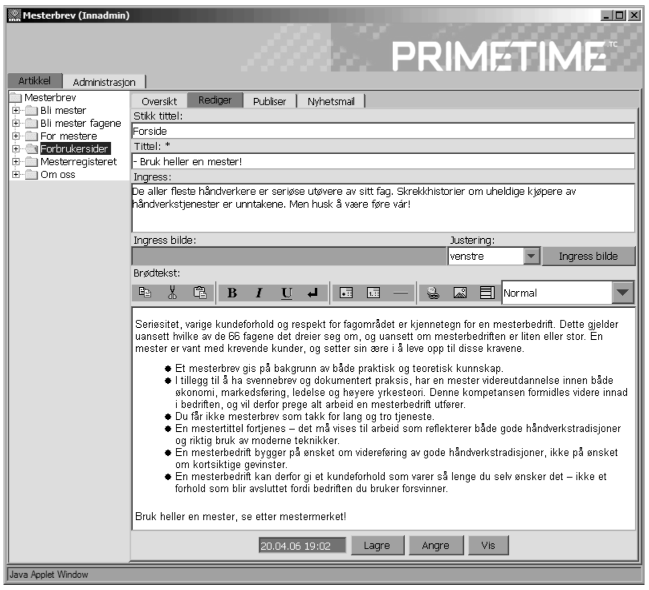
| Pr | Id | Tittel | Editert dato | Publiseres fra | Publiseres til | Pub | Eier |
|---|---|---|---|---|---|---|---|
| 0 | 774 | Last ned siste utgaver ... | 08.03.06 13:32 | 04.10.05 05:27 |  | v | innad... |
| 1 | 755 | God kommunikasjon gir... | 03.10.05 15:07 | 08.06.05 10:54 |  | v | innad... |
| 2 | 756 | Leder | 03.10.05 15:07 | 08.06.05 11:55 |  | v | innad... |
| 3 | 757 | Mestere i Ekstrem oppu... | 03.10.05 15:08 | 08.06.05 12:14 |  | v | birthe |
| 4 | 758 | Vellykket mestertreff i ... | 03.10.05 15:06 | 08.06.05 12:16 |  | v | birthe |
| 5 | 759 | Nå skal mesterne testes | 03.10.05 15:10 | 08.06.05 12:30 |  | v | innad... |
| 7 | 760 | Smart sløyebrett ga før... | 08.06.05 12:36 | 08.06.05 13:02 |  | v | innad... |
| 8 | 761 | Bruk av utenlandske hå... | 08.06.05 13:16 | 08.06.05 13:16 |  | v | innad... |
| 9 | 762 | Rettmessig bruk av me... | 08.06.05 13:48 | 08.06.05 13:48 |  | v | birthe |
| 10 | 763 | Juristspalten | 08.06.05 14:28 | 08.06.05 14:28 |  | v | innad... |
| 11 | 731 | Notiser | 01.03.05 09:17 | 01.03.05 11:50 |  | v | birthe |
| 12 | 732 | Møt en mester | 01.03.05 11:38 | 01.03.05 11:51 |  | v | birthe |
|  | 730 | - Mesterbrev er et kvali... | 01.03.05 09:15 | 01.03.05 11:50 |  | v | birthe |
|  | 727 | Bli sett på Internett | 14.04.05 16:06 | 01.03.05 11:49 |  | v | birthe |
|  | 729 | Mer effektiv som leder | 01.03.05 09:04 | 01.03.05 11:49 |  | v | birthe |
|  | 728 | Arbeidstaker søker om ... | 14.04.05 16:02 | 01.03.05 11:48 |  | v | birthe |
|  | 725 | Nå tester Forbrukerrap... | 14.04.05 16:07 | 01.03.05 11:47 |  | v | birthe |
|  | 724 | Mobilbruk med vett | 14.04.05 16:09 | 01.03.05 11:46 |  | v | birthe |
|  | 722 | Kjedet til et akademisk liv | 14.04.05 16:17 | 01.03.05 11:45 |  | v | birthe |
|  | 723 | Ny struktur gir mindre s... | 14.04.05 16:14 | 01.03.05 11:45 |  | v | birthe |
|  | 721 | Leder | 14.04.05 16:19 | 01.03.05 11:44 |  | v | birthe |
|  | 726 | Proffe kunder bruker pr... | 14.04.05 16:36 | 01.03.05 11:43 |  | v | birthe |
|  | 702 | - Stimuler dine ansatte ... | 30.11.04 14:17 | 01.12.04 09:09 |  | v | birthe |
|  | 700 | Notiser | 30.11.04 13:28 | 01.12.04 09:07 |  | v | birthe |
|  | 701 | Lede med glede | 14.04.05 13:31 | 01.12.04 09:07 |  | v | birthe |
|  | 694 | Bedriftsledelsen ønsker... | 14.04.05 15:25 | 01.12.04 09:06 |  | v | birthe |

Figure 7: Article management in Primetime Portal

### *Publishing*

Primetime Portal has a very straightforward publishing routine. Upon the creation of an article, a publishing date is set, and if necessary, an unpublished date. Whether an article is viewable online depends on whether the current date is within the specified timeline or not.

There is a special template for aggregating published articles into news columns, and via hyperlinks the reader can access singular articles.



Stikk tittel: Forside

Tittel: * - Bruk heller en mester!

Ingress: De aller fleste håndverkere er seriøse utøvere av sitt fag. Skrekkhistorier om uheldige kjøpere av håndverkstjenester er unntakene. Men husk å være føre vár!

Ingress bilde:

Justering: venstre

Brødtekst: Normal

Seriøsitet, varige kundeforhold og respekt for fagområdet er kjennetegn for en mesterbedrift. Dette gjelder uansett hvilke av de 66 fagene det dreier seg om, og uansett om mesterbedriften er liten eller stor. En mester er vant med krevende kunder, og setter sin ære i å leve opp til disse kravene.

- Et mesterbrev gis på bakgrunn av både praktisk og teoretisk kunnskap.
- I tillegg til å ha svennebrev og dokumentert praksis, har en mester videreutdannelse innen både økonomi, markedsføring, ledelse og høyere yrkesteori. Denne kompetansen formidles videre innad i bedriften, og vil derfor prege alt arbeid en mesterbedrift utfører.
- Du får ikke mesterbrev som takk for lang og tro tjeneste.
- En mestertittel fortjenes – det må vises til arbeid som reflekterer både gode håndverkstradisjoner og riktig bruk av moderne teknikker.
- En mesterbedrift bygger på ønsket om videreføring av gode håndverkstradisjoner, ikke på ønsket om kortsiktige gevinster.
- En mesterbedrift kan derfor gi et kundeforhold som varer så lenge du selv ønsker det – ikke et forhold som blir avsluttet fordi bedriften du bruker forsvinner.

Bruk heller en mester, se etter mestermerket!

20.04.06 19:02

Figure 8: Creating content in Primetime Portal

Published articles are viewed typically by requesting a certain page-ID from a dynamic JSP, like this page loads the news article by the ID-number 7,

```
http://www.mesterbrev.no/omoss/omoss.jsp?id=7
```

#### *Administration*

The administration controls of Primetime Portal client Applet are separated from the content controls into its own menu. From here the user administration is managed. Access rights are based on a model where access holders are either single users or user lists. An access right applies to one action on one category of content. For example, the *news-managers* list has reading, writing and publishing rights on the article category *news.* The single username *peter* is an administrator with reading, writing and publishing rights to every category of articles, as well as access to the administration module.

#### *Workflow*

Primetime Portal has no control of advanced workflow routines. As stated above, it has single-step publishing procedure where content is either published or not. Drafts can be simulated by publishing an article outside the current period.

### 4.1.3 Globalization

#### *Internationalization*

There are installations of Primetime Portal which have been internationalized. The internationalization module offers the article author an additional pane in which the text can be edited for each language introduced to the page. In reality there is actually one article in the database for each translation of the article, and

The administration interface of Primetime Portal is written entirely in Norwegian. There is no support to support internationalization of these interfaces without major upgrade to the interface architecture.

#### *Localization*

The implementation has no built-in module for localization. It is made by Norwegian developers for Norwegian users, and localization has never been an issue. It is of course possible to modify the JSP templates to attain localization functionality, and this was implemented as discussed when comparing the implementations in the next chapter.

### 4.1.4 Content Delivery

#### *Syndication*

Primetime Portal has the convenient option of publishing articles to a user list by e-mail. Even though this is a very typical feature of a news-publishing system, it has no innate place in a WCMS by definition. E-mail is not Web-content, rather e-mail is a protocol that runs on the Internet, in parallel, but not included in the World Wide Web. There are also some ready made JSP templates that produce content in XML news-feeds (RSS).

#### *Accessibility*

A simple experiment was made to test the accessibility of the alternative solutions. The text-based browser Lynx[14] displays web-pages in the most primitive way, sacrificing colors, images, frames, tables and design for terminal text so the browser can be run on terminals like Linux servers incapable of graphical user interfaces.

The Primetime Portal's web-shop presentation is more confusing and disfigured. Its content reaches across 60 lines of text for the page displaying a product. The text and image references are widespread, left- and right aligned, resulting in a chaotic browsing experience.

#### *Search*

Primetime Portal runs nightly compilations of the search index by *spidering* (Chakrabarti, 2002) the URLs and hyperlinks within the web-pages, and leaves the rest to the search engine based on Apache Lucene[15] which powers the search and sorts results by relevance.

This functionality is made available to the end user visiting the web-site controlled by the WCMS. The search engine can search both the database and the file archive.

#### *Communication*

For communication with the web-site's visitors, Primetime Portal features a forum. Also known as bulletin board, the forum is the oldest form of community communication on the

[14] http://lynx.browser.org
[15] http://lucene.apache.org

Internet right after e-mail and news groups. Primetime's solution here is pretty straight forward and simple to use, as most of the clients are not too concerned with this kind of communication.

It also features online surveys. Handing out questionnaires at the front door of a web-site is happening more and more often. Another variant are simple polls with single question and select options. It has also been used for online quiz competitions.

### 4.1.5 Costs

The majority of Primetime's costs are related to the time the programmers spend on developing the solution as developer hourly wages are high.

The training needed for the customer is virtually none since all the customers are long-time users of the Primetime Portal. They are comfortable with the client-side Applet and already have a good understanding on how to use and tweak the system

Additional costs for Primetime Portal includes the license for Oracle database, but that will not be considered in this context since the WCMS is actually independent on choice of database, and a free alternative such as PostGreSQL could have been chosen to cut costs.

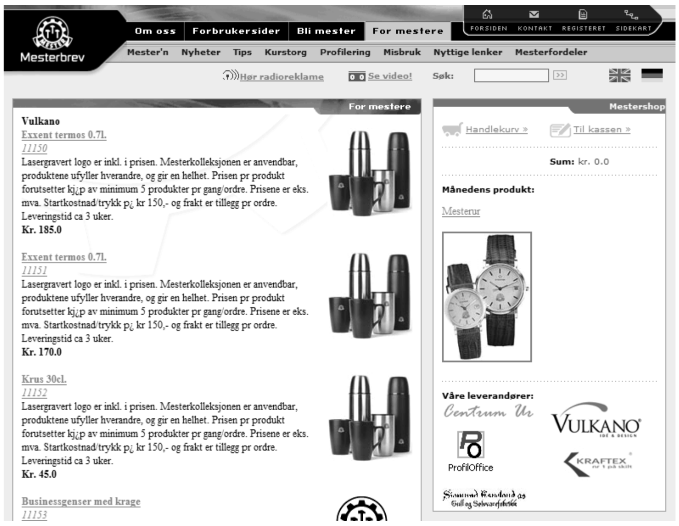
The customers seem to consider their web-site as a necessary expense for profiling their company online. They have full access to the earlier mentioned web-site's monitored statistics. By introducing web-shop functionality, as was done for testing the next requirement, it was proven that a WCMS can actually be used to generate larger amounts of revenue by making product catalogs available in online web-shops.

Unfortunately, Primetime had no insight into the customer's financial results, so the direct effect of introducing the web-shop was not measurable.

Primetime Portal also has one module for monitoring click-through on web-site ads and commercial banners. Third-parties often use the customer's web-pages for displaying ads, and measuring the amount of banners clicked is important functionality if this is provision for the payment of ads.

### 4.1.6 Extensibility

Implementing the web-shop in Primetime Portal was done as a development project over a one month period.



Vulkano
Exxent termos 0.7l.
11150
Lasergravert logo er inkl. i prisen. Mesterkolleksjonen er anvendbar, produktene ufyller hverandre, og gir en helhet. Prisen pr produkt forutsetter kj¿p av minimum 5 produkter pr gang/ordre. Prisene er eks. mva. Startkostnad/trykk p¿ kr 150,- og frakt er tillegg pr ordre. Leveringstid ca 3 uker.
**Kr. 185.0**

Exxent termos 0.7l.
11151
Lasergravert logo er inkl. i prisen. Mesterkolleksjonen er anvendbar, produktene ufyller hverandre, og gir en helhet. Prisen pr produkt forutsetter kj¿p av minimum 5 produkter pr gang/ordre. Prisene er eks. mva. Startkostnad/trykk p¿ kr 150,- og frakt er tillegg pr ordre. Leveringstid ca 3 uker.
**Kr. 170.0**

Krus 30cl.
11152
Lasergravert logo er inkl. i prisen. Mesterkolleksjonen er anvendbar, produktene ufyller hverandre, og gir en helhet. Prisen pr produkt forutsetter kj¿p av minimum 5 produkter pr gang/ordre. Prisene er eks. mva. Startkostnad/trykk p¿ kr 150,- og frakt er tillegg pr ordre. Leveringstid ca 3 uker.
**Kr. 45.0**

Businessgenser med krage
11153

*Figure 9: A product category in Acceptas*

A new module was created by the name of *Acceptas*. It contained a number of domain classes, including web-shop *product*, *catalog*, web-shop *category* and web-shop *cart* and *order*. The top level service functionality or interface to the web-shop functionality was provided by a Catalog class, which in turn handled data access objects (Sun, 2002) for products and categories. For storing products, the database was expanded with a table for keeping relations between product articles and product categories. The final result included the catalog template which is displayed in Figure 9 above.

The client-side Applet was modified to display an additional option while managing content articles of the kind *product*.

Two new web-page templates were created to present both the product categories and single products. These templates were JavaServer Pages based on copies from the existing mesterbrev.no article templates.

The source code of these classes are not included the Appendix. Primetime did offer to publish the code in this thesis, but because the module alone is 4261 lines of code, as well as being of very unreadable quality, including the source was deemed to be impractical. Figure 10 above gives a certain feel of the architecture and complexity of the module. We have included the source of the cart template to give an idea of what the idea of what the templates look like, but note that the template has been heavily modified by removing all design. The real cart template is 556 lines of code, scattered with snippets of JavaScript and JSP code, rendering it impractical for inclusion in the Appendix. There is one screenshot from the Eclipse JSP-editor in the Appendix (Figure 16: Working with Primetime Portal templates, page 103) which illustrates the re-usability of the code.

