Integrated Development and Maintenance of Software Products to Support Efficient Updating of Customer Configurations: A Case Study in Mass Market ERP Software

Slinger Jansen and Sjaak Brinkkemper
Utrecht University
Institute of Information and Computing Sciences
{slinger.jansen, s.brinkkemper}@cs.uu.nl

Gerco Ballintijn
Center for Mathematics and Computer Science
g.ballintijn@cwi.nl

Arco van Nieuwland
Exact Software
arvanie@gmail.com

Abstract

The maintenance of enterprise application software at a customer site is a potentially complex task for software vendors. This complexity can unfortunately result in a significant amount of work and risk. This paper presents a case study of a product software vendor that tries to reduce this complexity by integrating product data management (PDM), software configuration management (SCM), and customer relationship management (CRM) into one system. The case study shows that by combining these management areas in a single software knowledge base, software maintenance processes can be automated and improved, thereby enabling a software vendor of enterprise software to serve a large number of customers with many different product configurations.

Keywords: *product software, software configuration management, product data management, customer relationship management, software delivery, deployment*

1 Integrated Development and Maintenance

The maintenance, release, and deployment of enterprise application software is a complex task for a software vendor. This complexity is caused by the enormous scale of the undertaking. There are many customers for the vendor to serve, which all might require their own version or variant of the application. Furthermore, the application itself will consist of many (software) components that depend on each other to function correctly. On top of that, these components will evolve over time to answer the changing needs of the customers. As a consequence, the release and deployment of these applications take a significant amount of effort and is a time consuming and error-prone process.

To alleviate this problem we envision an intelligent software knowledge base (ISKB) that contains all facts about all artefacts together with their relevant attributes, relations and constraints. 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. The ISKB can improve the software maintenance processes at both the customer and the software vendor sites.

Exact Software (ES), a software manufacturer in the Netherlands serving 160,000 customers worldwide, has implemented an ISKB to manage and improve its software maintenance, release, and deployment processes. The ISKB implemented by ES has been implemented in its own product e-Synergy. In this paper we show that ES successfully supports its large customer base with an integrated product data management (PDM), software configuration management (SCM), and customer relationship management (CRM) system, thereby alleviating the process of software product maintenance. The paper describes how the processes of development, release, and deployment have been improved by integrating processes that were previously managed by utilizing different isolated systems. The paper also demonstrates how a central software knowledge base, containing all the relevant knowledge about software products, is implemented and used to support the processes of software maintenance. Finally, the paper describes four principles employed by ES to deal with general complexities in the software engineering discipline with respect to software maintenance.

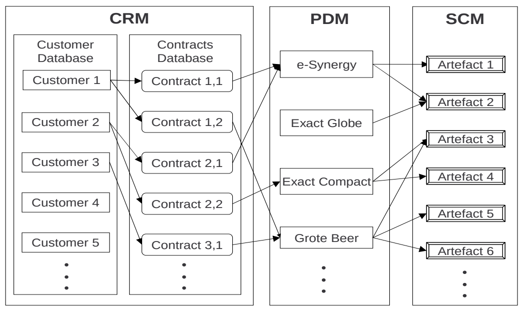

Figure 1. Integration of CRM, PDM and SCM

Figure 1 displays the overall architecture of the integrated CRM, PDM, and SCM systems in e-Synergy. The integration of these systems enables efficient maintenance of software configurations at a customer site. The customer system contains a contract module, in which all products that have been sold to a customer are stored. Each contract applies to a product that can be downloaded and activated by a customer. The products stored in the PDM are associated with their corresponding artifacts in the SCM and this enables a customer to download the correct files that are required for a product update or deployment. The PDM can generate a list of files that is compared to the list of files on the customer system and when differences are encountered the required files are downloaded.

The software maintenance processes on the vendor side also have improved by the integration of different systems. The integration of the SCM and PDM systems allow product managers to quickly oversee whether work still needs to be done on deliverables before the next release. The integration of the SCM and workflow management system enable traceability of changes on deliverables, thereby improving product quality. The integration of the SCM and PDM system also allow for quick deployment and testing of test versions for the quality assurance department. These and other improvements are discussed further in Section 3.

The rest of this paper is structured as follows. Section 2 describes the objective of our research at ES and a motivation. Section 3 describes ES and the tools it uses to integrate its SCM, PDM, and CRM systems. Section 4 describes the maintenance processes of the products at the vendor site and of the configurations at the customer site. Section 5 discusses the lessons learned from ES and what functionality we feel is lacking in Exact's ISKB. Related work is presented in Section 6 and finally Section 7 concludes our paper with a discussion.

2 Research Approach

2.1 Problem Overview

Two important parts of the maintenance process are release and deployment of software. The release and deployment processes for a software product involve a large amount of risk and effort for a software vendor. The required capabilities have been documented insufficiently in literature such as the Software Engineering Body of Knowledge[1] where the issue of delivery is not explicitly handled but seen as part of configuration management. Our research is focussed at further defining the problems in the area of software release and deployment and finding solutions to these problems.

