A Case Study of a product software vendor's Customer Configuration Updating Process: GX Creative Online Development

Slinger Jansen

Sjaak Brinkkemper

Jurriaan Souer

Department of Information and Computing Sciences,
Utrecht University

Technical Report UU-CS-2006-044

www.cs.uu.nl

ISSN: 0924-3275

A Case Study of a product software vendor's Customer Configuration Updating Process: GX Creative Online Development

Slinger Jansen
Utrecht University
s.jansen@cs.uu.nl

Sjaak Brinkkemper
Utrecht University
s.brinkkemper@cs.uu.nl

Jurriaan Souer
GX
jurriaan.souer@gx.nl

September 1, 2006

Abstract

This case study report describes the results of a case study of GX creative online development, a Dutch product software vendor, into its customer configuration updating (CCU) process. The aim of the research was to document the CCU process of GX, and to uncover strengths and weaknesses. The case study was performed by doing interviews and examining development documentation, software, and internally used tools. The results of the case study are organizational descriptions, software descriptions, and the descriptions of the CCU and development processes. The main conclusions of the case study are twofold. First the case study shows that explicit software knowledge management can automate parts of the CCU process. Secondly, the case study uncovers complex issues that are specific to content management system vendors.

1 Introduction

Product software vendors encounter many problems when attempting to improve the customer configuration updating process of their product software. To date product software is a packaged configuration of software components or a software-based service, with auxiliary materials, which is released for and traded in a specific market [1]. Customer configuration updating is defined as the combination of the vendor side release process, the product or update delivery process, the customer side deployment process, and the activation process [2]. To begin with, these processes are themselves highly complex considering vendors have to deal with multiple revisions, variable features, different deployment environments and architectures, different distribution media, and dependencies on external products [3]. Also, there are not many tools available that support the delivery and deployment of software product releases that are generic enough to accomplish these tasks for any product [4]. Finally, CCU is traditionally not seen as the core business of product software vendors, and seemingly does not add any value to the product, making product software vendors reluctant to improve CCU.

A number of sources show that CCU is often underestimated and requires more attention in the quickly changing software industry. First, the quality of deployment and upgrade processes can increase customer perceived quality of a software product significantly [5], making it important that these processes are managed explicitly. Also, field research has shown that by explicit management of CCU, software vendors are able to handle large amounts of customers [6]. Finally, Niessink et al. have shown that the development of software should be seen as product development, whereas maintenance should be seen as a customer service, thereby improving customer interaction [7], the latter being stressed again by the introduction of the Software Maintenance Maturity Model [8].

To alleviate this problem we propose to manage software knowledge explicitly, by storing all facts about all product artefacts together with their relevant attributes, relations and constraints in a software knowledge base (SKB). In this way, high-quality software configurations can be calculated automatically from a small set of key parameters. It also becomes possible to pose 'what-if' questions about necessary or future upgrades of a customer's configuration. This report describes a case study of the product software vendor GX to discover whether the concepts of the SKB are used or possible to implement in practice to automate the CCU process. The case study is part of a set of case studies we have performed at a number of product software vendors into the CCU process.

GX is a rapidly expanding company in The Netherlands that produces a content management system called WebManager (WM) for medium to large customers and provides services to those customers using WM, such as implementing and updating their web sites and web applications. The case study research approach is based on software study, documentation study, direct observations at the vendor, and a large number of interviews with GX employees. The case study has been undertaken using a case study protocol and a case study database.

The contribution of this report is threefold. First, it describes the Deliver project, the case study approach, and the validity threats. Secondly, this report describes the CCU and development processes of GX for its product WM. Thirdly, this report displays what concepts of the SKB have been implemented by GX to automate the CCU process and what other improvements are possible and feasible.

The report is structured as follows. Section 2 describes the aims of the Deliver research project in general, the aims of this case study, and the validity of the case study. Section 3 describes the organizational structure of GX, their products, the content management systems market, and gives a broad outline of the development processes of GX. Section 4 provides a detailed description of the development and release processes for WebManager (WM). Section 5 describes how WM is delivered and deployed at the customer's site. Section 6 discusses the processes GX implements, their strengths and weaknesses, and the opportunities for improvement of CCU within GX. Finally, we present our conclusions in Section 7.

2 Research Approach to the Case Study

This section describes the case study of GX. The case study, performed by the Deliver research group, attempts to identify potential problems in the field of software release and deployment, by researching aspects of GX CCU and their development processes. This section first describes the conceptual model around which the case study is centred, and how it relates to the Deliver aims. The section continues by describing the aims of the case study and the actual research in more detail. To conclude, this section describes the validity of the case study.

2.1 Deliver

The Deliver project studies the CCU and development processes of software. The research project is funded by NWO Jacquard and is based at the Centrum voor Wiskunde en Informatica (CWI) and at Utrecht University. The main aim of Deliver is to ease software release and deployment effort by managing software knowledge explicitly. The areas of research for the Deliver project are configuration management, product software, software distribution, and software deployment. To manage CCU effectively, it is essential to carefully administrate the specific combination of standard components and tailored components. One part of the Deliver project focuses on product software vendors and their CCU processes. This case study is a part of that research.

2.2 Definitions and Conceptual Model

The Deliver team has a vision of what CCU should look like for a product software vendor. This view is displayed in the CCU model shown in figure 1 and shows the interaction between a customer (right) and a vendor (left). The model is based on the states a customer goes through, since the processes in the model are triggered by customer actions. These actions are becoming aware of, downloading, deploying, reconfiguring, activating, and deactivating the release. When a vendor receives a customer request, the customer relationship management (CRM) system is used to identify the customer. The vendor then handles the request and interacts with the customer. The customer moves through a number of states when about to update its configuration. At first the customer is unaware of the update, until the customer requests information about a product. Once received, the customer downloads, deploys and activates it for use, in the mean time communicating with the vendor in the form of software, licenses, feedback, and product knowledge. The presented model provides four process areas: release, delivery, activation and usage, and deployment. The process areas are separated by dotted lines in figure 1. These process areas are described in more detail in [9].