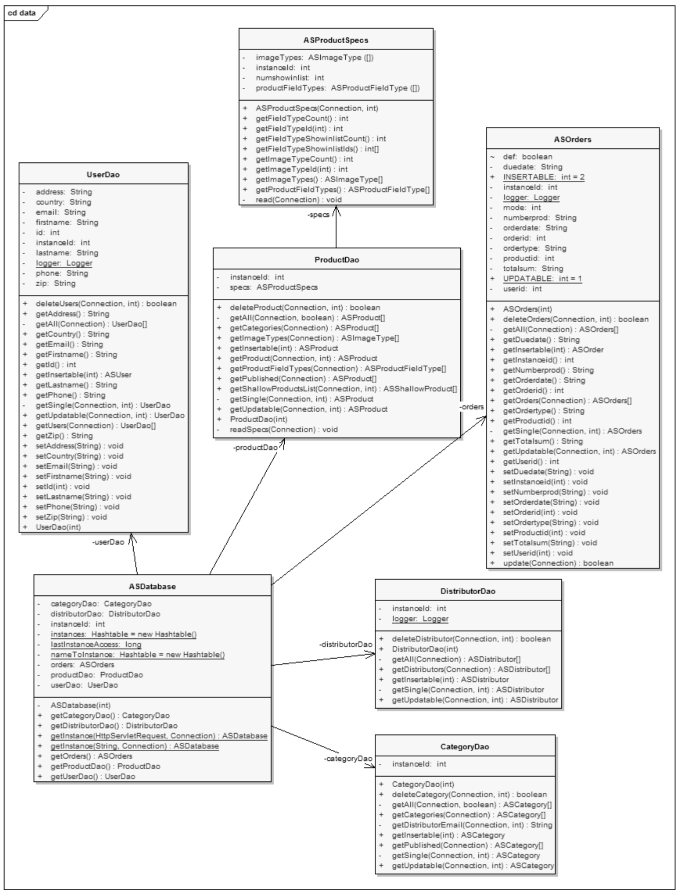

cd data
ASProductSpecs
- imageTypes: ASImageType ([])
- instanceId: int
- numshowinlist: int
- productFieldTypes: ASProductFieldType ([])
+ ASProductSpecs(Connection, int)
+ getFieldTypeCount() : int
+ getFieldTypeId(int) : int
+ getFieldTypeShowinlistCount() : int
+ getFieldTypeShowinlistIds() : int[]
+ getImageTypeCount() : int
+ getImageTypeId(int) : int
+ getImageTypes() : ASImageType[]
+ getProductFieldTypes() : ASProductFieldType[]
- read(Connection) : void
-specs
ProductDao
- instanceId: int
- specs: ASProductSpecs
+ deleteProduct(Connection, int) : boolean
- getAll(Connection, boolean) : ASProduct[]
+ getCategories(Connection) : ASProduct[]
+ getImageTypes(Connection) : ASImageType[]
+ getInsertable(int) : ASProduct
+ getProduct(Connection, int) : ASProduct
+ getProductFieldTypes(Connection) : ASProductFieldType[]
+ getPublished(Connection) : ASProduct[]
+ getShallowProductsList(Connection, int) : ASShallowProduct[]
- getSingle(Connection, int) : ASProduct
+ getUpdatable(Connection, int) : ASProduct
+ ProductDao(int)
- readSpecs(Connection) : void
-productDao
UserDao
- address: String
- country: String
- email: String
- firstname: String
- id: int
- instanceId: int
- lastname: String
- logger: Logger
- phone: String
- zip: String
+ deleteUsers(Connection, int) : boolean
+ getAddress() : String
- getAll(Connection) : UserDao[]
+ getCountry() : String
+ getEmail() : String
+ getFirstname() : String
+ getId() : int
+ getInsertable(int) : ASUser
+ getLastname() : String
+ getPhone() : String
- getSingle(Connection, int) : UserDao
+ getUpdatable(Connection, int) : UserDao
+ getUsers(Connection) : UserDao[]
+ getZip() : String
+ setAddress(String) : void
+ setCountry(String) : void
+ setEmail(String) : void
+ setFirstname(String) : void
+ setId(int) : void
+ setLastname(String) : void
+ setPhone(String) : void
+ setZip(String) : void
+ UserDao(int)
-userDao
ASOrders
~ def: boolean
- duedate: String
+ INSERTABLE: int = 2
- instanceId: int
- logger: Logger
- mode: int
- numberprod: String
- orderdate: String
- orderid: int
- ordertype: String
- productid: int
- totalsum: String
+ UPDATABLE: int = 1
- userid: int
+ ASOrders(int)
+ deleteOrders(Connection, int) : boolean
- getAll(Connection) : ASOrders[]
+ getDuedate() : String
+ getInsertable(int) : ASOrder
+ getInstanceid() : int
+ getNumberprod() : String
+ getOrderdate() : String
+ getOrderid() : int
+ getOrders(Connection) : ASOrders[]
+ getOrdertype() : String
+ getProductid() : int
- getSingle(Connection, int) : ASOrders
+ getTotalsum() : String
+ getUpdatable(Connection, int) : ASOrders
+ getUserid() : int
+ setDuedate(String) : void
+ setInstanceid(int) : void
+ setNumberprod(String) : void
+ setOrderdate(String) : void
+ setOrderid(int) : void
+ setOrdertype(String) : void
+ setProductid(int) : void
+ setTotalsum(String) : void
+ setUserid(int) : void
+ update(Connection) : boolean
-orders
ASDatabase
- categoryDao: CategoryDao
- distributorDao: DistributorDao
- instanceId: int
- instances: Hashtable = new Hashtable()
- lastInstanceAccess: long
- nameToInstance: Hashtable = new Hashtable()
- orders: ASOrders
- productDao: ProductDao
- userDao: UserDao
- ASDatabase(int)
+ getCategoryDao() : CategoryDao
+ getDistributorDao() : DistributorDao
+ getInstance(HttpServletRequest, Connection) : ASDatabase
+ getInstance(String, Connection) : ASDatabase
+ getOrders() : ASOrders
+ getProductDao() : ProductDao
+ getUserDao() : UserDao
-distributorDao
DistributorDao
- instanceId: int
- logger: Logger
+ deleteDistributor(Connection, int) : boolean
+ DistributorDao(int)
- getAll(Connection) : ASDistributor[]
+ getDistributors(Connection) : ASDistributor[]
+ getInsertable(int) : ASDistributor
- getSingle(Connection, int) : ASDistributor
+ getUpdatable(Connection, int) : ASDistributor
-categoryDao
CategoryDao
- instanceId: int
+ CategoryDao(int)
+ deleteCategory(Connection, int) : boolean
- getAll(Connection, boolean) : ASCategory[]
+ getCategories(Connection) : ASCategory[]
- getDistributorEmail(Connection, int) : String
+ getInsertable(int) : ASCategory
+ getPublished(Connection) : ASCategory[]
- getSingle(Connection, int) : ASCategory
+ getUpdatable(Connection, int) : ASCategory

*Figure 10: Class diagram for Acceptas module*

## 4.2 Magnolia

One of the most well established open source WCMS in the Java world is Magnolia WCMS[16]. It runs on top of the Magnolia framework, but is nonetheless most often referred to as just *Magnolia*. The project is sponsored by a commercial company in Switzerland called Obinary. Obinary employs the handful of developers who are driving the core development, but development is also supported by the community surrounding the Magnolia project environment.

As is evident from the state of the art there are many open source WCM systems in existence today. The choice of system to adopt fell on Magnolia for a number of reasons, the deciding factor being to adopt a system developed in Java. The company Primetime has specialized on developing web applications with Java since the beginning of this decade, and Java is also my preferred programming. The range of open source Java WCM systems was limited to a handful at the time when the choice was made. Some systems were ruled out because of restricting licenses, for example Jahia, with its own non-free license. Others were using outdated technologies, for example OpenCMS was based on the Apache Struts project, which now has ceased development.

### *Architecture*

Magnolia is designed to utilize a standardized interface for storage. This interface is Java Specification Request number 170, the Content Repository for Java technology API (JCP, 2006). It is often referred to as the JCR in the industry, and that term is the one which is used within this thesis. The reference implementation in particular is done in an open source project called Apache Jackrabbit (Jackrabbit, 2006). Other implementations have been made by Day (Day, 2006) and eXo (eXo, 2006). This has the advantage of detaching the Magnolia from one certain content repository implementation, having Apache Jackrabbit as the default choice.

All content is stored in this repository as nodes in a tree-structure. Rendering content for the Web is done by letting the templates dynamically generate web-pages based on the content in

[16] http://www.magnolia.info

the repository. To optimize the browsing experience, these generated web-pages are saved within the web application as static HTML files so they need not be generated again the next time they are viewed, resulting in higher performance. The generated HTML files are deleted when changes in their content are detected, and updated files is generated to replace the content.

The content of the repository is accessed by the Magnolia middleware. This is a set of service classes, most of which are irrelevant for the end-user or developer. The service layer is accessed via the Magnolia application programming interface (API)[17].

There are two ways to use the Magnolia API. The simplest and most practical way is to access and control the web content by using the Magnolia tag libraries. These tags are bundles of functionality which can be easily plugged into JavaServer Pages. The other way is done by accessing the Magnolia API directly through Java Servlets.

Like with Primetime Portal, content is rendered through templates made with JSP, although there is a clean separation of content and formatting with an extensive use of CSS and composite pages built from smaller modular JSPs combined together.

### *Use*

Several meetings were arranged between the customer and the developers during the development so we received concurrent feedback about the usability of the system. The overall impression was similar to the experience of developing with Magnolia compared to Primetime Portal, namely that although some learning curve had to be overcome at the start of the project, Magnolia eventually ended up as the most comfortable tool to work with.

With Magnolia there was an entirely new graphical user interface for the customer to handle. We held Magnolia tutorials in several sessions, added together approximately two hours, after which the customer seemed to manage well, occasionally needing support or instructions by e-mail or telephone.

[17] http://magnolia.sourceforge.net

The customer was also pleased with the fact that no special software other than the browser was required, and that the web-site content could first be previewed in the authoring instance before being published to the live site.

### *Development*

Magnolia is free software hosted at SourceForge[18]. It is free to develop new plug-ins for implementing the extra functionality customers could request, free to use and free to distribute, as long as procedures are in accordance with the LGPL license. Another major significance for Magnolia being an open source software is the community surrounding the project.

Communication and discussion on Magnolia is done through two mailing lists. One list is for developers where issues (being either bugs, improvements, new features or tasks) are discussed and solved. The other list is for Magnolia *users*, meaning external developers who are using Magnolia in their own environment. As the term *developer* is used in this thesis, the Magnolia users are actually developers who are implementing Magnolia as their WCMS.

The Magnolia developer documentation effort is a mix of collaborative additions onto their wiki[19] as well as documentation written by the developers which includes a user manual, a quick start guide to creating templates and developer documentation. All three documents are freely available for download from the Magnolia web-page.

Development follows a series of best practices within software development, including test-driven development, elements from eXtreme Programming (Beck, 2004) like issue tracking, and use of tools like Subversion[20] and Apache Maven[21]. The goal of this development method is of course to make it as easy as possible for any developer who wishes to participate in the project to do so.

The next sections discuss Magnolia's fulfillment of the WCMS requirements.

[18] http://www.sourceforge.net
[19] http://wiki.magnolia.info
[20] http://subversion.tigris.org
[21] http://maven.apache.org

### 4.2.1 Technical

#### ***Deployment***

The process of deploying Magnolia consists of dropping the web application packaged as a WAR-file into the application server, and the rest of the deployment is done automatically. The WAR-file is extracted into the standard web application directory structure, the embedded database service is started, the repositories are initialized from seed-content XML files lying within the application, and finally the web application context is started, ready to receive requests from the Web. The entire procedure takes approximately two minutes, but is highly dependent on the runtime environment, including which version of Java and Tomcat are used, and what the hardware specifications of the server are.

#### ***Integration***

The Magnolia project tries to comply with standards wherever possible, and this surfaces by example with the pluggable WYSIWYG-editor. It comes with its own simple text-editor, but due to the modular architecture of the content editing it is possible to exchange this with FCKEditor or the Kupu editor.

#### ***Templates***

Magnolia has a structured template configuration. They are handled from the administration interface's configuration, and enable heavy use of CSS for design, the Java Standard Tag Library (JSTL) and a well-documented set of custom tags[22].

The templates that come with Magnolia are all JSP Documents. They are XML-strict, leaving less room for breaking W3C's web standards.

After having worked with Magnolia for some days, the customer even found ways of using the templates which even we has developers had not thought of, by having product categories nested inside product categories. Due to the repository's hierarchical nature, this is a natural flexibility of the WCMS which proved quite useful to the customer and a positive surprise to both us and the customer.

[22] These technologies are part of Java Enterprise Edition – http://java.sun.com/javaee

#### *Backup*

Magnolia has a convenient exporting service. A convenience of Magnolia is not only the content itself, but also the configuration. User and role management and template registry are stored in the content repository.

#### *Monitoring*

Magnolia has no built-in monitoring functionality.

#### *Logging*

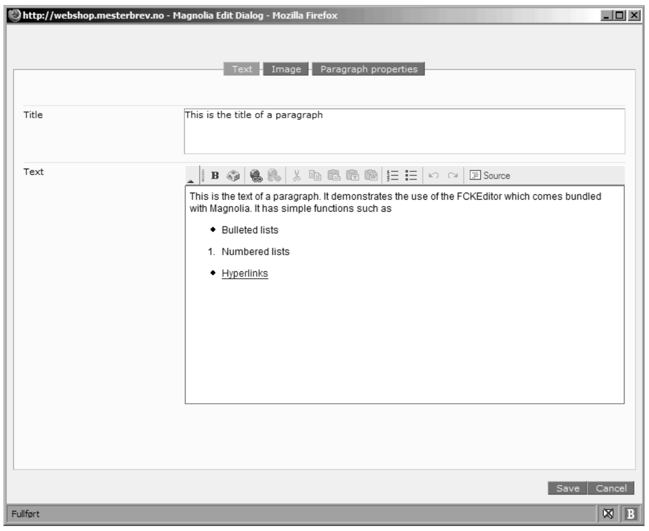
Logging is a different matter than with Primetime Portal on this point. Magnolia makes heavy but healthy use of the Log4J framework[23]. This means that the application can be run with different levels of logging enabled, outputting the logs themselves to neatly fitted files and formats. The default level produces only error-level messages in the logs, which means that the developer will only be notified when things go wrong. Other levels can be applied to produce more thorough loggings messages for debugging application bugs.

### 4.2.2 Management

#### *Creation*

The process of producing content inside Magnolia is very different from that of working inside Primetime Portal.