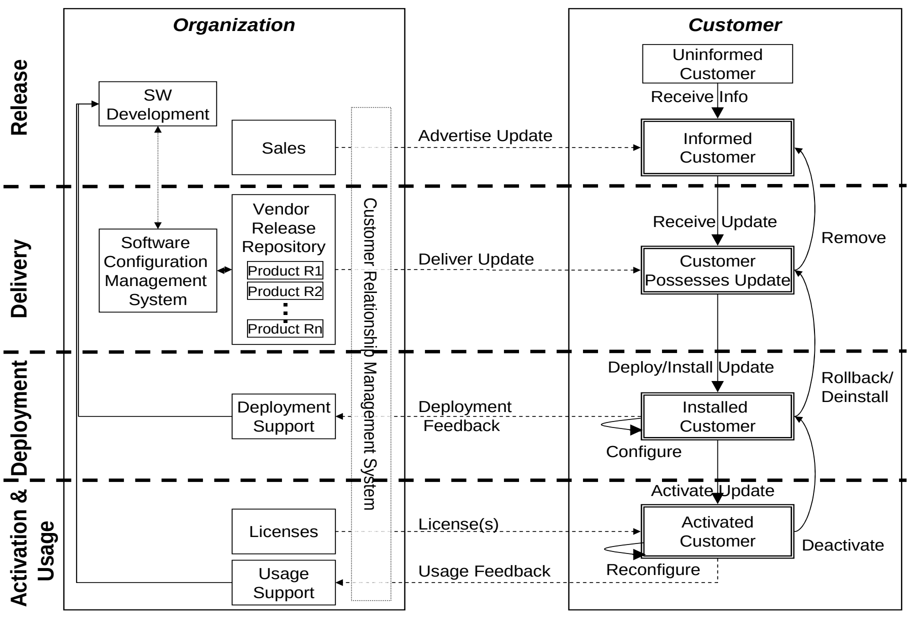

Figure 1: CCU Model

CCU can be automated by explicitly managing the knowledge that is required in these processes [6, 10]. To explicitly manage such knowledge we propose a distributed software knowledge base that is used to support the CCU and development processes. A simplified version of the distributed software knowledge base is displayed in figure 2. There a product software vendor and its customers are displayed, where the customers possess subsets of the components and knowledge about these components offered by the vendor.

Hoek et al. [11] define software release as "to package and make a software system available to a customer." Looking at this definition, the process of releasing is not only the finishing step of development, but also packaging customer-specific installations. According to Hall et al. [12], the deployment process potentially contains the "delivery, assembly, and maintenance of a particular [installed] software system at a site." The deployment process description thus consists of detailed descriptions of the processes of delivery, assembly, and maintenance. By implementing the SKB, many tasks that are part of the processes of release and deployment will be supported, so they can be done (semi-)automatically. Performing these tasks,

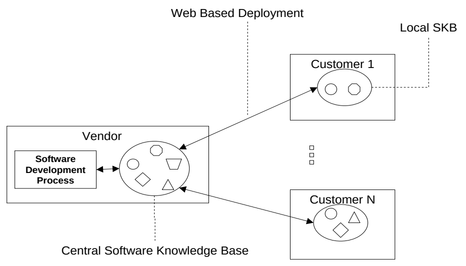

Figure 2: The Deliver View of Software Delivery

such as consistent and complete deployment of components, will alleviate the workload on the product software vendor and reduce the complexity of the release and deployment processes, thereby decreasing time-to-market.

2.3 Software Knowledge Base

The Software Knowledge Base (SKB) [13] stores exhaustive information on software artefacts and assists the product software vendor in many ways during development. The SKB implements features from product data management (PDM) [6], software updaters [4], software deployment tools [14], and software configuration management (SCM) [15]. In this chapter the features implemented in our SKB that are relevant to this case study are explained.

2.3.1 Structure

The SKB supports the processes of release and deployment. Each of these processes poses different challenges for an implementation of an SKB. Support for the release process with the SKB is founded on the idea that the artefacts of every software system are stored in some kind of repository for development. The SKB assists developers performing development tasks, such as version tracking, release storage, and component composition by establishing new relationships between components. These operations require knowledge about components the relationships among them [16].

Parts of the knowledge stored in the SKB can be published once a release is performed. This knowledge is publicly available for the SKB to assist the process of delivery and deployment. The delivery process contains processes such as media creation, web delivery, and ‘what-if’ query handling. Finally, to assist and support the deployment of software at a customer site, a customer side knowledge base holds all the information on deployed software. This part of the SKB assists the customer in finding out what components are required to get functionality not yet deployed on the customers’ machine.

2.3.2 Usage

The following list summarizes some features and processes that are improved by implementing an SKB as proposed by Deliver.

- **Version Control System (VCS) -** The Deliver project is not planning to implement yet another VCS; Deliver assumes, however, that software products are stored in some kind of repository. This repository typically stores different versions of source code that can be extracted at will. Relevant to the Deliver group is whether dependencies are administered between different versions of sources in a VCS. This enables delivery of a component or software package with compatible sources, even if they are from different released software versions. The same holds for inter-component dependencies. Currently Deliver envisions an SKB that uses a feature description language (FDL) [17] to describe dependencies among components.

- **Build Systems -** The knowledge stored in the SKB can be used to build products. When a product is designed, the dependency relationships between source files can be used to assure consistency and completeness. The SKB is able provide dependency information to build systems.

- **Customer Configuration and ‘what-if’ questions -** The SKB implements mechanisms to generate an installation for a customer depending on the components the customer has installed. This means that some kind of querying mechanism is needed to generate a list of what components need to be installed at the customer site to provide features a customer has requested, or a list must be generated stating the conflicting components. The SKB should enable some kind of mechanism to do updates, and detect conflicting and inconsistent sets of components. There should also be some kind of auto update facility, so the applications based on the SKB can support push, automatic pull, and pull mechanisms.

- **Customer Installation -** When the software is delivered to the customer site, a tool that deploys, configures and possibly builds the software is needed. Rollback functionality is a feature that is required for configurability. One of the main features Deliver wishes to see in the SKB is support for web delivery. The SKB should support all features mentioned here through some kind of web interface.

- **Software update and Installer Media Creation -** The applications built around the SKB should also support some kind of installer creation. This installation package can then be transported, through some media. It should be possible to create these installers automatically.

2.4 Research Questions

The main research question is: **How do the CCU processes of a product software vendor relate to the Deliver models?**

This research question can be split into four parts:

- **Describing the processes -** To answer the research question the CCU processes of GX are modelled and described in detail. These descriptions consist of a description of the products GX sells, the development methodologies used, and finally a description of CCU.
- **Comparing the processes to the Deliver models -** Once described, the processes can be compared to the processes that are part of the CCU model.
- **Comparing the processes to previous cases -** The case study of GX is part of the broader research into software release, delivery, and deployment. Previous case studies allow for comparison between product oriented software developers and distributors and service oriented software companies. These cases consist of case studies at a large product software vendor [18] and two medium sized product software vendors [19, 20].
- **Proposing improvements to the processes -** By evaluating the similarities and differences between de CCU model and the GX CCU process descriptions potential improvements are uncovered and described.

To answer the research question a case study database and a case study protocol have been created.

2.5 Research Methods

During the case study facts have been collected to answer the research questions. The means through which the Deliver group gathered these facts were:

- **Interviews -** The main research questions have been answered in part during the interviews with the people responsible for the development and usage of the GX product. These interviews consisted of two sessions, one exploratory session and one reflective session. The reflective session was used to establish the correctness of other sources of information.
- **Studying the software -** During our research the tools used to support development, release, and deployment were studied. Tools, such as the Ant building tool, were tested, evaluated, and documented.
- **Document study -** GX provided us with documents such as software architecture descriptions and process descriptions. These documents were studied and added to our case study database.
- **Direct observations -** Direct Observations were made during our presence at GX. These direct observations were later confirmed during the interviews.

The validity threats to the case study are construct, internal, external, and repeatibility [21] threats. With respect to construct validity, the same protocol was applied to each case study, which was guarded by closely peer reviewing the case study process and database. To create a complete and correct overview, both the CCU and development processes have been documented extensively. The internal validity was threatened by incorrect facts and results from the different sources of information. By crosschecking these results and observing the processes as they were going on a complete view could be created. With respect to external validity, the case is representative for the Dutch product software vendor market domain because the general information about GX has been compared to other vendors that are active in the Platform for Productsoftware[1], an organization that aims to share knowledge between research institutes and product

[1]http://www.productsoftware.nl/

software vendors in The Netherlands, with over 100 members. The comparison shows that the case is typical for the Dutch software industry with respect to size, number of customers, and income. Finally, to defend repeatibility we would gather the same results if we redid the case study, due to the case study protocol.

2.6 Related Work

Previously we have undertaken similar research at Exact Software [18], Planon [19], and Chipsoft [20] to uncover problems in the areas of release and deployment. Other work, performed by Tiihonen et al [22], describes the state of the practice of configuration management and the deployment process in the software industry of Finland. Jaring and Bosch [23] have studied variability of MRI scanner software and is different from our work in that Jaring and Bosch do not focus on enterprise resource planning software.

In the following period we are planning to perform more case studies, at different types of companies. We are also trying to compare the results of different case studies, to generalize the results and provide generic advice to product software companies. A final contribution could be a website on which product software vendors can fill in a questionnaire to benchmark and compare their release, delivery, and deployment processes in relationship to other product software vendors.

3 GX

GX creative online development is a web technology company that focuses on content- and web management, online application development, and integration of backend systems into web portals. The services of GX include consulting, development, implementation, integration and support for interactive web applications. These services are supported by the GX WebManager (WM) product. GX attempts to find a personified solution for each customer organisation. Generally, this leads either to GX installing a local copy of WM on the customer's web server or a model where GX places a webserver at a partner of GX (currently KPN[2]).

3.1 GX Software Products and Services

GX started was founded in 1995 and builds interactive websites and webapplications for large organisations. Nowadays, GX offers one product and a range of services. Primarily, GX delivers websites and content management systems (CMS), with the possibility to communicate with legacy database management systems. GX uses its own product, WM, to deliver websites and services to customers. The services can range from site design to remote deployment and site management (not content) by GX. GX started out as a purely service oriented company. Only once the GX customer base started to grow and noticed that with the explicitly limited growth rate it could not support any more customers, GX introduced a partner program and started selling WM as a product to other implementing organisations.

3.1.1 WebManager Product

WM is a content management system developed by GX. The system has a front-end and a backend, where the backend allows designers, content managers, developers, and contributors to work on a collaborative web based environment and where the front-end displays the website and its applications to users. WM is an independent product that is built in such a way that an external company can use the product to build websites without the involvement of GX. Due to the fact that WM wants to become more product oriented and less service oriented, WM is developed in such a way that other companies, called partners, can buy WM and implement it to create websites for customers. GX does this because the websites that can be built using WM are of a much broader scope than the capabilities of one company. Partners can, for instance, specialize on the design or business integration facilities of WM.

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

3.1.2 Projects

When GX is contracted by a customer to build a website a project organisation is created by GX. Contracts specify the type of service and implementation a customer wishes to receive. As a part of the contract, licenses to WM components are purchased, such as a news component or a forum component. Once the terms have been set for a project, a newly composed team of developers is assigned to it. These developers are responsible for understanding the requirements of the customer and implementing them, using WM.

GX manages projects according to a method GX developed itself [24], based on PRINCE2[3], a typical information technology project development method. The customer and a newly created project group sign a cooperation agreement to establish the relationship between them throughout the implementation of the project. The project is divided into a set of separate phases, containing at least a requirements analysis phase, an implementation phase, and a support period. After negotiating the final requirements, the developers in the project group implement the solution the customer and GX agreed on. Once the project has been developed, the software is deployed at the customer site. After deployment it is not uncommon for the requirements to be added by the customer. It is handed over to customer support by the project manager after the customer has approved of the project, which marks the end stage of the project implementation phase. The time period from the creation of a project team to the handing over to the customer support department generally takes approximately 1.5 months. In the case of larger projects delivery cycles are extended to three months.

GX implements three types of projects for customers, which are standard, complex, and migration. A standard project is when a customer wants a website that consists completely of standard components. When a customer wants to use external datasources, product configurators, or any other type of large customisation, GX assigns a complex project to this customer. Finally, when a customer already implements WM and wants to upgrade its website, GX generally starts a migration project, to migrate from the previous site to the new one. The migration project includes and update of WM and of the look and feel of a website. Each of these projects is implemented using WM.

3.2 Partner Web

To enable growth GX that is not restricted to autonomous growth GX has introduced a partner program, which is a network of companies that can implement WM at customers. These companies buy a WM license for each customer they obtain. The partner can then deploy, implement, and support WM for customers. Partners can obtain releases of WM for development and implementation purposes through the Partnerweb, a website where partners can find WM, release notes, support information, licensing information, and everything else that is required to maintain an intensive relationship with GX's partners.

3.3 GX Employees

GX currently employs 85 people. GX employees can be divided into six categories (see Figure 3): Customer services, Product Development, Engineering Services, Sales and Marketing, Solutions Delivery, and Management. Because GX provides products and services, both a product development and project development department have been created. Both these departments are regularly supported by members from the engineering services department. The engineering services department consists of employees with flexible contracts who work for either product or project development. The Day to day business of GX is run by two business managers.

There are three groups of developers at GX, the project development group, the WM development group, and a group of developers that can work in either group, depending on the workload in either the product development or project development department. These floating developers are employees of an external company and are employed on a call basis by GX.

An interesting characteristic of GX is that its employees are largely academic graduates. It will not surprise the reader that the company (then non-profit organisation) was founded by a group of university students. Because that most people working at GX are of an academic level has some interesting effect on the company strategy, since it allows the company to retain a competitive and service oriented edge over its competitors. On the other hand the company must focus on efficient use of its highly trained and expensive human resources.