There are two modes to browse content for the Magnolia content manager. The first is within the AdminCentral where all the content nodes are viewed in a hierarchical overview.


http://webshop.mesterbrev.no - Magnolia Edit Dialog - Mozilla Firefox
Text
Image
Paragraph properties
Source
Save
Cancel
Fullført

Title: This is the title of a paragraph

Text: This is the text of a paragraph. It demonstrates the use of the FCKEditor which comes bundled with Magnolia. It has simple functions such as

- Bulleted lists

1. Numbered lists

- Hyperlinks

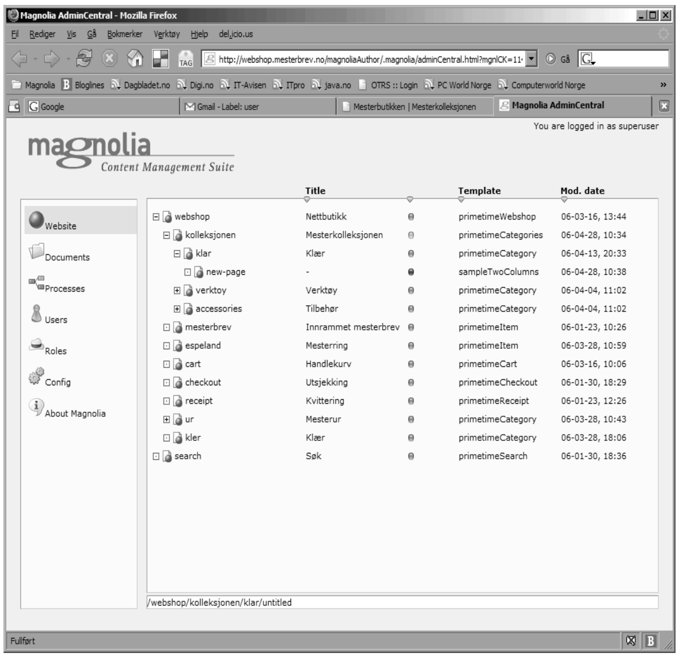
*Figure 11: Creating paragraphs in Magnolia*

[23] http://logging.apache.org/log4j/

Once a node is opened for editing, the manager enters the other mode of browsing where the content is perceived much like it will appear on the published web-page. The content of this page is of a much more component-based structure than in traditional content management systems where one piece of content is usually edited as one large document. Magnolia's content editing forces the author to think in a new way where each paragraph on a page is one piece of content, and this piece can either be a piece of text (see Figure 11), a picture, a table or a file that can be downloaded.

#### *Publishing*

Magnolia has one authoring and one public instance, one for authoring web-pages and another for displaying them online. This way the online content can be previewed and tested in the authoring instance before being activated or published onto the public instance. All content on the public instance is visible to the visitors. Inside the authoring environment the contents is browsed using the AdminCentral interface as displayed in Figure 12. Published content is indicated by "traffic lights", where red indicates unpublished, yellow indicates published but modified, and green indicates fully published. Note that the preferred Magnolia term for publishing is *activation*.


Magnolia AdminCentral - Mozilla Firefox
Fil Rediger Vis Gå Bokmerker Verktøy Hjelp del.icio.us
http://webshop.mesterbrev.no/magnoliaAuthor/.magnolia/adminCentral.html?mgnlCK=11
Gå
Magnolia Bloglines Dagbladet.no Digi.no IT-Avisen ITpro java.no OTRS :: Login PC World Norge Computerworld Norge
Google
Gmail - Label: user
Mesterbutikken | Mesterkolleksjonen
Magnolia AdminCentral
You are logged in as superuser
magnolia Content Management Suite
Website
Documents
Processes
Users
Roles
Config
About Magnolia
/webshop/kolleksjonen/klar/untitled
Fullført

| | Title | | Template | Mod. date |
|---|---|---|---|---|
| webshop | Nettbutikk | ● | primetimeWebshop | 06-03-16, 13:44 |
| kolleksjonen | Mesterkolleksjonen | ● | primetimeCategories | 06-04-28, 10:34 |
| klar | Klær | ● | primetimeCategory | 06-04-13, 20:33 |
| new-page | - | ● | sampleTwoColumns | 06-04-28, 10:38 |
| verktoy | Verktøy | ● | primetimeCategory | 06-04-04, 11:02 |
| accessories | Tilbehør | ● | primetimeCategory | 06-04-04, 11:02 |
| mesterbrev | Innrammet mesterbrev | ● | primetimeItem | 06-01-23, 10:26 |
| espeland | Mesterring | ● | primetimeItem | 06-03-28, 10:59 |
| cart | Handlekurv | ● | primetimeCart | 06-03-16, 10:06 |
| checkout | Utsjekking | ● | primetimeCheckout | 06-01-30, 18:29 |
| receipt | Kvittering | ● | primetimeReceipt | 06-01-23, 12:26 |
| ur | Mesterur | ● | primetimeCategory | 06-03-28, 10:43 |
| kler | Klær | ● | primetimeCategory | 06-03-28, 18:06 |
| search | Søk | ● | primetimeSearch | 06-01-30, 18:36 |

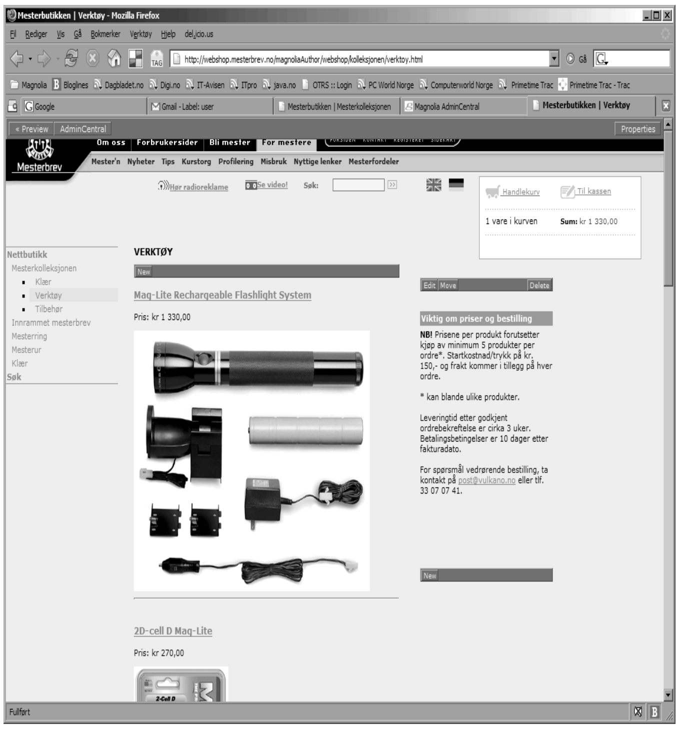
Figure 12: AdminCentral in Magnolia

All pages are accessed through a readable friendly Web address,

```
http://webshop.mesterbrev.no/webshop/kolleksjonen.html
```

This filename makes it easier for browsers to deal with the page concerning bookmarks and content type. Additionally, the page can be cached as a static file on the web-server, providing instant reload the second time the page is called. This has a significant impact on how long a page takes to load, given that the page does not have to be dynamically compiled at load-time, like the cart in the web-shop.

The customer seemed to grab the concept of activation quite quickly, but later had problems with this feature as hierarchical content is not recursively activated unless Magnolia is instructed explicitly to do so. As with the editing of content, using this functionality takes some practice to get used to. Figure 13 displays the category template in edit-mode. Note the menu to the left is generated from the content hierarchy in AdminCentral.


Mesterbutikken | Verktøy - Mozilla Firefox
Fil Rediger Vis Gå Bokmerker Verktøy Hjelp del.icio.us
http://webshop.mesterbrev.no/magnoliaAuthor/webshop/kolleksjonen/verktoy.html
Magnolia Bloglines Dagbladet.no Digi.no IT-Avisen ITpro java.no OTRS :: Login PC World Norge Computerworld Norge Primetime Trac Primetime Trac - Trac
Google Gmail - Label: user Mesterbutikken | Mesterkolleksjonen Magnolia AdminCentral Mesterbutikken | Verktøy
« Preview AdminCentral Properties
Om oss Forbrukersider Bli mester For mestere
Mesterbrev
Mester'n Nyheter Tips Kurstorg Profilering Misbruk Nyttige lenker Mesterfordeler
Hør radioreklame Se video! Søk:
Handlekurv Til kassen
1 vare i kurven Sum: kr 1 330,00
Nettbutikk
Mesterkolleksjonen
Klær
Verktøy
Tilbehør
Innrammet mesterbrev
Mesterring
Mesterur
Klær
Søk
VERKTØY
New
Mag-Lite Rechargeable Flashlight System
Pris: kr 1 330,00
Edit Move Delete
New
2D-cell D Mag-Lite
Pris: kr 270,00
Fullført

Viktig om priser og bestilling

**NB!** Prisene per produkt forutsetter kjøp av minimum 5 produkter per ordre*. Startkostnad/trykk på kr. 150,- og frakt kommer i tillegg på hver ordre.

* kan blande ulike produkter.

Leveringtid etter godkjent ordrebekreftelse er cirka 3 uker. Betalingsbetingelser er 10 dager etter fakturadato.

For spørsmål vedrørende bestilling, ta kontakt på post@vulkano.no eller tlf. 33 07 07 41.

*Figure 13: Creating content in Magnolia*

### *Administration*

Magnolia's main administration interface (AdminCentral) is divided into management of the *web-site*, *users*, *roles* and *configuration*. *Website* is where all the conventional content is stored, while the other administration configuration details are stored in the respective repositories. Magnolia's access control model is based on that a *role* has privileges to a set of actions, and a user can have one or more *roles*.

### *Workflow*

Magnolia can simulate workflow by setting up users and access rights. One can specify a drafting space in a part of the JCR where all authors have write access. Only the editor, however, has write access to the published part of the web-site, and may move content and publish as it becomes accepted.

Magnolia also has another method of creating workflow. In addition to the two default public and authoring instances, one can install several other instances and configure a path of activation from one instance to the next, for example draft-instance, author-instance, editing-instance and finally the published instance. In reality, if this happened to an important requirement from the customer, we would probably have chosen to deploy another system with such functionality embedded.

### 4.2.3 Globalization

#### *Internationalization*

Translating Magnolia content is a question of organizing the web content by language. One needs to manually define a site structure which will be repeated for each language. The advantage is a dynamic set of translations, but still there is some manual structuring to be done.

All administration text is stored in separate language property files. Translation is done by providing a new set of administration texts into a new property file, and invoking this inside the Magnolia interface. Each user can personally define the preferred language. Currently the Magnolia AdminCentral has been translated into 15 major languages.

#### *Localization*

Like Primetime Portal, Magnolia did not originally have any particular functionality for this feature, but it was implemented as a JSP custom tag, as witnessed in the discussions of this thesis.

### 4.2.4 Content Delivery

#### *Syndication*

Magnolia has another kind of syndication mechanism included. The activation process which takes place in between the authoring and public instances is actually part of the JSR 170 specification. The intention is that other WCMS which are JCR-compliant can be able to receive content from Magnolia installations. Another JCR bonus is that all content can be directly exported as XML identical in structure to how the content is stored in the JCR, but this XML model is complex and can require some extensive transformation to use elsewhere. One advantage of this is that content can be stored with meta-data from for example the Dublin Core name-space, making it easy to export in a semantically correct RDF-format.

#### *Accessibility*

Browsing the web-shop implemented in Magnolia with Lynx yielded a quite usable result. The header of the page does complicate the navigation somewhat, but it is still quite easy to browse, add and remove products from the web-shop cart. A typical screen displaying one product in the web-shop is presented across 41 lines of text, most of the content aligned neatly to the left, with the hierarchical menu for navigating the web-shop placed at the bottom of the page.

#### *Search*

Magnolia's search comes defined by the JSR 170 and is included in Jackrabbit's implementation of the JCR. It features an interface for searching both with an SQL-like format, as well as an XPath query format. The search also includes meta information surrounding the content nodes when performing the search, so that fields like author and modified-dates can be added as parameters to the search.

#### *Communication*

Magnolia has a ready made template for providing feed-back from the people browsing the web-pages. This template is configured with the access to a mail server, and can be easily modified to include various input fields such as text fields, check boxes, and selection boxes.

### 4.2.5 Costs

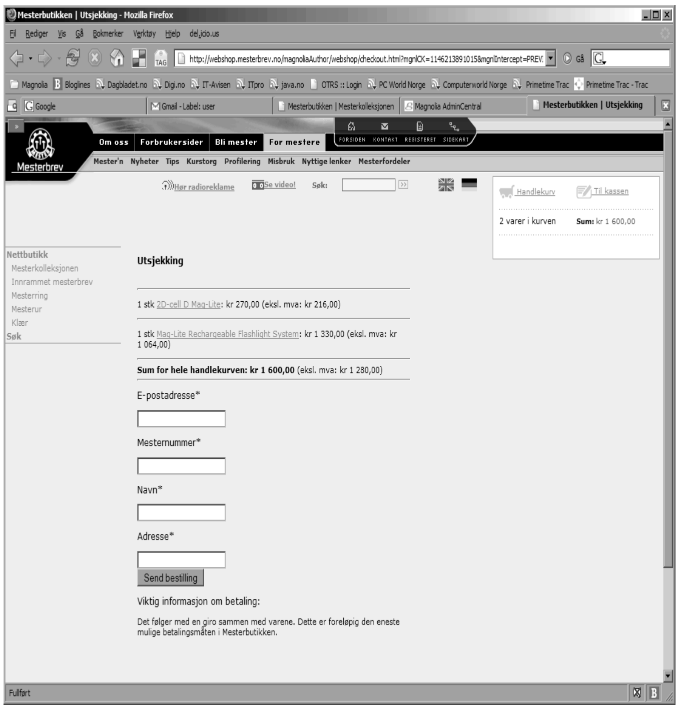
The costs related to development of the Magnolia web-shop module are of a different composition from that of Primetime Portal. Downloading the standard Magnolia installation is free, but on the other hand more training was needed to make the customer understand the new way of creating content.

It took approximately two weeks for one developer to create the web-shop module, but before this development could begin the programmer needed to acquire knowledge of the Magnolia architecture and JSR 170. This was done parallel to Primetime Portal development over a period of three months, and it is estimated that approximately 40 hours were spent learning Magnolia.

Magnolia does have the disadvantage of running a larger load of the content management processes on the server-side than Primetime Portal. A given number of Magnolia installations will require a more powerful hardware on the server-side, or fewer installations per physical server.

### 4.2.6 Extensibility

Magnolia has no native web-shop module. We put the extensibility of Magnolia to the test by implementing this module.


Mesterbutikken | Utsjekking - Mozilla Firefox
Fil Rediger Vis Gå Bokmerker Verktøy Hjelp del.icio.us
http://webshop.mesterbrev.no/magnoliaAuthor/webshop/checkout.html?mgnlCK=1146213891015&mgnlIntercept=PREV
Magnolia Bloglines Dagbladet.no Digi.no IT-Avisen ITpro java.no OTRS :: Login PC World Norge Computerworld Norge Primetime Trac Primetime Trac - Trac
Google | Gmail - Label: user | Mesterbutikken | Mesterkolleksjonen | Magnolia AdminCentral | Mesterbutikken | Utsjekking
Mesterbrev
Om oss Forbrukersider Bli mester For mestere
FORSIDEN KONTAKT REGISTERET SIDEKART
Mester'n Nyheter Tips Kurstorg Profilering Misbruk Nyttige lenker Mesterfordeler
Hør radioreklame Se video! Søk:
Handlekurv Til kassen
Nettbutikk
Mesterkolleksjonen
Innrammet mesterbrev
Mesterring
Mesterur
Klær
Søk
Send bestilling
Fullført

2 varer i kurven **Sum:** kr 1 600,00

**Utsjekking**

1 stk 2D-cell D Mag-Lite: kr 270,00 (eksl. mva: kr 216,00)

1 stk Mag-Lite Rechargeable Flashlight System: kr 1 330,00 (eksl. mva: kr 1 064,00)

**Sum for hele handlekurven: kr 1 600,00** (eksl. mva: kr 1 280,00)

E-postadresse*

Mesternummer*

Navn*

Adresse*

Viktig informasjon om betaling:

Det følger med en giro sammen med varene. Dette er foreløpig den eneste mulige betalingsmåten i Mesterbutikken.

*Figure 14: Checking out the cart from Magnolia web-shop*

In short this included the creation of five templates. The JSP was XML-strict and had a clear separation between content, style-sheets (CSS) and JavaScripts. Magnolia also came with a number of sample templates, and many of these were re-used in the implementation, for example the feed-back template was used to construct the checkout template. The source code of the template is available in the Appendix (cartPreview.jsp and cartMainColumn.jsp), and the result is displayed in Figure 14. A screenshot from editing templates is included in the Appendix (Working with Magnolia templates Figure 15, page 103).

Two Java classes were made, namely *cart* and *item.* Additionally there was a utility class and a custom tag enabling localized currency representation. These classes are available among the source code entries in the Appendix. To understand the source of these classes it is advised that one first becomes familiar with the JCR specification and the Magnolia API.

The Magnolia web-shop module consists of 219 lines of code. It is available online, although it has not been "released" to the members of the Mesterbrevnemd yet[24].

[24] http://webshop.mesterbrev.no

## 4.3 Comparing Evaluations

Throughout this section we compare and discuss how the suggested solutions fulfilled the WCMS requirements.

### 4.3.1 Technical

#### *Deployment*

Primetime Portal involves too many manual steps for it to be able to compete with the Magnolia on deployment. On the point of deployment, Magnolia clearly ahead of its proprietary adversary.

#### *Integration*

As the developers have been able to access and modify the source code of both systems, integrating the WCM systems with other services has not been hindered by obstacles of lock-ins or product restrictions. The real problems of integration appear as the systems become infested with snippets and hacks within modules to connect them to other applications.

Both systems follow a modular structure, and generally such integration can be modularized to a usable extent. It is however my opinion that the experiment offered no insight into how integration with other systems would perform. Both have the architecture of a stand-alone WCMS and have no special requisites for integration with other systems.

I would also like to note that the upcoming version of Magnolia 3.0 will feature an architecture based on the Spring framework[25], which will heavily impact on architecture and integration possibilities.

#### *Templates*

Both solutions use JSP in the view-layer, but Magnolia is years ahead of Primetime Portal when it comes to readability and structure of the templates. The fact remains that with an extensive effort Primetime Portal's templates could have been upgraded to match today's

[25] http://www.springframework.org/

standards, moving as much design as possible out to cascaded style-sheets (CSS) and meshing together templates to increase re-use of design. This would still lead Magnolia a step ahead because of its many controls which bundles amounts of functionality into single-line custom JSP tags.

#### *Backup*

Neither system excels with any special template backup functionality. The template files of both systems are set up with backup by file system copies to remote servers, and this is done at the discretion of the server technicians the same way they backup other files.

#### *Monitoring*

Responsibilities for tracking web-site traffic can be left to the web-server. The two solutions are both web applications running inside a web-server, and the server itself should be responsible for keeping track of traffic coming in and out of the server, as well as monitoring server loads on processor and memory use. None of the solutions provide any internal monitoring, although this would certainly be appreciated by many users who want to see content statistics and other relevant numbers.

#### *Logging*

There is no doubt that Magnolia's logging technique is superior to that of Primetime Portal. One might suggest that the reason for this lies in the nature of the open source programming style as the process implementing logging is tedious for a programmer, but publishing code without logging is considered “unclean” and can prove to be embarrassing for the authoring programmer.

### 4.3.2 Management

#### *Creation*

The advantage with Magnolia's more atomic content structure is that depending on the template, every area of the web-page can be edited. Primetime Portal is more rigid in its design. The editor can only change the content within the article itself. All arbitrary columns, menus, footers and headers are strictly generated by the template.

The customers who were part of the experiment were all seasoned users of Primetime Portal. The customer's web editors showed negative reactions towards Magnolia's foreign and component-based way to treat content. This took some getting used to, and training from the developer through several sessions was necessary. They were however pleased with the WYSIWYG-editor and had minimal problems working with the WCMS through the web browser as opposed to using a standalone desktop application or Applet.

The Applet includes a user-friendly content editor. The application can edit HTML content in a WYSIWYG manner. Composing, editing, publishing and management otherwise of the application is done through this Applet, as is the administration of the other modules.

Magnolia uses a “web-end” content editor, being actually nothing else than a simple web-page, utilizing HTML, CSS and JavaScript for a thick client-feel and touch. This means that the Magnolia client is lightweight, and will work inside any conventional web browser like Internet Explorer or Mozilla Firefox.

#### *Publishing*

As the users gained proficiency within the use of both solutions, the publishing mechanism of Primetime Portal was still the one preferred, especially due to the nature of web-shop content. Occasionally the customer wishes to publish a special offer or product over a limited period only. As of today, only Primetime Portal supports this period-publication functionality.

#### *Administration*

Both solutions provide administration of access control. While Primetime Portal has a very linear and straightforward, Magnolia has a generic model where access control can be hierarchically modified, somewhat akin to how access control is done on file systems. As developers we preferred the latter solution, but customers naturally preferred the Primetime Portal way of doing administration.

#### *Workflow*

Both systems are lacking when it comes to control of workflow, but this might change in the future for Magnolia. The project leaders are in the process of merging their efforts with another open source project called OpenWFE that implements features for workflow, so the situation is likely to change (Obinary, 2005). According to Magnolia's roadmap the merged software will be released May 2006.

### 4.3.3 Globalization

#### *Internationalization*

Primetime Portal has a certain edge on the point of translation, as alternative language content is edited in such an intuitive fashion. Magnolia more or less leaves internationalization to the developer, letting them implement parallel content for each language in the JCR. The Magnolia community has produced several viable best-practices on how to use the content repository in this manner.

#### *Localization*

By design principle, the view-layer of the web application is responsible for localization, herein the JSP templates. The templates can figure out the locale of the visitor by checking the header of the request packages which are sent from the browser. These requests are obliged to contain an information field describing which locale and language the user prefers.

A simple example of localization was implemented for the solutions. Countries use different annotations of currencies, and monetary notation is used quite frequently in a web-shop. Prices are represented internally as mere a data-pieces of the type *double*, for example,

```
245.5
```

A localized Norwegian price should be noted with the abbreviation of Norwegian kroner which is *kr,* followed by the number, and using a comma to separate decimals,

```
kr 245,50
```

While an American price would be annotated with a punctuation mark to split away the decimals,

```
$245.50
```

Of course the same price can not be displayed for different currencies (US Dollars are more worth than Norwegian kroner), but this is outside the scope of the localization functionality. None of the solutions had this sort of functionality built in, so it was implemented as a pluggable custom currency tag where the locale could be set with one parameter. In the developer's case the parameter would be “no”, the locale identifier for Norwegian/Norway. Upon request the price represented by a double is evaluated by the currency tag which adjusts the notation of the price accordingly. The Currency tag class and its descriptor are available in the Appendix of this thesis.

As the feature was implemented as a JSP custom tag, it is in principle fully possible to apply it to both solutions. There is no difference in the performance between Magnolia and Primetime Portal on this point.

### 4.3.4 Content Delivery

#### *Syndication*

None of the solutions really provide satisfactory syndication functionality, but we will admit that Magnolia shows great potential since the JCR can virtually store content in any kind of XML document object model. It is merely one template away from providing news-feeds of content. Still, Primetime Portal already has RSS-templates ready for content syndication, and this puts the proprietary solution one step ahead on this requirement.

#### *Accessibility*

The result of browsing the solutions with Lynx is tightly connected to the way the templates have been implemented, witness the discussion related to the technical requirements. The conclusion is the same, namely that the Primetime Portal templates could have been implemented in a better way following web standards, but this has not been the case so far. Until then Magnolia's accessibility lies far ahead.

***Search***

Both solutions use a search engine based on the same software, the Apache Lucene project, but the search indexing mechanisms are implemented in different ways. The immediate search results of the two solutions are very similar, but Primetime Portal's search result is harder to modify.

***Communication***

Even though the modules are somewhat hard to deploy as Primetime Portal is in general, it is still some way ahead of Magnolia on this point. The Magnolia developers are at the time of writing trying to solve the problem of accepting content from visitors, as their architecture is built quite rigidly for only accepting content which comes one-way from the authoring instance.

### 4.3.5 Costs

There are different aspects to consider when judging the cost of software.

Consider a hosted WCMS service expanding its modules with one web-shop module. The one customer that desired the module could pay for the entire development of the module even if it the module can be re-used with other customers, but this is likely to be a to steep price for one customer to pay as these users of hosted services are often smaller businesses. It does not seem fair that each customer should pay development costs for completed implementation.

On the other hand, this might not be worthwhile for the WCMS developer to create new modules if there is no guarantee that several customer will pay for the benefit of having this additional module.

So there are two perspectives to consider. WCMS development can either be owned and paid for by the developer, or be under the ownership of the customer who is need of the developed functionality.

The perspective chosen in this thesis is that of the developer's. The total cost of ownership and return on investment is here organized by the expenses of the developer. This grants the most realistic and actual cost of implementing and using the solution, disregarding market powers such as demand and competition which might affect the price the customer ends up paying.

So the question is whether it is cheaper to build in-house or outsource the solution.

If one were to compare the entire cost of development of the Primetime Portal architecture to that of acquiring the Magnolia system for free, the latter alternative would by no doubt quickly win any comparison on TCO.

We believe that a Magnolia web-site will see no different amount of traffic from that of Primetime Portal. The belief is that the resulting revenue will not be affected on whether the underlying system was powered by Magnolia or Primetime Portal due to the similar ending graphical user interface. This is because the end-result of the different WCM system are quite similar to the visiting customer. The returns of switching to Magnolia are instead increased internally as it becomes easier to integrate the web content with the web-shop items.

Since Magnolia is a smaller investment than Primetime Portal and have lesser costs, while at the same time providing equal returns on traffic, Magnolia is easily the solution with the largest ROI.

### 4.3.6 Extensibility

The web-shop was first implemented through Primetime Portal, while the Magnolia implementation was done two months later. Note that when implementing the web-shop for the second time, it was experienced as a much simpler process. The domain was already explored, the customer had become familiar to the development process, and most of the requirements were ready.

Developing a web-shop module on a WCMS is a complex procedure. First one needs to understand the WCMS, its architecture and its code. Magnolia and Primetime Portal are widely different, but some common procedure was recognizable in the development between the two implementations.

In each case a *cart* class was made to keep track of how many products of which kind were put in the virtual shopping cart. The object was attached to the session object of a visiting browser so the web application could keep track of which products were put in the cart for each visitor. This is normal session handling in JSP technology and was done quite similarly in both cases.

Templates were made for displaying the following

- The front-page of the web-shop with a summary of product categories available
- A product category page with a short summary of each product
- A product page with full products details and the option of adding it to the cart
- A cart page where items could be removed from the cart
- A checkout page with a cart overview and a personal information form for submitting the order

The content of these templates were widely different. Both were based on existing templates, and both had similar logic for removing and adding items from the cart. The size of the implementation was different. The Primetime Portal Acceptas library consists of roughly 30 relatively large utility classes. These are mainly duplicates of Primetime Portals normal core libraries, only they have been extended to fit product articles as well.

The Magnolia web-shop module has four small classes, and by this number alone it is clear that the complexity of this implementation was by far the lightest.

A key difference between using the templates in Primetime Portal and Magnolia is that in the latter alternative, you must first register the template properly within the administration interface, AdminCentral. Afterwards the templates can be freely applied to any page in a user-friendly manner. In Primetime Portal the JSP template is simply copied into the web directory, and used directly by its file name. In a way, this means that content and design is mixed.

Consider the the checkout page. In Primetime Portal, this is a page, and it is also functionality in a template. In Magnolia a page is first created and then coated with the template type checkout. Magnolia separates the content from the template. Primetime Portal does not. As a result it is harder to re-use the latter's template in later solutions.

Another consequence of extending Primetime Portal's database design with the product category was that the search did not extend to this new content type. In Magnolia, the web-shop products were equal of any other content on the installation, so the search-engine indexed the products into the search-base automatically. Primetime Portal's search engine would have had to be tuned into indexing the new product table in the database.

## 4.4 Summary

In this chapter we have presented the two solutions which sought to solve the web content management challenges. Primetime Portal and Magnolia are both WCM systems of sizable implementation, they are both built on Java technology and they are both trying to meet similar requirements. They differ in architecture, method and performance in a varying degree that has been measured and compared. Based on the findings we will proceed to study and discuss the collected experiences in the next chapter.

# 5 Discussion

The exploration of the previous chapter has so far been a benchmark on the performance and adaptability of the two WCM systems. It should be repeated that this has not been an objective evaluation with the goal of discovering which solution is most suitable for a typical customer in need of a web-shop for their web-site. Rather it has been an in-depth participative experiment where practical research has been made by trialling with the implementation of a WCMS from a developer's point of view.

Table 2: Evaluation of Requirements below summarizes the findings in the iteration of the requirements and the solutions' performance, on a scale from 0 to 3 where 0 indicate no compliance or implementation, ranging up to 3 indicating excellent or full compliance or implementation. The requirements of costs and extensibility are exempted from the summary due to their non-functional nature.

| ***Requirement*** | ***Primetime Portal*** | ***Magnolia*** | ***Notes*** |
|---|---|---|---|
| Deployment | 1 | 3 | Open source |
| Integration | 0 | 2 | Standards |
| Templates | 1 | 3 | JSP |
| Backup | 1 | 2 | Implementation |
| Monitoring | 2 | 0 | Implementation |
| Logging | 0 | 3 | Open source |
| Content creation | 2 | 2 | Implementation |
| Publishing | 2 | 1 | Implementation |
| Administration | 2 | 3 | Implementation |
| Workflow | 0 | 1 | Implementation |
| I18N | 0 | 2 | Open source |
| L12N | 1 | 1 | JSP |
| Syndication | 1 | 1 | JSP |
| Accessibility | 1 | 3 | Standards |
| Search | 1 | 2 | Implementation |
| Communication | 0 | 0 | Implementation |

*Table 2: Evaluation of Requirements*

Note that some of the differences in performance are due to different quality of implementation. This is connected to the actual age of the code as Magnolia is simply newer than Primetime Portal. A wider range of modern tools and frameworks were available at the time of implementation. The ones noted with *JSP* and *Implementation*, are requirements which the developers admit that given some time and resources, the Primetime Portal solution could be made equivalent of Magnolia's.

The next sections aim to find which requirements are dependent on open standards, and which ones are more easily fulfilled by being implemented by an open source project.