[3]http://www.prince2.com/whatisp2.html

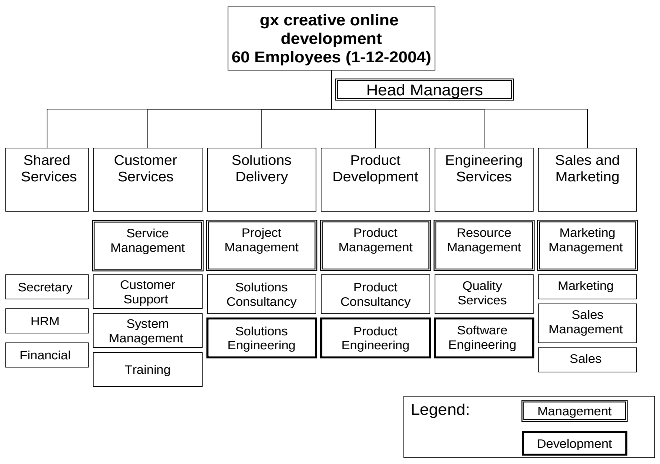

Figure 3: GX Employees and Organisation Chart (31-12-2004)

3.4 GX Customers

GX targets the medium to large business market for its customers. Some of its larger customers are soccer club Ajax, car producer Daimler-Chrystler, KPN, Planet Internet, and many government organisations in the Netherlands. Because GX traditionally was a service oriented organisation, GX still provides its services in very different environments for customers. An example of an a-typical customer hosts WM and maintains its website independently of GX and relies only on GX for new releases of WM. Other customers generally purchase GX's services to design and deliver their site as well. The customers host WM and the site themselves or at an external hosting party (such as KPN, GX's current hosting partner). In some cases, such as Planet Internet, the company is mature and experienced enough to host and maintain their own site. In other cases the company or organisation, such as the Dutch National Police Corps, uses WM for a closed off WLAN setting that is not connected to the Internet, thus not allowing a situation where GX can directly access the WM server.

3.5 GX and the Content Management Systems Market

Generally, the content management systems market is less developed with respect to CCU than the complete product software market [25]. A number of features that are appearing more and more in the product software market, such as automatic updates and automatic install scripts, have not yet penetrated the CMS market, due to the high complexity and multiplicity of web server architectures and the high level of knowledge of their maintainers. Some problems that trouble this market are automatic deployment of components, automatic configuration of components due the high level of configurability of these components, update problems, and complex system architectures due to load balancers and server parks. GX is similar to other companies in that it has to cope with these problems. On the other hand, because deployed content management systems are generally accessible through the Internet, error feedback reporting and product status reporting through the Internet are common compared to the product software market, where this aspect of customer feedback is underemphasized.

Another aspect of the CMS market is that customers generally buy a full website, instead of just buying a copy of the CMS itself. GX, much like other CMS vendors, offers it's customers a website as a product, which is supported by WM. WM as such is thus never sold separately. This influences the CCU processes in a number of interesting ways, as can be seen in the following sections.

4 GX WebManager: Development and Release

WM is the content management product built by GX. WM, currently at version 8.3, is a J2EE based product, consisting of a front-end and backend for content management. Previously, WM was built and used internally to provide services to a customer. The customer only saw the web-based user interface for editing content. However, since version 7, GX has decided to switch to a more product-oriented view. GX changed their views after seeing that specific management of WM as a product could increase GX's profitability. Before this switch, WM was seen as part of the generic development cycle. However, GX increasingly discovered that the implementation of websites and the development of a content management application with generic content components are conceptually two different things.

During the development of websites as part of GX services, GX discovered that they were commonly rebuilding certain components for websites. To reuse such components GX introduced templates. Template code elements are written in a language developed by GX [26] or in Javascript. These elements can be used and configured in WM, such that site maintainers do not need to write any template code. Next to the customisation that is offered by Javascript, other customisations can be built using application programming interfaces (APIs). These APIs provide mechanisms to build advanced functionality not yet provided by WM.

4.1 Development

The developers working on WM develop their Java and template code using Eclipse[4] and Subversion[5]. WM currently consists of 345 kloc[6]. The two types of development within GX are project development and product development.

Product developers are working on WM. The development artifacts of WM are stored in a Subversion repository. The code of WM is written in Java Eclipse. WM is developed completely along J2EE standards. **Project developers** work on a unique repository for each customer. Internal releases are constructed from the development artifacts of WM. These releases consist of an archive file with all artifacts required for using WM. All development artifacts (Java & JSP code) created during a project are stored in a separate project specific repository containing artifacts belonging to that project. Project developers have a local working directory with the artifacts from the release archive and the project artifacts. Each time project developers finish working, they commit the project artifacts they have worked on. When a developer is finished with a component or new functionality, it is tested extensively by the quality assurance members of the project. The tests are performed in a well documented order, being integration, functional, technical, documentation, and finally acceptation testing.

When a project has been tested thoroughly, project developers construct a deployable application from their work directory which is passed on to the system administrator for deployment on a server. Some customers use WM to its full extent and build custom features for their website. When this is the case and GX manages the customers' project files, GX and the customer open a shared repository on which both the developers at GX and at the customer can do operations. Such a shared repository avoids version incompatibilities and ensures that no merging of templates is required when GX performs an update of the customers installation. Development within GX is done according to incremental changes, bug fixes, and periodical releases.

4.1.1 Versioning

WM is released as shown in Figure 4. Once a major release is done, the trunk of the development tree is branched. From there on there exists a minor release branch, which is progressively tagged for each bug fix and patch. There are three different types of releases:

- **Major -** A major release comes out every 18 to 24 months. A major release, such as the transition from WM 7 to 8, is generally related to the release of any combination of a large amount of new features, architectural changes, or a strategic change of GX.

[4]http://www.eclipse.org

[5]http://Subversion.tigris.org

[6]May 18th, 2005

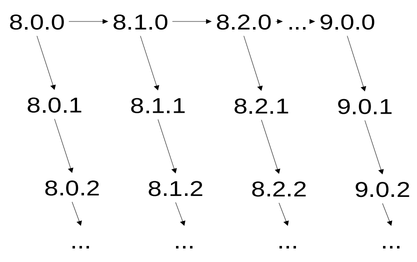

Figure 4: WebManager Versioning

- **Minor -** A minor release, which occurs approximately every 9 months within GX, adds functionality to the product without any serious changes to the architecture of WM.
- **Patch -** A patch release is made by GX whenever urgent bugs have been fixed. A patch release contains bug fixes that were fixed in the period since a previous major release.

The process of creating releases is similar for each release. First, the software is built using a standard Ant script. Another specific Ant script is used to create a WM War file. The WM War file is copied to a site on the intranet that is visible to each employee.

WM depends on a J2EE application server such as Tomcat, JRun, and JBoss to function correctly. Tomcat, which is chosen by 90% of GX customers as their webserver technology, is a Java Servlet container and web server from the Jakarta project of the Apache[7] software foundation. It can be used standalone or used behind traditional web servers such as Apache httpd, with the traditional server serving static pages and Tomcat serving dynamic servlet and JSP requests. WM uses Tomcat to display it's JSP templates. Version compatibility of Tomcat and WM versions are stored explicitly in a document freely available on the GX Intranet. This information is required when a consultant is deploying WM and Tomcat at a customer.

Since GX does not constrain their customers to work with the newest version of WM and many customers indeed work with older releases, many bugs are fixed in these earlier releases, but at the lowest release in a branch (such as 8.0.4 in our example). Approximately once a week these bug fixes in earlier releases are merged upward. Though this is a manual process that takes up to two hours, the fixes propagate upwards and remove the bug from all of the newer releases.

4.1.2 Documentation

WM product documentation is written by developers and trainers. Developers write JavaDOC API documentation and documentation for each functionality in the product, whereas documentalists and trainers write the client documentation (user manuals). To maintain integrity of the documentation and keep it up-to-date with the latest release of the software, developers need to change the documentation each time features are changed or added. The documentation is used by quality assurance to check whether the functionality and its description match. Next to this large amount of documentation stored digitally with the product, trainers write documentation by using the product. The documentation written by the developers is not available to users of the product since this documentation is intended for developers only to support development.

4.1.3 Software Architecture

WM provides two main user interfaces:

- a website front-end that displays the website to visitors
- a website backend that allows a user to edit and manage the content and some parts of the layout

[7]http://www.apache.org

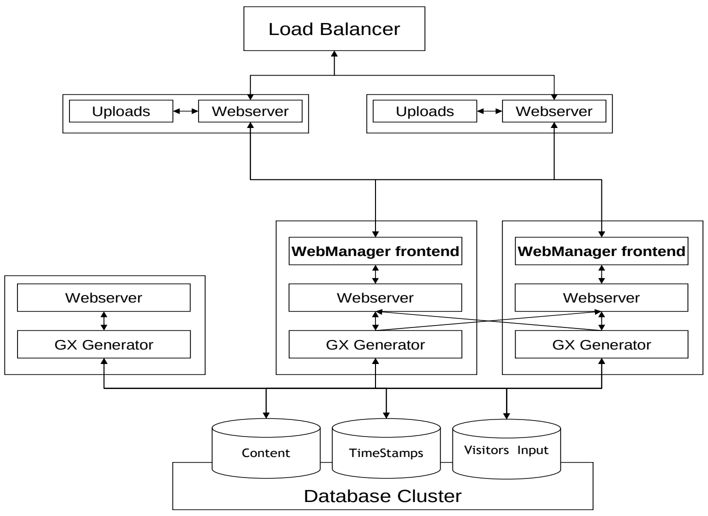

Figure 5: GX WebManager Software Architecture

These two interfaces are both accessed through a browser. WM websites consist of parameterized JSP content templates that, when combined, make up a page of a site. Such templates can be parametrized using the backend of the product to be stored in the database as content elements, which can be displayed on the site through the front-end. The template parameters, such as text and images, are stored in the WM database. A WM website that is displayed by the WM front-end thus consists of multiple content elements that can contain other content elements.

An example installation of WM is shown in figure 5. From top to bottom the picture shows a load balancer, a number of web servers, three content servers, and a database cluster. The three servers in the middle consist of two servers that generate content using the GX database for site visitors. The third one is used by content managers to access the site's backend module, to allow them to change the content of the site. Generally the backend is placed within the firewall, although sometimes the backend server is accessible from the outside through yet another web server. The arrows amongst each element in figure 5 indicate the flow of data between the different servers, software components, and databases. The three databases in the figure are used to store all the content and user data that is gathered by the content management system in operation.

The number of servers can be increased by adding servers to either pool to enable GX to support customers with large amounts of visitors and large processing loads. GX Generators can be used to generate content elements from a custom or legacy database. GX Generators are used to perform operations on external data sources through the WM interfaces. This allows GX to connect any datasource to their product, using interfaces that are custom built by GX.

4.1.4 Directory Structure

One part of WM installation is unpacking the released archive. The archive contains a number of directories, some of them contain product specific artifacts, some contain project specific artifacts, and some contain both. By dividing these artifacts GX enables separate updating of WM and the project specific files. There are instructions for employees and system managers describing how to perform such an update. An update of WM cannot be performed automatically. The War file in which WM is sent to customers must first be manually configured to contain the configuration information for a specific site. A project update can also not be done automatically, but generally is much simpler than a WM update, since it does not require any configuration steps. The new project War file simply replaces the old project War.

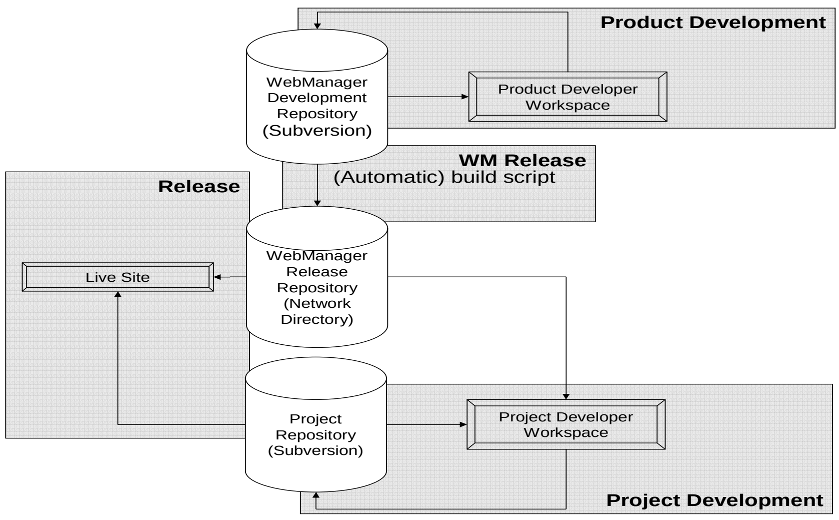
Product Development
WebManager Development Repository (Subversion)
Product Developer Workspace
WM Release
(Automatic) build script
Release
Live Site
WebManager Release Repository (Network Directory)
Project Repository (Subversion)
Project Developer Workspace
Project Development

Figure 6: Release and Development Processes of GX

4.2 Release

When WM is released it becomes available to two groups. First, project developers install the new version on their local machines as soon as it is released. This ensures that the version that is delivered to customers is always the latest release of the software available. The second group are GX's implementation partners. The release that is available to the developers through a network drive is posted onto the partnerweb to provide the implementation partners with the most recent version of the software.