## 5.1 Requirements that benefit from Open Standards

Third-party developer regularly desire access to the web content of Primetime's customers. For security reasons, Primetime can not simply grant them a direct connection the database management system. The most usual solution is to custom tailor a JSP that performs the data access and delivers it in XML-format to the third party. Without delving too deeply into the negative consequences this has for the service in the long run, it suffices to say that such is a quick and dirty integration.

The JCR offers a standardized service of content exchange. With some investment into the use of the JCR-standard, developers of remote services can access exports from the JCR. The specification comes with support for authentication, versioning, transaction management, observation (for monitoring changes in the repository), and search. A third party who knows the JCR specification is fully able to make use of the WCMS content, as long it has permission and credentials to do so.

Accessibility is the other requirement which benefits largely from the use of open standards. This has little consequence for the developer, other than that as long as she makes use of open standards there will be fewer complaints on the usability of the web-sites.

## 5.2 Requirements that Benefit from Open Source

The success of an open source project relies heavily on how easy it is to build and install the software. If it is easy to get started, more people are likely to join the open source community, so the Magnolia developers have focused on keeping *deployment* of Magnolia to a single-step process. This has broad appeal to both technicians and developers.

The same appeal is likely to be the reason for Magnolia's logging utilities. When a developer encounters a bug or error in the implementation she will apply to the mailing list with her problem. A well made logging system makes it easier for the rest of the community to isolate the bug and assist the developer in solving the problem.

The reason *internationalization* is noted with *open source* in Table 2 is the international community surrounding the project. Had it not been for Obinary turning Magnolia into an open source project, it might would have still had a German-only administration interface, like Primetime Portal has a Norwegian interface today.

## 5.3 Performance

It is the experience of the developer that Magnolia is a heavier program than Primetime Portal when it comes to server-load. This is mostly due to the JCR which runs bundled inside Magnolia, each with its individual database. Primetime Portal's entire portfolio of installations uses one single database management system on a remote server.

Magnolia has a wider footprint on memory usage as well, and this is connected to the use the implementation makes of other open source libraries. Primetime Portal uses a handful of libraries while Magnolia's library directory contains the grand total of 25 third party components. The negative side about Primetime Portal is that most of the functionality is built in-house. Where Magnolia makes use of Xerces' XML-parsing functionality, Primetime Portal has its own implementation. The home-grown solution is stream-lined and relatively fast, but very hard to re-use in new modules.

## 5.4 The Paradox between Functionality and Extensibility

Primetime Portal's development has been pushed forward by its customers. Whenever there is a new requirement and the customer is willing to pay, the functionality is sown into the existing code-base of the project.

The insight into Primetime Portal suggests that this improvement of functionality reduces the extensibility of the product. More code makes it harder to read and use existing code, and every decision made in the certain case of one customer makes the module unusable in the eyes of another customer. For example, the Mesterbrevnemd had a very specific requirement that each web-shop product be editable by its distributor or producer. To make this work in Primetime Portal, the products were simply divided by distributor, and correct access privileges were granted to one distributor user for her products, respectively. For another Primetime customer, the distributor-product scheme could be completely useless, and the entire web-shop would have to be re-factored for such a purpose.

The question becomes whether the case be any different with an open system. One point was that the developers avoided changing Magnolia source directly, as any changes made here would be overwritten (or have to be dealt with) on rolling out the next version of Magnolia. As a result the extension was entirely made outside the Magnolia source in its own module.

The restriction of extensibility can be avoided by having a central architecture which is unnecessary to change. Such an architecture would have to be generic to adopt to possible use-cases (like the web-shop for instance), and that has been achieved in the JCR. The content nodes are flexible enough to be changed into web-shop products, and the developers are confident that they would withstand other WCMS extending transformations as well.

## 5.5 Proprietary Software and Open Standards

Another way to attack the question of open standards in WCMS is to ask why proprietary software uses less open standards than open source software. These are the reasons as experienced by the developer.

Proprietary developers do not use open standards because *they do not have to*. Having a closed set of developers invalidates the need to use extra-common standards. While some of the developers most likely know several applicable open standards that can be used in the project, including these might increase the learning curve for other developers. An open source project does not necessarily have the collaborative luxuries of an office with a crew of geographically concentrated people. The learning curve of joining the project must be overcome with explicit documentation, and the specification of the standards involved is a great place to start.

Proprietary software is not necessarily *shared* (Hanseth, 2002). The software is made, and put to use. Libraries (software components) of an open source project, on the other hand, might have to be used by other open source projects, and thus need standards to enforce possible interaction between projects. Proprietary software seldom has to cross borders between companies, with the exception of retailed software like operating systems, tool suites and computer games.

*It takes away the software's edge.* Using open standards gives the world a window into the code and its workings. This makes it possible for others to use or exploit functionality or the storage of the software directly instead of using the intended client software. An example is if Primetime Portal enabled WebDAV (Whitehead, 1999) to transfer content between the client and server side. Enabling this protocol on a proprietary content management system would give other software access to the content, and as proprietary CM systems are marketed, this is not always the desired result. It would be possible to migrate content away from the Primetime Portal installation, thus removing the vendor's lock-in.

*Obscurity means security.* Or does it? As well as the previous paragraph reasons to guarantee that the software vendor keeps as much of the customer's money as possible, it also gives hackers a harder time getting into the system. The security holes that can not be seen can not be exploited. However, this can be a false sense of security. Unfortunately it has acquired mythical status, and many believe that open source means insecure, while it actually means well-tested and security hole-less software (Wheeler, 2003).

To digress further on this point, ActiveX is a Microsoft specification allowing powerful functionality in Internet Explorer that can be activated across the net, but the protocol has been heavily exploited by malware, viruses and worms to such a degree that many technical administrators have disabled this feature on company computers (Solomon, 2005).

## *5.6 Advantages of Open Source WCMS*

The web-shop development projects have collected many positive experiences on working with open source and open standards. To summarize, the advantages of using an open source system utilizing open standards were found to be the following.

*Exposure*. Web content must be available to as many visitors as possible, regardless of browser and operating system. Open standards make it easier for browser windows to handle different formats of content and maximize accessibility.

*Extensibility*. The content must be available for third party software and plug-ins. As a WCMS has an infinite set of requirements which no single software company can hope to satisfy by itself. Open source software can be indefinitely modified to suit requirements, as long as there are resources for such development.

*Portability*. This goes for all kinds of server-side software. Different customers rely on different operating systems for their servers and to maximize the segment of the customer base, the software should be built on open standards to ensure platform independence. While Primetime Portal relies on running in a Linux environment with a back-end Oracle database, Magnolia can be deployed virtually any kind of operating system.

*Lock-in*. Or rather the improbability of it. To avoid locking the organization to the current data repository, the WCMS should use open standards for storage and transport. A WCMS quickly builds up a huge amount of content, and being locked to a single vendor could prove to be a gold mine for the vendor. Using open source detaches the *customer* from the WCMS vendor. Any other vendor can download the Magnolia product and deploy the customer's content on another web-site. This means that the customer is not locked into using Primetime to host their solution.

*Reusability.* Both content and functionality should be reusable in new systems. The customer might have bought expensive plug-ins and built an excessive amount of well-structured content.

Finally there is the low cost-of-entry to be considered. Many WCM efforts are tied to a low-budget process. Larger WCMS vendor might not be willing to audition their tools for small or medium sized customers free of charge, much less deploy for testing purposes.

## 5.7 Advantages of Proprietary WCMS

This far in the discussion, the majority of the points have been heavily leaned towards favoring Magnolia, so the reader might be wondering if the proprietary alternative had any edge at all over the open source solution.

The greatest advantage of Primetime Portal is the existing user- and developer base. The users know how to use it, and the developers know how its built.

It is streamlined for its purposes. While earlier argued that this restricts the extensibility of the software, it also enables more effective content management. Most of Primetime's customers use Primetime Portal as a medium for news-publishing, and the administration interface is designed in ways to make news-publishing as fast and simple as possible. Magnolia, on the other hand, is designed to give the user absolute control over the content of the web-site. Creating a news-item in Magnolia involves navigating to the correct position in the JCR, creating the new news-content node, moving it to the desired order, editing its contents, and finally activating the node so it is published to the web-site.

Another advantage of using software built in-house is control. There are no foreign community members who try to steer the direction of the project away from the intentions of the core developers. In the Magnolia community there can be instances of other project participants who have customers with different needs, and depending on their investment into the community effort, the development might focus on their requirements instead of Primetime's.

## 5.8 Some Words of Caution

Participating in an open source community feels good to the developer. They get the feeling that they are doing things the *right* way. In a community of mixed developers where software design decisions are discussed in the open, the result is usually to prefer quality and best-practice before expedience and returns. However, in the software development industry, the choice between pragmatism and scrupulosity is often weighed between profit and quality. The key to success in software development lies in finding the balance on this issue, and the I would like to offer the following advice for readers considering participation in open source projects, or use of open source products.

#### *Standard frenzy*

Standards are signs of quality, compliance, openness and re-usability. They appeal to the cooperative nature of developers as social beings, but trying to comply with all suitable standards in a project can be an eternal endeavor.

A web-page alone has many standards to choose from. It should make use of XHTML and CSS for design and layout, but it may also comply with the RDF specification, have content stored in XML, transformed into the correct XHTML by XLST, be exportable into RSS, PDF and pure text format and be available through the WebDAV protocol. The web application can comply with various specifications; the JSR-168 for Portlet-compliance, OASIS' WSRP and JSR-170 for the often mentioned JCR.

The danger is that an application can always get more *right.* It is important to remember to occasionally leave the perfectionism behind so results can be made, and the users or customers can see progress.

***License***

The description of the state of the art in open standards briefly touched upon the Free Software Foundation. These are behind the GPL license (GNU, 2005) which many developers consider a license with a viral effect since it can not be distributed with other software components that are not GPL or free software. This is a serious limitation to many commercial software producers who wish to participate in open source projects. They generally have to implement projects using permissive licenses, like the Apache and BSD licenses. These are compatible with, and can be used as components in commercial software.

Magnolia happens to be using the Limited GPL, which is an adaption of the GPL license formed to allow connection with commercial software components.

Primetime's company strategy is to act as a service provider which does not distribute software. Instead they offer hosting services, so it is actually their customers who are using the open source products. This bypasses the implications wrought by “hostile” licenses. Primetime developers merely act as servitors for their customers.

Should any company try to release or distribute their own version of Magnolia, it is highly recommended that they seriously consider the juridical implications of the LGPL and make sure that no rules are broken.

***Evangelization***

The research experienced a tendency in the Magnolia community. Most participants of the project would actively defend their commitment to the project and advocate its use whenever possible. This might be because of a cynical view – that they have everything to gain from more developers participating in the project. After all, the more able hands that join the project, the more the project will evolve and community support increase. At the same time they defend their own decision to invest time in the project. New community members early receive feedback from the others, and quickly develop a feeling of membership or ownership in the project.

This mezmerization effect is as fascinating as it is dangerous to the objectivity of a developer or researcher. When choosing between two WCMS solutions, a developer who has been active in the development of either will most likely claim that his or her solution is the best, and if any feature is lacking in their choice, it is easy to implement. It is important for a reviewer of such solutions to maintain a certain distance and remain critical of such developers' opinions.

## 5.9 Summary

The discussion has digested the experiences from the evaluations and considered the compared performances. This has produced some key points of theory and knowledge which we hope can be used in the field of web content management. The final chapter repeats the key points of the entire thesis, summarizes the findings of this chapter and finally suggests future research and improvements that can be made in field.

# 6 Conclusion

We have put two different web content management systems through extensive development with and without the use of open standards. The one system has undergone development inside a commercial company, and the other is the product of an open source community.

Based on the experiences from these two projects, we have drawn the relations between open source development, the use of open standards and WCMS requirements.

The *technical* requirements gain advantage from being met in open source projects. The same is true for the requirement of *internationalization*, as an open source project often is made as an international effort in order to attract as high number of participants as possible.

Open standards are particularly beneficial for meeting the requirements of *integration* and *accessibility*. This applies both to standards of transport and storage format.

### *Contributions*

The complex field of WCMS products has been explored. We have found the implications of doing proprietary versus open source WCMS development. The functional requirements of WCM systems have been discovered, and the non-functional requirement of *extensibility* has been singled out as the key factor to a successful WCMS.

We have developed a web-shop module for the WCMS Magnolia. We hope to be able to donate this module back to the open source community, as Primetime will continue to participate in the project.

### *Future research*

This thesis has been delimited to *web* content management systems with a particular focus on Java technology. There are a multitude of open source content management systems available, and it is tempting to continue the research into a broader field. There are two directions this research can take.

Horizontally, a broader scope of solutions can be valuated to find new requirements and refine the existing ones. This can can be furthered into either the proprietary or the open source field.

Vertically, one can set out to do more extensive evaluations of the existing solutions. Primetime Portal has become deprecated, perhaps even beyond repair, but Magnolia's development still flourishes, and it would be of great value to go on researching the potential uses of the Java Content Repository, as well as the other specifications Magnolia could make use of in the future, like Portlets, web services and business process languages.

As a final note, I hope that this thesis has opened the door between research and open source communities further. Reading about the academic attempts at creating content management systems, I have found that there are very few of these prototypes that are still in use or available. Instead of continuously trying to re-invent the concepts, I hope that WCMS researchers in the future will consider contributing their efforts to open source projects, or create new ones if suitable existing projects can not be found.

# Bibliography

Ashley, L. 2003, *Integrative Document and Content Management: Strategies for Exploiting Knowledge*, Idea Group Inc.

Beck, K. 2003, *Test-driven development: by example*, Addison-Wesley

Beck, K. 2004, *Extreme Programming Explained: Embrace Change*, Addison-Wesley

Belam, M. 2006, "Fine Tuning Your Enterprise Search - How To Get The Best Results To Your Users" [http://www.currybet.net/articles/fine_tune/index.php] Retrieved 9. April, 2006

Berners-Lee, T., Fischetti, M. 1999, *Weaving the Web: the original design and ultimate destiny of the World Wide Web by its inventor*, HarperSanFrancisco

Berners-Lee, T., Hendler, J., Lassila, O. 2001, "The Semantic Web" [http://www.w3.org/2001/sw/] Retrieved 29. April, 2006

Blood, R. 2000, "weblogs: a history and perspective" [http://www.rebeccablood.net/essays/weblog_history.html] Retrieved 30. April, 2006

Boiko, B. 2005, *Content Management Bible, 2nd Edition*, Wiley Publishing, Inc.

Boye, J. 2006, "Portals and CMS: What's the difference?" [http://www.cmswatch.com/Trends/652-Portals-and-CMS:-What's-the-difference] Retrieved 3. April, 2006

Burner, D. 2002, "Building the Mosaic: Writing tips for Content Management", conference proceedings from Reflections on Communication, IEEE

Byrne, T. 2001, "CM vs DM vs KM vs DAM vs SCM vs DRM -- Which One is Right for You?" [http://www.cmswatch.com/Feature/53] Retrieved 3. April, 2006

Byrne, T. 2006, "The CMS Report", CMS Works Inc.