Figure 6 displays the release, delivery, and deployment processes and the repositories on which they operate. The developers contribute their work to a Subversion development repository. Once a release procedure is started, an Ant script is run to create the release artifacts. These artefacts are stored on a network drive that is open to all employees of the company. In the future the release will also be published on-line such that it is accessible to partners. The release is then distributed to the customers who require an update and to each project developer, to enforce that all project developers are using and deploying the same version of WM. The project developer stores her artefacts in a project repository, sometimes even including the WM release. Once the project is (pre-)released it is burnt sent to the customer by FTP by project development, and deployed onto the customer environment by a system engineer.

5 Delivery and Deployment

5.1 Deployment Process

Before a deployment takes place, GX first has a dialogue with the customer about the infrastructure of the software. The discussion settles which database technology will be used (MySql, MS-SQL, Oracle, etc), which web server (Apache, IIS, etc), and the server setup (multiple servers vs. single server, location in the network, etc.)

On current hardware it takes two hours to deploy WM and its prerequisite components on a server. A system manager, usually a consultant from GX, installs the prerequisite components, such as JDK, Tomcat, Apache, and Ant, on the server from a CD. Once these prerequisite components have been installed and are running, the database is set up and the WM archive file is unpacked on the server. The archive file contains the WM War file, the project specific War file (containing the license.xml, stylesheets, and specific templates, for instance), and two XML configuration files, config.xml and web.xml. These two xml files specify the configuration of WM and WM's place in the infrastructure.

Now, before the WM War file can be loaded into Tomcat, the War must first be configured. The WM configuration contains such information as the locations of all the prerequisite components, the database types, and the type of web server. The configuration file is extracted from the WM War file and put back after these configuration settings have been added accordingly. Over the last year, GX has been automating this process as much as possible. However, to allow for customisation, GX applies the concept of "configuration by exception", which means that when a customer wants the default, the deployment can be done automatically. However, if a customer wishes to customize certain directories or external components, the configuration needs to be adjusted. Often the result of customisation is that deployment cannot be performed automatically.

When a deployment has been performed there is a long manual checklist to check whether the deployment is successful. These manual steps consist of many items, such as the checking of configuration directories, checking whether a webpage can be created using the software, and checking whether the different URLs work, such as "thesite.com" or "www.thesite.com".

5.2 Deployment Types

There are different deployment types. There is the live environment, the project developer environment, and a testing environment. Each of these three types is deployed in the same fashion.

On a developer environment one instance of the software is deployed after some prerequisites have been met. Developers generally need Apache, Tomcat, Ant, Eclipse, MySQL, Tortoise SVN, and the JDK, before WM can be installed. The developer usually deploys these software components onto his local workspace. After all prerequisites have been met, the developer installs WM. After WM is installed, a project developer downloads the project specific artifacts (for the project she is working on) from the projects repository and install them over the installation of WM. The developer can now access both the front-end and the backend of the project site under development on her own personal computer.

On a live environment the deployment depends on whether this is a single or distributed installation. The simplest install occurs when both the database and the server are installed on one system. Generally, however, customers have a separate database server. It is also possible to create a dedicated server for content management and a cluster of database servers. Also, once a customer receives a large amount of visitors it is possible to balance the load amongst different servers, which requires a separate deployment of the WM software on each system. This type of deployment generally takes between 2 hours and a full day. After a deployment has been done on the first server within a multiple server network infrastructure, subsequent servers take significantly less time.

The deployment on a developer platform costs little time because the developer usually already has all the prerequisite components installed. A deployment of WM on such a system takes no more than six minutes, using an Ant deployment script. The deployment on a company server, however, takes a lot more time because that all required components need to be installed first and that WM cannot be installed with all the default options set. The large amount of time to configure the product is caused by the fact that GX allows deployments in a multitude of different network infrastructures and operating platforms. Often GX creates a test environment for new customers with a standard network infrastructure with GX. The setup of such a test environment has been fully automated and takes approximately one hour.

5.2.1 Contracts and Licensing

Once a license contract has been signed that lists the purchased components, a license.xml file is created using a Java tool called the "Config Tool". This file is used by WM to supply only the purchased features. The XML file cannot be edited freely however because of a checksum in the file. There is thus only one version of WM available, in which features are activated and de-activated to fit the customer's wishes. GX is presently content with the licensing scheme. Licenses are unencoded and do not expire. A GX license basically lasts forever, but since customers need to upgrade each time they want to update large portions of their website, the license file is also updated.

5.2.2 Product Deployment and Configuration

The deployment process is a complex procedure that is executed by a GX consultant. The consultant usually brings a CD containing the latest WM release, and the components required by WM, such as Tomcat, Apache, the Java runtime environment, and if required MySql. The consultant starts by determining, with a system manager of the customer, in what directories these prerequisites are to be installed. The consultant then installs all prerequisites and WM. The total installation and configuration time is at least 2 hours, mostly because all dependencies need to be defined manually by specifying the directories of components on which the WM component depends. The setup of the database and web server also take a lot of time. In the future, partners will deploy WM in the same manner as GX.

5.2.3 Variability

WM has been tested with MS-SQL, Oracle, and MySql. Also, the product can work with any web server application that can provide static content to a site user, such as IIS or Apache. These variabilities can be bound during deployment and in special cases can be changed at a later stage.

5.2.4 Updates and Upgrades

There are different types of updates for projects and WM. A project is generally updated when a serious error has been encountered in the customized parts of a website deployment or when a customer wishes to upgrade the looks and functionality of a site. The following three different types of updates for WM are deployed when a customer wants to update their complete site or when a big problem is encountered:

- **Complete Update -** Once a new release of WM is available and there is a reason for GX to update the version of WM on a customers server(s) the complete archive is sent to the customer to be deployed by a GX employee or even one of the site administrators. Customers and GX personnel usually first test the update in a testing environment. A complete update of WM is accompanied by an update of the project specific files, since these usually require adjustments to work with the new version of WM. Complete updates can be performed from any major release to any other.
- **Patch -** Patches are distributed adjustments to a WM deployment. Patches are released in patch releases and are delivered in an archive file similar to the archive file created for a new deployment.
- **Hotfix -** Hotfixes generally contain a sole set of updated templates, or a War (the WM War or the project specific War) that needs to be installed manually by the customer. The process is customised on a per hotfix basis, since hotfixes are generally simple quickfixes that replace a small number of WM files.