Canfora, G., Manzo, S., Rollo, V. F., Villani, M. L. 2002, "ContentP2P: a peer-to-peer content management system", conference proceedings from COMPSAC'02, IEEE

Chakrabarti, S. 2002, *Mining the Web: Discovering Knowledge from Hypertext Data*, Morgan-Kaufmann Publishers

Challenger, J., Dantzig, P., Iyengar, A. 2005, "A Fragment-Based Approach for Efficiently Creating Dynamic Web Content", 2, p. 359-389

Chawner, B. , "F/OSS in the Library World: An Exploration", conference proceedings from 5-WOSSE, ACM

Claudio U. Ciborra and Associates 2000, *From Control to Drift*, Oxford University Press

Cunningham, W., Leuf, B. 2001, *The Wiki Way: Collaboration and Sharing on the Internet*, Addison-Wesley

Davenport, T. H., Prusak, L. 1998, *Working Knowledge: How Organizations Manage What They Know*, Harvard Business School Press

Day Software Holding AG. 2006, "CRX - A Full-Featured Java Content Repository" [http://www.day.com/site/en/index/products/content-centric_infrastructure/content_repository.html] Retrieved 3. April, 2006

De Argaez, E. 2003, "Broadband Usage in 2003" [http://www.internetworldstats.com/articles/art030.htm] Retrieved 3. April, 2006

Di Iorio, A., Vitali, F. 2005, "Web authoring: a closed case?", conference proceedings from HICSS-38, IEEE International

Dick, B. 2000, "You want to do an action research thesis?" [http://www.scu.edu.au/schools/gcm/ar/art/arthesis.html] Retrieved 27. April, 2006

Doyle, B. 2004, "CMS Genesis: Who Did What When?" [http://www.econtentmag.com/Articles/ArticleReader.aspx?ArticleID=6819] Retrieved 5. April, 2006

Doyle, B. 2005, "Where the Wild CMSs Are!" [http://www.econtentmag.com/Articles/ArticleReader.aspx?ArticleID=14534&AuthorID=155] Retrieved 5. April, 2006

Dudek, D. T., Wieczorek, H. A. 2003, "A Simple Web Content Management Tool as the Solution to a Web Site Redesign", conference proceedings from SIGUCCS'03, ACM

Economist 2006, "Open, but not as usual" [http://www.economist.com/business/PrinterFriendly.cfm?story_id=5624944] Retrieved 18. March, 2006

eXo Platform SARL 2006, "eXo Platform Overview" [http://www.exoplatform.org/documents/exo.site/] Retrieved 6. April, 2006

First Author 2006, "The Monster Mash-Up" [http://www.firstauthor.org/research_tools.html#Mashups] Retrieved 22. April, 2006

Floridi, L. 2005, "Stanford Encyclopedia of Philosophy: Semantic Conceptions of Information" [http://plato.stanford.edu/entries/information-semantic/] Retrieved 22. April, 2006

Forrester Research, Inc. 2001, "Managing Content Hypergrowth", The Forrester Report

Gilbane, F. 2000, "What is Content Management?", The Gilbane Report

Gilbane, F. 2003, "The Classification & Evaluation of Content Management Systems", The Gilbane Report

GNU Project 2005, "GNU General Public License" [http://www.gnu.org/licenses/gpl.html] Retrieved 27. April, 2006

GNU Project 2006, "The Free Software Definition" [http://www.gnu.org/philosophy/free-sw.html] Retrieved 23. April, 2006

Goodwin, S., Vidgen, R. 2002, "Content, content, everywhere... time to stop and think? The process of web content management", April 2002, p. 66-70

Gottlieb, S. 2005, "Content Management Problems and Open Source Solutions" [http://www.optaros.com/wp/wp_5_cms_report.shtml] Retrieved 27. April, 2005

Gottlieb, S., Wohlrapp, S. 2006, "Unleashing the Power of Open Source in Document Management" [http://www.optaros.com/wp/wp_6_os_docManagement.shtml] Retrieved 10. April, 2006

Hallikainen, P., Kivijärvi, H., Nurmimäki, K. 2002, "Evaluating Strategic IT Investments: An Assessment of Investment Alternatives for a Web Content Management System", conference proceedings from HICSS-35, IEEE International

Hanseth, O. 1998, "Inscribing behaviour in information infrastructure standards" [http://heim.ifi.uio.no/~oleha/Publications/siste.enkel.doc.html] Retrieved November 25, 2005

Hanseth, O. 2001, "Gateways - just as important as standards. How the Internet won the "religious war" about standards in Scandinavia", 14/3, p. 71-89

Hanseth, O. 2002, "From systems and tools to networks and infrastructures - from design to cultivation. Towards a theory of ICT solutions and its design methodology implications." [http://heim.ifi.uio.no/~oleha/Publications/ib_ISR_3rd_resubm2.html] Retrieved 30. April, 2006

Huang, S., Tilley, S. 2001, "Issues of Content and Structure for a Multilingual Web Site", conference proceedings from SIGDOC'01, ACM

IBM Media Relations 2005, " IBM Acquires Gluecode Software" [http://www-03.ibm.com/press/us/en/pressrelease/7658.wss] Retrieved 30. April, 2006

Iverson, S. P. 2002, "Content Management Beyond English", conference proceedings from IPCC 2002, IEEE International

Jackrabbit, Apache Software Foundation 2006, "Apache Jackrabbit - The Open Source Content Repository for Java" [http://jackrabbit.apache.org/] Retrieved 6. April, 2006

JCP - Java Community Process 2003, "JSR 168: Portlet Specification" [http://www.jcp.org/en/jsr/detail?id=168] Retrieved 27. April, 2006

JCP - Java Specification Request 2006, "JSR 170: Content Repository for Java technology API" [http://www.jcp.org/en/jsr/detail?id=170] Retrieved 28. April, 2006

Junco, N. L., Bailie, R. A. 2004, "A Case Study of Content Management", conference proceedings from IPCC 2004, IEEE

Kennedy, P. 2006, "Accessibility tips for website construction" [http://www.steptwo.com.au/papers/kmc_accessibilitytips/index.html] Retrieved 11. April, 2006

LII 2005, "Digital Millennium Copyright Act, § 1201. Circumvention of copyright protection systems" [http://www4.law.cornell.edu/uscode/html/uscode17/usc_sec_17_00001201----000-.html] Retrieved 27. April, 2006

Lin, C-W. 2004 "Content, Management, System: Aufbau eines Bewertungsprototyp für CMS aus kommunikativen Perspektiven - Content, Management, System", Freie Universität Berlin, Fachbereich Politik- u. Sozialwissenschaft

Martins, J. 2004, "The Structured-Unstructured Information Continuum" [http://www.dmgrc.com/dmg/weblog/index.php?itemid=25] Retrieved 10. April, 2006

McGovern, G. 2006, "Web navigation is about moving forward" [http://newsweaver.ie/gerrymcgovern/e_article000558500.cfm] Retrieved 9. April, 2006

McGovern, G. 2006, "Your website is for your most important customers" [http://newsweaver.ie/gerrymcgovern/e_article000562657.cfm] Retrieved 9. April, 2006

Microsoft Corp. 2006, "XML Paper Specification" [http://www.microsoft.com/whdc/xps/xpsspec.mspx] Retrieved 3. April, 2006

Mortensen, T., Walker, J. 2002, "Blogging thoughts:personal publication asan online research tool", *Researching ICTs in Context*, Intermedia/UniPub, ch. 11, p. 249-279

Netcraft 2006, "April 2006 Web Server Survey" [http://news.netcraft.com/archives/web_server_survey.html] Retrieved 10. April, 2006

Nielsen, J. 1996, "Why Frames Suck (Most of the Time)" [http://www.useit.com/alertbox/9612.html] Retrieved 22. April, 2006

Obinary, Kraft, B. 2005, "Magnolia and OpenWFE form strategic alliance" [http://www.magnolia.info/en/magnolia/about-magnolia/magnolia-in-the-press/magnolia-openwfe.html] Retrieved 30. April, 2006

Open Source Initiative 2001, "The Open Source Definition" [http://web.archive.org/web/20011019055512/opensource.org/docs/definition.html] Retrieved 23. April, 2006

The Open Source Initiative 2005, "Open Source Case for Business" [http://www.opensource.org/advocacy/case_for_business.php] Retrieved 18. March, 2006

Patching, D. 1990, *Practical soft systems analysis*, Pitman

Pelz-Sharp, A. 2006, " ECM + WCM = ?" [http://doingitbetter.blogspot.com/2006/02/ecm-wcm.html] Retrieved 2. March, 2006

Porter, R. 2003, "What is Digital Asset Management?" [http://its.psu.edu/dmr/dam.html] Retrieved 22. April, 2006

Primetime 2006, "Primetime Portal product description (Norwegian)" [http://www.primetime.no/products/portal.html] Retrieved 25. March, 2006

Raible, M. 2005, "Open Source CMS Evaluation" [http://raibledesigns.com/page/rd?entry=open_source_cms_evaluation_part] Retrieved 12. April, 2006

Raymond, E. S. 2000, "The Cathedral and the Bazaar" [http://catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/] Retrieved 3. April, 2006

Reenskaug, T. M. H. 2006, "MVC XEROX PARC 1978-79" [http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html] Retrieved March 26, 2006

Robertson, J. 2006, "Grand enterprise projects: why are we wasting our time?" [http://www.steptwo.com.au/columntwo/archives/001991.html] Retrieved 8. August, 2005

Shreves, R. 2006, "10 Popular Open Source Content Management Systems: An Overview", 1, p. 39-40

Smith, E. 2005, "A review of open source content management systems" [http://www.openadvantage.org/articles/oadocument.2005-04-19.0329097790] Retrieved 12. April, 2006

Solomon, A. 2005, "Java, ActiveX and the Virus Threat" [http://vx.netlux.org/lib/aas05.html] Retrieved 3. October, 2005

Spangler, T. 2006, "OpenDocument vs. Office Open XML" [http://www.baselinemag.com/article2/0,1540,1933690,00.asp] Retrieved 5. April, 2006

Staelin, C., Elad, M., Greig, D., Shmueli, O., Vans, M. 2004, "Biblio: Automatic meta-data extraction" [http://www.hpl.hp.com/techreports/2004/HPL-2004-190.html] Retrieved 25. August, 2005

Sun Microsystems Inc. 2002, "Core J2EE Patterns - Data Access Object" [http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html] Retrieved 27. April, 2006

Sun Microsystems Inc. 2005, "The Java Tutorial: Custom Networking" [http://java.sun.com/docs/books/tutorial/networking/index.html] Retrieved 2. February, 2006

Sun Microsystems Inc. 2005, "Sun's Open Source Initiatives" [http://www.sun.com/software/opensource/learnmore.jsp] Retrieved 3. April, 2006

W3C 2003, "SOAP Version 1.2 Part 1: Messaging Framework" [http://www.w3.org/TR/soap12-part1/] Retrieved 5. September, 2005

Walli, S. 2005, "Free and Open Source Licenses, Software Development and Distribution" [http://www.optaros.com/pdf/wp_SWalli_FOSS.pdf] Retrieved 30. April, 2006

Ward, T. 2003, "Find ROI: Measuring Intranet Investments" [http://www.prescientdigital.com/Prescient_Research/White_Papers/Finding_ROI.htm] Retrieved 30. April, 2006

Weill, P., Broadbent, M. 1998, "Creating Business Value through Information Technology & Rethinking Technology Investments: The Information Technology Portfolio", *Leveraging the new infrastructure*, HBS, ch. 1 & 2, p. 1-45

Weitzmann, L., Dean, S. E., Meliksetian, D., Gupta, K., Nianjun, Z., Wu, J. 2002, "Transforming the Content Management Process at ibm.com", Experience Design Case Study Archive

Wheeler, D. A. 2003, "Secure Programming for Linux and Unix HOWTO" [http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/index.html] Retrieved

Wheeler, D. A. 2005, "Why Open Source Software / Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers!" [http://www.dwheeler.com/oss_fs_why.html] Retrieved 3. April, 2006

Whitehead, J., Goland, Y. Y. 1999, "WebDAV: A network protocol for remote collaborative authoring on the Web", conference proceedings from ECSCW'99,

Wikipedia 2006, "About Wikipedia" [http://en.wikipedia.org/wiki/Wikipedia] Retrieved 3. April, 2006

Voras, I., Zimmer, K., Zagar, M. 2005, "Distributing Web-based Content Management System - "FERweb"", conference proceedings from ITI 2005,

The Yankee Group 2001, "Managing the Content Explosion into Content-Rich Applications", The Yankee Group Report

# Appendix

## Source Code Entries

### no.primetime.magnolia.webshop.Item

```
package no.primetime.magnolia.webshop;

import info.magnolia.cms.core.Content;

/**
 * Webshop Item. Wrapper for content
 * nodes that represent items in a webshop.
 *
 * @author Thomas Ferris Nicolaisen
 *
 */
public class Item {

	private Content content;

	public Item(Content content){
		this.content = content;
	}

	public double getPrice(){
		return content.getNodeData("price").getDouble();
	}

	public String getId(){
		return content.getUUID();
	}

	public String getHandle(){
		return content.getHandle();
	}

	public String getTitle(){
		return content.getTitle();
	}
}
```

## no.primetime.magnolia.webshop.Cart

```
package no.primetime.magnolia.webshop;

import java.util.Collection;
import java.util.HashMap;
import java.util.Iterator;

public class Cart {

      private HashMap items;

      private HashMap itemCounts;


      public Cart() {
            items = new HashMap();
            itemCounts = new HashMap();
      }

      public boolean isItemInCart(String itemId) {
            if (items.containsKey(itemId))
                  return true;
            else
                  return false;

      }

      public void addToCart(Item item, Integer itemCount) {

            if (isItemInCart(item.getId())) {

                  Integer oldCountInt = ((Integer)
itemCounts.get(item.getId()));

                  Integer newCount = new Integer(itemCount.intValue() +
oldCountInt.intValue());

                  itemCounts.put(item.getId(), newCount);
            }

            else {
                  // Item is not in cart, add it
                  items.put(item.getId(), item);
                  itemCounts.put(item.getId(), itemCount);

            }

      }

      public void setItemCount(String itemId, Integer itemCount) {
            itemCounts.put(itemId,itemCount);
      }

      public void removeFromCart(String itemId){
            items.remove(itemId);
            itemCounts.remove(itemId);
      }
      public void emptyCart(){
            items.clear();
            itemCounts.clear();
      }

      public HashMap getItemCountsMap(){
            return itemCounts;
      }

      public Collection getItemCounts() {
            return itemCounts.values();
      }

      public Collection getItems() {
            return items.values();
      }

      public double getSum(){
            double sum = 0.0;

            for(Iterator it = items.values().iterator(); it.hasNext();){
                  Item item = (Item)it.next();
                  Integer itemCount =
(Integer)itemCounts.get(item.getId());
                  double itemTypeSum =
item.getPrice()*itemCount.doubleValue();

                  sum+=itemTypeSum;
            }

            return sum;
      }

      public int getTotalItemCount(){
            int totaltItemCount = 0;

            for(Iterator it =
itemCounts.values().iterator();it.hasNext();){
                  Integer itemCount = (Integer)it.next();
                  totaltItemCount+=itemCount.doubleValue();
            }

            return totaltItemCount;
      }

}
```