An interesting fact is that both a complete update and a patch update are performed in the same manner. Usually the update is as simple as using and restarting Tomcat to reload the WM War file. An update thus takes an average of five minutes of downtime. Updates are performed on a weekly basis after a site has been deployed recently. After some months, this decreases. Some customers do not perform updates anymore since their website and content management system is working up to their expectations. GX's main problems with respect to release, delivery, and deployment can be found within the area of complete and patch updates. These problems are caused by the customisations that have been made to a customers' configurations, conversions of the datamodel, and the large amounts of different environments that are supported by GX. For more information on the releases of updates see section 4.1.1.

5.3 Customer Communication

Whenever a customer calls for support, the support engineer can find the version of the software that customer is using by going to a special information page that belongs to a site. This page lists the version number of the deployed GX version, specific to the revision date and number.

Every night, if the customer deployment has access to the internet, a comparison is made between the deployed WM files and the WM files in the GX Subversion repository using the Live Site Checker tool created by GX. Whenever a discrepancy is found a report is sent to GX since a discrepancy might mean that the system manager has been editing the configuration without permission. This method is currently not used to see whether the license.xml file has changed.

At present WM does not support the sending of usage reports. Such reports can be used to supply GX with information on the most used features of WM, but also to explore other licensing mechanisms, such as pay-per-usage. WM also does not support some type of automatic information provision, such that someone who starts up the backend of WM sees a message about a new product release being available.

Finally, GX recognizes that one of their weak points in customer communication is that it does not communicate well enough about new product releases or features. GX has lost customers due to the fact that their sales engineers did not contact the right customers at the right time. Such problems can be solved by integrating the customer relationship management system with the software configuration management system, or by using the previously mentioned product information messages.

6 Discussion

In the discussion of the GX case study we first compare GX's solutions to the SKB techniques (see Section 2.3). The strengths and weaknesses of GX are indicated with respect to their development, release, and deployment processes. Finally, it is shown how GX could, with the introduction of SKB techniques, use fewer resources and improve quality of service to customers.

First and foremost we believe that GX is a successful company in its field, and their high level of expertise (most employees possess at least a masters degree) supplies GX with a competitive edge in the CMS market. GX has recently expanded their ability to handle more customers by introducing the partner program and GX does not restrict itself to a specific area of the CMS market. We believe this to be good strategic decisions, also because the CMS market displays constant growth [25].When compared to the SKB concepts, however, GX and WM do not display a high level of capabilities in the areas of release, delivery, and deployment. To summarize:

- The deployment of external components and WM is mostly a manual process
- Deployment knowledge is too dependent on one source of information
- Update process is too complex
- Not enough separation of WM product and customisations
- Central storage of locations and sources of feedback from WM
- Parts of product documentation can be (re-)used in WM

These weaknesses can all improved by applying the concepts of the CCU model. The main points for improvement are within the area of updating and deployment.

6.1 Deployment and Updating

To begin with, deployments are for the large part a manual process, especially when including the external components such as Tomcat, the JVM, and database applications. There are a number of causes why automating the process is complex. First, GX wishes to offer the maximum amount of customisation options to customers, to prevent any customers being left out or unsatisfied with a system deployment. Secondly, the architecture of WM is highly complex, due to the many different components and often even multiple servers involved. Thirdly, GX has not performed a large number of deployments at customer sites (currently they have a little over 150 customers) that automatic deployment is a requirement for success. Recently GX has, however, standardized the deployment process of WM, using "configuration by exception".

Another problem GX is experiencing is that deployment knowledge is stored in just one unshared location. At present, just one person is responsible for correct deployment at customers, though recent training has started for two others. Also, when looking at the SKB concepts, on-line delivery of both software releases and patches are deployed using the Internet. GX, however, wishes to maintain their CD releases, due to the facts that many of their customers use WM for closed WANs and that phone line modem speeds are generally too low to download the full WM package quickly. Patches, on the other hand, are distributed via e-mail.

Post deployment checks are performed by the deployment engineer and these checks have been documented in an Excel sheet. According to the SKB concepts, however, such checks should either be avoided or performed automatically. GX also creates a number of "how-to" documents at each product release. At the same time, documentation ("specs") is typed by each developer during the development and quality assurance phases. This documentation might be used in the product to support users while using WM. This would require a change in thinking, however, since developers are currently producing the specs for GX personnel only instead of product users.

GX has indicated that their main problems with respect to release, delivery, and deployment can be found within the area of complete updates of WM. When customers request a new site, often a full update of WM is required. Even though GX uses J2EE technology, these complete updates are highly complex due to several reasons. The components that need to be updated are often riddled with customisations, such that overwriting of these customisations will remove them from the product completely. Other complexities play a part as well, such as complex software architecture and complex relationships between external components. Often GX will perform a complete fresh deployment, to avoid the complete update problem. These problems can be removed by formalizing and automating the full update process. To do so a number of problems need to be solved first. The customisations need to be stored as separate components that use the WM's APIs, an effort that GX is consciously making at present. Also, the dependencies on external components need to be formalized and managed explicitly by either a manual or automatic process. Finally, the complexities introduced by using complex ANT scripts for deployment must be simplified or a different technology for WM updates and deployments must be chosen.

6.2 Customer Communication and Licensing

In the area of licensing GX could also improve in a number of areas. Even though GX's firm belief is that licenses do not need to be renewed periodically and can be stored unencoded, the SKB concepts suggest that communicating about licenses encourages communication and awareness of software products within a customer organisation. One of the problems GX has experienced in the past is that customers are not aware of newer WM versions, and thus first approach other vendors. One way to improve customer communication, is by introducing a daily hint in WM, that is updated through the web regularly, for instance by means of an RSS feed. GX would have to tackle the problem of closed WAN customers, though.

GX has applied the concept of (semi-)automatic feedback from WM in different manners, according to the SKB concepts. The knowledge about this feedback, however, has not been made available centrally to support engineers. The SKB concepts vouch for a central storage place for such knowledge, making it available to both the support and development departments.

A lot of knowledge on WM is stored on the GX internal network. GX has a product roadmap available throughout the company, and many different types of instruction manuals are available to GX employees. Also, with the introduction of the partner web, such information is now also available to partners online. Such manuals for instance include user manuals and documents describing how to deploy external components.

The introduction of the partner program will uncover many of these problems stated since in the future both partners and GX themselves will be delivering and deploying solutions to customers. GX must transfer many of these skills to their partners, thus introducing a need for standardization and a transparent software architecture on top of which customisations can be built. Such standardization, however, is held back because GX supports any type of network infrastructure and environment.

In other case studies we researched the options to release "light" versions of the product under research. In the case of GX, however, it is clear that to serve a large number of customers, a large number of employees are required, and since GX hires academic level personnel mostly, such employees would be far too expensive.