## cart.jsp (Acceptas)

```
<%@ page contentType="text/html;charset=WINDOWS-1252" %>
<%@ page import="java.util.*" %>
<%@ page import="no.inn.acceptas.web.Cart" %>
<%@ page import="no.inn.acceptas.service.Catalog" %>
<%@ page import="no.inn.acceptas.elements.ASProduct" %>
<%@ taglib uri="http://java.sun.com/jstl/core_rt" prefix="c" %>

<%
      Cart cart = (Cart)session.getAttribute("cart");
      Catalog catalog = (Catalog)session.getAttribute("catalog");

      //if productId in parameters, add it to the cart
      String productId = request.getParameter("productId");
      String quantity = request.getParameter("quantity");
      String remove = request.getParameter("remove");
      if(productId!=null){
            //Either remove or add

            int productIdInt = Integer.parseInt(productId);
            ASProduct product = catalog.getProduct(productIdInt);

            if(remove!=null){
                  cart.remove(product);
            }
            else if(quantity!=null){
                  int quantityInt = Integer.parseInt(quantity);
                  cart.add(product,quantityInt);
            cart.updateSum();
            }
      }

%>
<c:set var="cart" value="${sessionScope.cart}"/>
<c:forEach var="item" items="${cart.items}">
      <a href="product.jsp?productId=${product.id}"><c:out
value="${item.product.name}"/></a>
      <a href="product.jsp?productId=${product.id}"><c:out
value="${item.product.id}"/></a>
      <c:out value="${item.product.shortDescription}"/>
      <c:out value="${item.quantity}"/>
      Product price <c:out value="${item.product.price}"/>
      Price of these products <c:out value="${item.sum}"/>
      <a href="cart.jsp?productId=${item.product.id}&remove=yes">Remove
from cart</a>
</c:forEach>
Price of entire cart <c:out value="${cart.sum}"/>
without tax: <c:out value="${cart.sum - cart.sum/1.25}"/>
<a href="order.jsp"><b>Check out cart</b></a>
<a href="front.jsp"><b>Order more</b></a>
Product of the month: <a
href="product.jsp?productId=${catalog.productOfTheMonth.id}">
<c:out value="${catalog.productOfTheMonth.name}"/>
<img
src="http://images.inn.no/mesterbrev/${catalog.productOfTheMonth.imageId}.j
pg"/></a>
```

## cartMainColumn.jsp (Magnolia)

```
<jsp:root version="1.2"
xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:pt="urn:jsptld:pt-taglib"
xmlns:cms="urn:jsptld:cms-taglib"
xmlns:cmsu="urn:jsptld:cms-util-taglib"
xmlns:c="urn:jsptld:http://java.sun.com/jsp/jstl/core">
<jsp:directive.page import="info.magnolia.cms.core.Content" />

<jsp:directive.page import="java.util.Iterator" />
<jsp:directive.page import="java.util.HashMap" />
<jsp:directive.page import="no.primetime.magnolia.webshop.Cart" />
<jsp:directive.page import="no.primetime.magnolia.webshop.Item" />

<jsp:useBean id="cart" class="no.primetime.magnolia.webshop.Cart"
scope="session"/>

<h1><cms:out nodeDataName="title"/></h1>
<br/>

    <!--Cart item is a touple of itemId and itemCount/qty.)-->
<c:forEach var="item" items="${cart.items}">
    <p>
    <a
href="${pageContext.request.contextPath}${item.handle}.html">${item.title}<
/a>
    Stykkpris: <pt:currency amount="${item.price}" language="no"
country="NO"/><br/>
    Pris: <pt:currency amount="${item.price*cart.itemCountsMap[item.id]}"
language="no" country="NO"/>
    </p>
    <form action="${pageContext.request.contextPath}${actpage.handle}.html"
method="post">
            <input type="hidden" name="setItemCount" value="true"/>
            <input type="hidden" name="itemId" value="${item.id}"/>
            <input type="text" name="itemCount"
value="${cart.itemCountsMap[item.id]}"/>
            <input type="submit" value="Nytt antall"/>
    </form>

    <form action="${pageContext.request.contextPath}${actpage.handle}.html"
method="post">
        <input type="hidden" name="removeFromCart" value="true"/>
        <input type="hidden" name="itemId" value="${item.id}"/>
        <input type="submit" value="Fjern fra handlevogn"/>
    </form>



   <hr/>
<br/>
</c:forEach>
    <p>
    <strong>Sum for hele handlekurven:
            <pt:currency amount="${cart.sum}" language="no" country="NO"/>
```

```
      </strong>
      </p>

    <form action="${pageContext.request.contextPath}${actpage.handle}.html"
method="post">
            <input type="hidden" name="emptyCart" value="true"/>
            <input type="submit" value="TÃ¸m handlekurv"/>
    </form>

    <form action="checkout.html" method="post">
      <input type="submit" value="Til utsjekking"/>
    </form>

</jsp:root>
```

## cartPreview.jsp (Magnolia)

```
<jsp:root version="1.2" xmlns:jsp="http://java.sun.com/JSP/Page"
      xmlns:pt="urn:jsptld:pt-taglib"
      xmlns:cms="urn:jsptld:cms-taglib"
      xmlns:cmsu="urn:jsptld:cms-util-taglib"
      xmlns:c="urn:jsptld:http://java.sun.com/jsp/jstl/core">
      <jsp:directive.page import="no.primetime.magnolia.webshop.Cart" />
      <jsp:directive.page import="no.primetime.magnolia.webshop.Item" />
      <jsp:directive.page import="info.magnolia.cms.core.HierarchyManager"
/>
      <jsp:directive.page
import="info.magnolia.cms.security.SessionAccessControl" />

      <jsp:useBean id="cart" class="no.primetime.magnolia.webshop.Cart"
scope="session"/>

<jsp:scriptlet>
/*
 * This JSP contains some logic for adding and removing items in the cart
 */
if (request.getParameter("addToCart")!=null) {
      Integer itemCount = new Integer(request.getParameter("itemCount"));
      String itemId = request.getParameter("itemId");
      HierarchyManager hm =
SessionAccessControl.getHierarchyManager(request);
      Item item = new Item(hm.getContent(itemId));
      cart.addToCart(item, itemCount);
}

else if(request.getParameter("setItemCount")!=null){
      String itemId = request.getParameter("itemId");
      Integer itemCount = new Integer(request.getParameter("itemCount"));
      cart.setItemCount(itemId,itemCount);
}

else if(request.getParameter("removeFromCart")!=null){

      String itemId = request.getParameter("itemId");
      cart.removeFromCart(itemId);
}

else if(request.getParameter("emptyCart")!=null){
      cart.emptyCart();
}
</jsp:scriptlet>

      <c:set scope="page" var="contextPath"
            value="${pageContext.request.contextPath}" />

      <table border="0" cellpadding="1" cellspacing="0" bgcolor="#daa520"
            width="275">
            <tr>
                  <td>
                  <table border="0" cellpadding="0" cellspacing="0"
bgcolor="white"
                        width="273">
                        <tr>
                              <td valign="top" width="10"></td>
                              <td valign="top">
                              <div align="left">
                              <table width="100%" border="0" cellspacing="0"
cellpadding="0">
                                    <tr height="10">
                                          <td valign="middle" width="126"
height="10"></td>
                                          <td valign="middle" width="127"
height="10"></td>
                                    </tr>
                                    <tr>
                                          <td width="126"><a
href="${contextPath}/webshop/cart.html"> <img

      src="${contextPath}/docroot/primetime/imgs/cart.gif" width="25"
                                                height="18"
align="absmiddle" border="0" alt="" /> <font
                                                face="Verdana,Arial,sans-
serif" size="1"> Handlekurv </font> </a>
                                          </td>
                                          <td width="127"><a
href="${contextPath}/webshop/checkout.html"> <img

      src="${contextPath}/docroot/primetime/imgs/checkout.gif"
                                                width="25" height="17"
align="absmiddle" border="0" alt="" /> <font
                                                face="Verdana,Arial,sans-
serif" size="1"> Til kassen </font> </a>
                                          </td>
                                    </tr>
                                    <tr height="21">
                                          <td colspan="2" width="253"
height="21"><img


src="${pageContext.request.contextPath}/docroot/primetime/imgs/dots.gif"
                                                alt="" height="5"
width="254" border="0" /></td>
                                    </tr>
                                    <tr>
                                          <td width="126"><c:if
test="${cart.totalItemCount==0}">Ingen varer i kurven</c:if>
```

```
                                          <c:if
test="${cart.totalItemCount==1}">${cart.totalItemCount} vare i
kurven</c:if>
                                          <c:if
test="${cart.totalItemCount>1}">${cart.totalItemCount} varer i
kurven</c:if>
                                          </td>

                                          <td width="127"><font size="1"
face="Verdana,Arial,sans-serif"><b>Sum: </b>
                                          </font><font size="1"
face="Verdana,Arial,sans-serif"
                                                color="#b22222">
<pt:currency amount="${cart.sum}" language="no" country="NO"/> </font></td>
                                    </tr>
                                    <tr height="21">
                                          <td colspan="2" width="254"
height="21"><img

      src="${contextPath}/docroot/primetime/imgs/dots.gif" alt=""
                                                height="5" width="254"
border="0" /></td>
                                    </tr>

                                    <tr height="21">
                                          <td colspan="2" width="254"
height="21"><img

      src="${pageContext}/docroot/primetime/imgs/dots.gif" alt=""
                                                height="5" width="254"
border="0" /></td>
                                    </tr>

                              </table>
                              </div>
                              </td>
                              <td valign="top" width="10"></td>
                        </tr>
                  </table>
                  </td>
            </tr>
      </table>

</jsp:root>
```

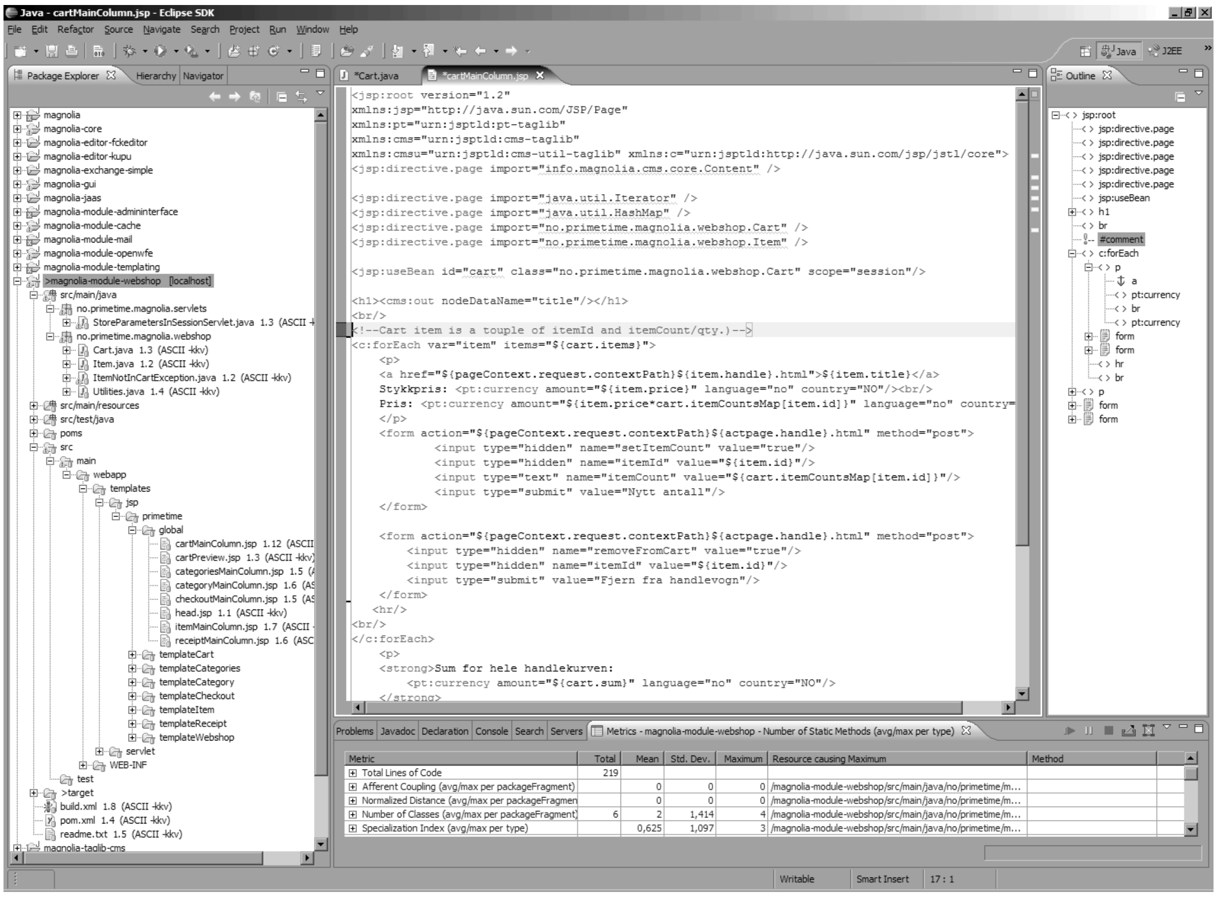
## Currency tag descriptor

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
        PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2//EN"
    "http://java.sun.com/dtd/web-jsptaglibrary_1_2.dtd">
<taglib>
    <tlib-version>2.1</tlib-version>
    <jsp-version>1.2</jsp-version>
    <short-name>pt</short-name>
    <uri>pt-taglib</uri>
    <description>Tag library for magnolia-module-webshop</description>
    <tag>
```

```
        <name>currency</name>
        <tag-class>no.primetime.magnolia.webshop.taglibs.Currency</tag-
class>
        <body-content>EMPTY</body-content>
        <display-name>currency</display-name>
        <description>
            Converts a double to a string with the appointed locale's
currency format
        </description>
        <attribute>
            <name>amount</name>
            <required>true</required>
            <rtexprvalue>true</rtexprvalue>
            <type>double</type>
            <description>The amount to be converted</description>
        </attribute>
        <attribute>
            <name>country</name>
            <required>true</required>
            <rtexprvalue>true</rtexprvalue>
            <description>Country of the intended locale, two-character
uppercase country code</description>
        </attribute>
        <attribute>
            <name>language</name>
            <required>true</required>
            <rtexprvalue>true</rtexprvalue>
            <description>Language of the intended locale, two-character
lowercase language code</description>
        </attribute>
        <example>
            <![CDATA[
<pt:currency amount="300.0" country="NO" language="no"/>
]]>
        </example>
    </tag>