A final point for discussion is whether the uncovered problems and proposed improvements will actually provide GX with a competitive edge. This report shows, however, that because most CMS vendors are experiencing fairly similar problems, GX could attain a competitive edge by dealing with (a subset of) these problems. [25]

7 Conclusions

GX is a successful software vendor in the area of content management systems. GX is growing quickly, and with a larger number of customers, GX needs to invest more effort in the release, delivery, and deployment processes of their product to reduce cost and improve customer perceived software quality. Fortunately, GX implements and automates many concepts of CCU. GX stores and manages all releases, GX stores information about the delivery and deployment processes centrally, and this information is made available to customers, developers, and distributors of the software product. Many of these implementations have already lead to more cost efficient delivery and deployment of software. However, GX can effectively decrease costs and improve product quality by implementing some other CCU concepts. GX can still automate many parts of the deployment process, decrease the number of configuration parameters, externalize customizations from the product installation location, improve licensing, and most of all improve the update process of WM from release to release.

This report describes the observations done by Deliver during a case study of GX. The final conclusions are that GX did not yet implement all the concepts of an Software Knowledge Base and can reduce costs and improve product and service quality by implementing more of these concepts.

References

[1] L. Xu and S. Brinkkemper, “Concepts of product software: Paving the road for urgently needed research,” in *First International Workshop on Philosophical Foundations of Information Systems Engineering*. LNCS, Springer-Verlag, 2005.

[2] S. Jansen and S. Brinkkemper, “Definition and validation of the key process areas of release, delivery and deployment of product software vendors: turning the ugly duckling into a swan,” in *proceedings of the International Conference on Software Maintenance (ICSM2006, Research track)*, September 2006.

[3] ——, “Modelling deployment using feature descriptions and state models for component-based software product families,” in *3rd International Working Conference on Component Deployment (CD 2005)*, ser. LNCS. Springer–Verlag, 2005.

[4] S. Jansen, S. Brinkkemper, and G. Ballintijn, “A process framework and typology for software product updaters,” in *Ninth European Conference on Software Maintenance and Reengineering*. IEEE, 2005, pp. 265–274.

[5] A. Mockus, P. Zhang, and P. L. Li, “Predictors of customer perceived software quality,” in *ICSE ’05: Proceedings of the 27th international conference on Software engineering*. New York, NY, USA: ACM Press, 2005, pp. 225–233.

[6] S. Jansen, G. Ballintijn, S. Brinkkemper, and A. van Nieuwland, “”integrated development and maintenance of software products to support efficient updating of customer configurations: A case study in mass market erp software”,” in *”21st International Conference on Software Maintenance (ICSM ’05)”*, 2005.

[7] F. Niessink and H. van Vliet, “Software maintenance from a service perspective,” in *Journal of Software Maintenance: Research and Practice*, vol. 12, no. 2, 2000, pp. 103–120.

[8] A. April, J. H. Hayes, A. Abran, and R. R. Dumke, “Software maintenance maturity model (smmm): the software maintenance process model.” in *Journal of Software Maintenance*, vol. 17, no. 3, 2005, pp. 197–223.

[9] S. Jansen and S. Brinkkemper, “Definition and validation of the key process areas of release, delivery and deployment of product software vendors: turning the ugly duckling into a swan,” in *Technical Report, UU-CS-2005-041*. Utrecht University, 2005.

[10] B. Meyer, “The software knowledge base,” in *Proceedings of the 8th international conference on Software engineering*. IEEE Computer Society Press, 1985, pp. 158–165.

[11] A. van der Hoek, R. S. Hall, D. Heimbigner, and A. L. Wolf, “Software release management,” in *Proceedings of the Sixth European Software Engineering Conference (ESEC/FSE 97)*, M. Jazayeri and H. Schauer, Eds. Springer–Verlag, 1997, pp. 159–175.

[12] R. S. Hall, D. Heimbigner, and A. L. Wolf, “A cooperative approach to support software deployment using the software dock,” in *International Conference on Software Engineering*, 1999, pp. 174–183.

[13] S. Brinkkemper and P. Klint, "Intelligent software knowledge management and delivery," in *Project Proposal*, 2003.

[14] A. Carzaniga, A. Fuggetta, R. Hall, A. van der Hoek, D. Heimbigner, and A. Wolf, "A characterization framework for software deployment technologies," 1998.

[15] T. van der Storm, "Continuous release and upgrade of component-based software," in *Proceedings of the 12th International Workshop on Software Configuration Management (SCM-12)*, 2005, to appear.

[16] ——, "Variability and component composition," in *Software Reuse: Methods, Techniques and Tools: 8th International Conference (ICSR-8)*, ser. Lecture Notes in Computer Science. Springer, June 2004, pp. 86–100.

[17] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," SEI, CMU, Pittsburgh, PA, Tech. Rep. CMU/SEI-90-TR-21, Nov. 1990.

[18] S. Jansen, G. Ballintijn, and S. Brinkkemper, "Software release and deployment at exact: a case study report." CWI technical report, SEN-E0414.

[19] S. Jansen, "Software Release and Deployment at Planon: a case study report." CWI Technical Report, SEN-E0504, 2005.

[20] G. Ballintijn, "Software Release and Deployment at Chipsoft: a case study report." CWI Technical Report, SEN-E0506, 2005.

[21] R. K. Yin, "Case study research - design and methods." SAGE Publications, 3rd ed., 2003.

[22] J. Tiihonen, T. Soininen, T. Mannisto, and R. Sulonen, "State of the practice in product configuration – a survey of 10 cases in the finnish industry," in *Knowledge Intensive CAD, First Edition*. Chapman et Hall.

[23] M. Jaring, R. L. Krikhaar, and J. Bosch, "Representing variability in a family of mri scanners," *Softw. Pract. Exper.*, vol. 34, no. 1, pp. 69–100, 2004.

[24] I. van de Weerd, S. Brinkkemper, J. Souer, and J. Versendaal, "A situational implementation method for web-based content management system-applications: Method engineering and validation in practice," in *special issue of Software Process: Improvement and Practice*. IEEE, 2006.

[25] J. Lundy, K. Chin, and K. M. Shegda, "Who Will Own the Enterprise Content Management Market?" in *Technical Report Gartner*, 2005.

[26] M. van Berkum, S. Brinkkemper, and A. Meyer, "A combined runtime environment and web-based development environment for web application engineering." in *CAiSE*, 2004, pp. 307–321.

Acknowledgements

We would like to thank GX for enabling us to do the case study and present our results. We would like to especially thank Jilles van Gurp, Martijn van Berkum, Leon Brouwers, Mark Koenen, Inge van de Weerd and all the other interviewees. Finally, we would like to thank Tijs van der Storm and Vedran Bilanovic for their helpful reviews and discussions.