</taglib>
```

## no.primetime.magnolia.webshop.taglibs.Currency

```
package no.primetime.magnolia.webshop.taglibs;

import java.io.IOException;

import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.TagSupport;

import org.apache.commons.lang.StringUtils;
import org.apache.log4j.Logger;

import java.text.NumberFormat;
import java.util.Iterator;
import java.util.Locale;


public class Currency extends TagSupport {
      /**
       * Generated by Eclipse
       */
      private static final long serialVersionUID = -3600279660989796201L;

      private static Logger log = Logger.getLogger(Currency.class);

    private String country;

    private String language;

    private Double amount;

      public Double getAmount() {
            return amount;
      }

      public void setAmount(Double amount) {
            this.amount = amount;
      }

      public String getCountry() {
            return country;
      }

      public void setCountry(String country) {
            this.country = country;
      }

      public String getLanguage() {
            return language;
      }

      public void setLanguage(String language) {
            this.language = language;
      }

      public int doEndTag() throws JspException {
            this.display();
            this.amount=null;
            this.language=null;
            this.country=null;
            return EVAL_PAGE;
      }

      public int doStartTag() throws JspException {

            //Uneccesary?
            String country = this.getCountry();
            String language = this.getLanguage();
            Double amount = this.getAmount();

            return SKIP_BODY;
      }

    protected void display() {
        try {

            String value = StringUtils.EMPTY;
            value = formatAsCurrency(amount,language,country);

            JspWriter out = pageContext.getOut();
            try {
                out.print(value);
            }
            catch (IOException e) {
                log.debug("Exception caught: " + e.getMessage(), e);
//$NON-NLS-1$
            }
        }
        catch (Exception e) {
            log.debug("Exception caught: " + e.getMessage(), e); //$NON-
NLS-1$
        }
    }

    public void release() {
        super.release();
            this.amount=null;
            this.language=null;
            this.country=null;
    }

      /**
       * @param amount Price or amount of currency which is to be formatted
       * @param language Language of the Locale in which the format is used
       * @param country TODO Country of the Locale in which the format is
used
       * @return The price with currency code
       */
      public static String formatAsCurrency(Double amount, String language,
String country){

            Locale locale = new Locale(language, country);
            NumberFormat currencyFormatter =
NumberFormat.getCurrencyInstance(locale);
            return currencyFormatter.format(amount);

      }

      /**
       * The value added tax of Norway
       */
      public static Double VAT_NO = new Double(1.25);

      public Double getNorwegianVat(){
            return VAT_NO;
      }



}
```


Java - cartMainColumn.jsp - Eclipse SDK
File Edit Refactor Source Navigate Search Project Run Window Help
Java
J2EE
Package Explorer
Hierarchy
Navigator
*Cart.java
*cartMainColumn.jsp
Outline
Problems
Javadoc
Declaration
Console
Search
Servers
Metrics - magnolia-module-webshop - Number of Static Methods (avg/max per type)
Writable
Smart Insert
17 : 1

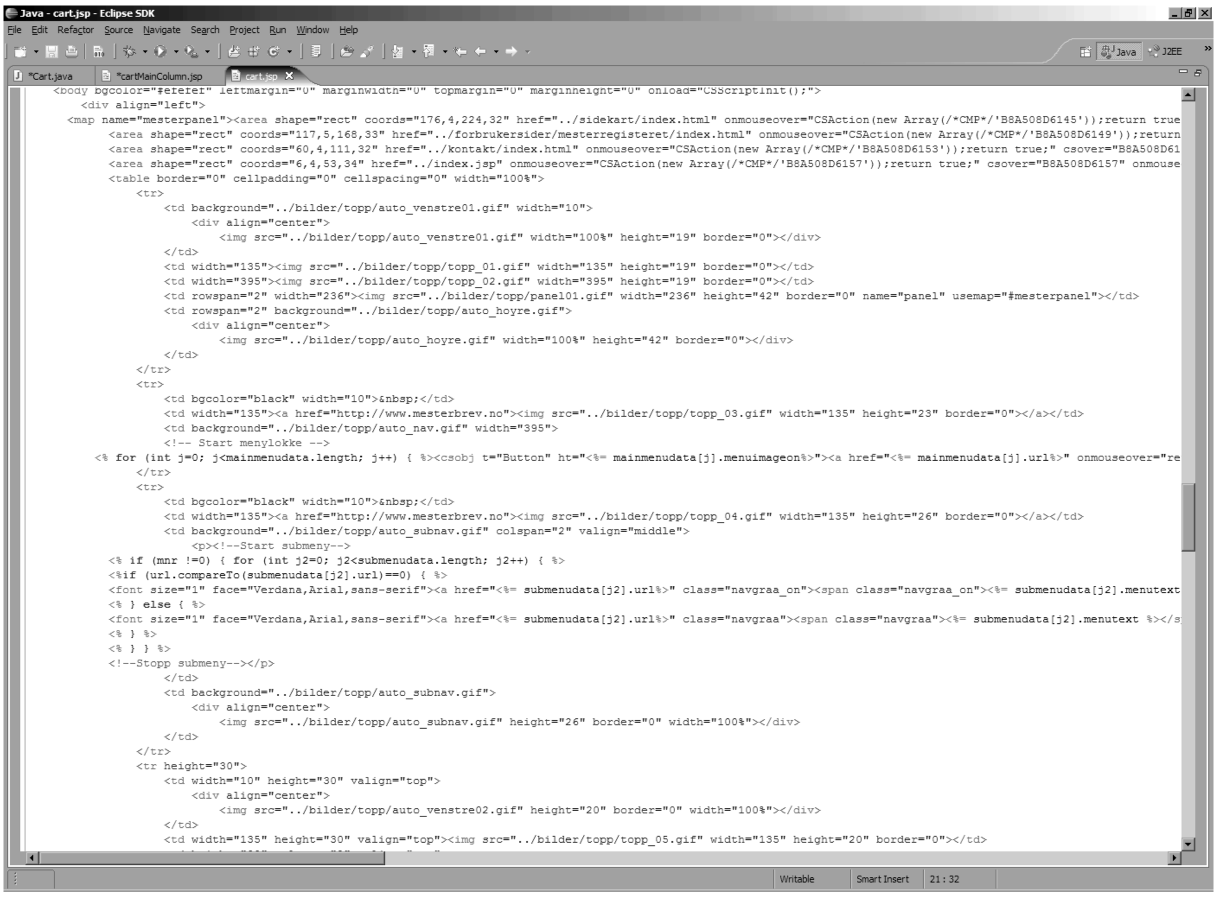
```
magnolia
magnolia-core
magnolia-editor-fckeditor
magnolia-editor-kupu
magnolia-exchange-simple
magnolia-gui
magnolia-jaas
magnolia-module-admininterface
magnolia-module-cache
magnolia-module-mail
magnolia-module-openwfe
magnolia-module-templating
>magnolia-module-webshop  [localhost]
  src/main/java
    no.primetime.magnolia.servlets
      StoreParametersInSessionServlet.java  1.3  (ASCII -
    no.primetime.magnolia.webshop
      Cart.java  1.3  (ASCII -kkv)
      Item.java  1.2  (ASCII -kkv)
      ItemNotInCartException.java  1.2  (ASCII -kkv)
      Utilities.java  1.4  (ASCII -kkv)
  src/main/resources
  src/test/java
  poms
  src
    main
      webapp
        templates
          jsp
            primetime
              global
                cartMainColumn.jsp  1.12  (ASCII
                cartPreview.jsp  1.3  (ASCII -kkv)
                categoriesMainColumn.jsp  1.5  (A
                categoryMainColumn.jsp  1.6  (AS
                checkoutMainColumn.jsp  1.5  (AS
                head.jsp  1.1  (ASCII -kkv)
                itemMainColumn.jsp  1.7  (ASCII -
                receiptMainColumn.jsp  1.6  (ASC
              templateCart
              templateCategories
              templateCategory
              templateCheckout
              templateItem
              templateReceipt
              templateWebshop
          servlet
        WEB-INF
    test
  >target
  build.xml  1.8  (ASCII -kkv)
  pom.xml  1.4  (ASCII -kkv)
  readme.txt  1.5  (ASCII -kkv)
magnolia-taglib-cms
```

```
<jsp:root version="1.2"
xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:pt="urn:jsptld:pt-taglib"
xmlns:cms="urn:jsptld:cms-taglib"
xmlns:cmsu="urn:jsptld:cms-util-taglib" xmlns:c="urn:jsptld:http://java.sun.com/jsp/jstl/core">
<jsp:directive.page import="info.magnolia.cms.core.Content" />

<jsp:directive.page import="java.util.Iterator" />
<jsp:directive.page import="java.util.HashMap" />
<jsp:directive.page import="no.primetime.magnolia.webshop.Cart" />
<jsp:directive.page import="no.primetime.magnolia.webshop.Item" />

<jsp:useBean id="cart" class="no.primetime.magnolia.webshop.Cart" scope="session"/>

<h1><cms:out nodeDataName="title"/></h1>
<br/>
<!--Cart item is a touple of itemId and itemCount/qty.)-->
<c:forEach var="item" items="${cart.items}">
    <p>
    <a href="${pageContext.request.contextPath}${item.handle}.html">${item.title}</a>
    Stykkpris: <pt:currency amount="${item.price}" language="no" country="NO"/><br/>
    Pris: <pt:currency amount="${item.price*cart.itemCountsMap[item.id]}" language="no" country=
    </p>
    <form action="${pageContext.request.contextPath}${actpage.handle}.html" method="post">
            <input type="hidden" name="setItemCount" value="true"/>
            <input type="hidden" name="itemId" value="${item.id}"/>
            <input type="text" name="itemCount" value="${cart.itemCountsMap[item.id]}"/>
            <input type="submit" value="Nytt antall"/>
    </form>

    <form action="${pageContext.request.contextPath}${actpage.handle}.html" method="post">
        <input type="hidden" name="removeFromCart" value="true"/>
        <input type="hidden" name="itemId" value="${item.id}"/>
        <input type="submit" value="Fjern fra handlevogn"/>
    </form>
   <hr/>
<br/>
</c:forEach>
    <p>
    <strong>Sum for hele handlekurven:
        <pt:currency amount="${cart.sum}" language="no" country="NO"/>
    </strong>
```

```
jsp:root
  jsp:directive.page
  jsp:directive.page
  jsp:directive.page
  jsp:directive.page
  jsp:directive.page
  jsp:useBean
  h1
  br
  #comment
  c:forEach
    p
      a
      pt:currency
      br
      pt:currency
    form
    form
    hr
    br
  p
  form
  form
```

| Metric | Total | Mean | Std. Dev. | Maximum | Resource causing Maximum | Method |
|---|---|---|---|---|---|---|
| Total Lines of Code | 219 | | | | | |
| Afferent Coupling (avg/max per packageFragment) | | 0 | 0 | 0 | /magnolia-module-webshop/src/main/java/no/primetime/m... | |
| Normalized Distance (avg/max per packageFragmen | | 0 | 0 | 0 | /magnolia-module-webshop/src/main/java/no/primetime/m... | |
| Number of Classes (avg/max per packageFragment) | 6 | 2 | 1,414 | 4 | /magnolia-module-webshop/src/main/java/no/primetime/m... | |
| Specialization Index (avg/max per type) | | 0,625 | 1,097 | 3 | /magnolia-module-webshop/src/main/java/no/primetime/m... | |

*Figure 15: Working with Magnolia templates*


Java - cart.jsp - Eclipse SDK
File Edit Refactor Source Navigate Search Project Run Window Help
Java
J2EE
*Cart.java
*cartMainColumn.jsp
cart.jsp
Writable
Smart Insert
21 : 32

```
    <body bgcolor="#efefef" leftmargin="0" marginwidth="0" topmargin="0" marginheight="0" onload="CSScriptInit();">
        <div align="left">
      <map name="mesterpanel"><area shape="rect" coords="176,4,224,32" href="../sidekart/index.html" onmouseover="CSAction(new Array(/*CMP*/'B8A508D6145'));return true
            <area shape="rect" coords="117,5,168,33" href="../forbrukersider/mesterregisteret/index.html" onmouseover="CSAction(new Array(/*CMP*/'B8A508D6149'));return
            <area shape="rect" coords="60,4,111,32" href="../kontakt/index.html" onmouseover="CSAction(new Array(/*CMP*/'B8A508D6153'));return true;" csover="B8A508D61
            <area shape="rect" coords="6,4,53,34" href="../index.jsp" onmouseover="CSAction(new Array(/*CMP*/'B8A508D6157'));return true;" csover="B8A508D6157" onmouse
            <table border="0" cellpadding="0" cellspacing="0" width="100%">
                <tr>
                    <td background="../bilder/topp/auto_venstre01.gif" width="10">
                        <div align="center">
                            <img src="../bilder/topp/auto_venstre01.gif" width="100%" height="19" border="0"></div>
                    </td>
                    <td width="135"><img src="../bilder/topp/topp_01.gif" width="135" height="19" border="0"></td>
                    <td width="395"><img src="../bilder/topp/topp_02.gif" width="395" height="19" border="0"></td>
                    <td rowspan="2" width="236"><img src="../bilder/topp/panel01.gif" width="236" height="42" border="0" name="panel" usemap="#mesterpanel"></td>
                    <td rowspan="2" background="../bilder/topp/auto_hoyre.gif">
                        <div align="center">
                            <img src="../bilder/topp/auto_hoyre.gif" width="100%" height="42" border="0"></div>
                    </td>
                </tr>
                <tr>
                    <td bgcolor="black" width="10">&nbsp;</td>
                    <td width="135"><a href="http://www.mesterbrev.no"><img src="../bilder/topp/topp_03.gif" width="135" height="23" border="0"></a></td>
                    <td background="../bilder/topp/auto_nav.gif" width="395">
                    <!-- Start menylokke -->
          <% for (int j=0; j<mainmenudata.length; j++) { %><csobj t="Button" ht="<%= mainmenudata[j].menuimageon%>"><a href="<%= mainmenudata[j].url%>" onmouseover="re
                </tr>
                <tr>
                    <td bgcolor="black" width="10">&nbsp;</td>
                    <td width="135"><a href="http://www.mesterbrev.no"><img src="../bilder/topp/topp_04.gif" width="135" height="26" border="0"></a></td>
                    <td background="../bilder/topp/auto_subnav.gif" colspan="2" valign="middle">
                        <p><!--Start submeny-->
            <% if (mnr !=0) { for (int j2=0; j2<submenudata.length; j2++) { %>
            <%if (url.compareTo(submenudata[j2].url)==0) { %>
            <font size="1" face="Verdana,Arial,sans-serif"><a href="<%= submenudata[j2].url%>" class="navgraa_on"><span class="navgraa_on"><%= submenudata[j2].menutext
            <% } else { %>
            <font size="1" face="Verdana,Arial,sans-serif"><a href="<%= submenudata[j2].url%>" class="navgraa"><span class="navgraa"><%= submenudata[j2].menutext %></s
            <% } %>
            <% } } %>
            <!--Stopp submeny--></p>
                    </td>
                    <td background="../bilder/topp/auto_subnav.gif">
                        <div align="center">
                            <img src="../bilder/topp/auto_subnav.gif" height="26" border="0" width="100%"></div>
                    </td>
                </tr>
                <tr height="30">
                    <td width="10" height="30" valign="top">
                        <div align="center">
                            <img src="../bilder/topp/auto_venstre02.gif" height="20" border="0" width="100%"></div>
                    </td>
                    <td width="135" height="30" valign="top"><img src="../bilder/topp/topp_05.gif" width="135" height="20" border="0"></td>
```

*Figure 16: Working with Primetime Portal templates*