The goal of our research is to simplify the software release and deployment effort. We propose to do so by managing all the knowledge about a software product explicitly. By managing software knowledge explicitly, a software vendor can improve the upgrade process of its software. Furthermore, the explicit management of software knowledge enables the evaluation of "what if" scenarios, such as, what will happen to the current configuration of customer X, if she upgrades application component Y? These evaluations help in assessing the risk of the deployment process, and these assessments, in turn, improve interaction between customer and software vendor because the vendor can guarantee whether a combination of components can function correctly together.

Managing software knowledge is, however, only part of the story. The software still has to be delivered to customers. We aim to support dynamic delivery of software via the Internet, both in the form of upgrades and of full packages. The previously mentioned product and component knowledge is used to compute the difference between the existing software configuration at a customer and the desired configuration. This difference can be used to create the required upgrades.

Central to the maintenance activities we envision, is the intelligent software knowledge base. This software knowledge base can be seen as an integrated SCM/PDM/CRM system that stores all information about all the artefacts that are part of the applications life cycle. The ISKB stores the information of all available applications in all available versions at the vendor site, whereas at the customer site the ISKB stores information about the installed applications, application settings, and configurations.

As part of our research, we are performing case studies at product software companies to evaluate the state-of-the-practice of software vendors in the Netherlands, such as ES. ES is relevant to our research because ES has implemented one of its own products, e-Synergy, to support the

[1]www.swebok.org

processes of release and deployment and to function as an ISKB, which partly validates the theory that an ISKB can improve software release and delivery.

2.2 Exact Software

ES is a manufacturer of software for accounting and enterprise resource planning (ERP), based in Delft, the Netherlands. Since its founding in 1984, ES has established an international customer base of over 160,000 customers, mainly in the small to medium enterprise sector. Through autonomous growth and a number of acquisitions the number of employees has grown to 2025 in 2004 (see Table 1). Of these employees, 18 percent are active in the development of software. They are spread out over the 31 locations worldwide. The International Development department employs most developers with 180 employees in Kuala Lumpur and 20 employees in Delft.

A typical application sold by ES is Exact Globe, a back-office application that integrates business processes, such as finances, logistics, product data management, CRM, etc. A recent product is e-Synergy, a front office application that provides organizations with real-time financial information, multi-site reporting, and relationship and knowledge management capabilities. Employees, customers and partners are provided with real-time access to information across an entire organization.

Based on more than 20 years of experience in developing software products for the SME market, ES enforces 4 main principles for product development:

- **Uniform architecture -** All software developed by ES has a three layered architecture. The user application layer (a browser or a stand alone client), the application server layer (containing the business logic), and the database layer.
- **One-X -** ES has developed a strategy for developing its ERP software, called One-X. One-X aims to develop all software around one single instance of truth, making the data available to all ES applications, such that the data can be created and provided to all the stakeholders. The idea behind One-X is that any data needs to be entered only once and that extensive navigation is possible through integration.
- **KISS -** To support such a large customer base within such a complex problem domain, ES follows the principle of KISS (Keep It Small and Simple) for its development process. The use of KISS within ES has resulted in a development cycle in which a fully functional prototype is produced by a spearhead team first. Once the prototype is released the product enters a maintenance cycle, instead of a development cycle. From then on the product can only be changed by the maintenance team through well-defined maintenance procedures. All procedures are monitored by a large quality assurance team (see Table 1). These procedures allow ES to keep the maintenance of its products simple and controlled.
- **Eat your own dogfood -** ES uses its own software products to support internal processes, which is called "eat your own dogfood" by Microsoft [3]. This internal use provides the maintenance departments with early bug reports and feedback.

Table 1. ES full-time employment (2004)

Department	FTE	Percentage
Support	546	27,0
Services	263	13,0
Sales and Marketing	445	22,0
Finance and Administration	142	7,0
Staff	142	7,0
General Management	81	4,0
Development	294	14,5
Quality Assurance	96	4,7
Release and Deployment	15	0,7
Total	2025	100

2.3 The Case Study

There were three reasons for performing the case study at ES [9]. To begin with, we wished to prove that the ISKB is a relevant solution to manage the processes of release and deployment. Secondly, ES provided us an example ISKB and showed how an ISKB can be applied. They also allowed us to review the reasons for implementing an ISKB to support its processes. Finally, ES gave us an opportunity to see the advantages and disadvantages of using an ISKB in daily life.

During the three month case study, facts have been collected from several sources:

- **Interviews -** To study ES and confirm our hypotheses, interviews were held with the people responsible for the development and usage of the e-Synergy product.
- **Studying the software -** ES granted an academic license for the e-Synergy software. This license helped to gather many facts by examining, using, and experimenting with the software.
- **Document study -** Many of the documents found in the document management system of e-Synergy supported the research and gave an in-depth view of the ES maintenance processes.
- **Direct observations -** Since our research took place at ES's International Development department, we were

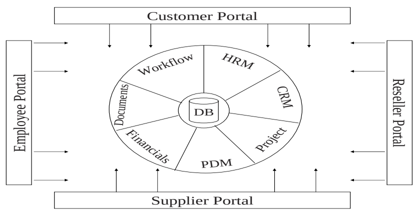

Figure 2. Abstract Architecture of e-Synergy

able to directly observe and document day to day operations.

A threat to the validity of the case study is the correctness of the answers provided by the interviewees. The interviews consisted of two sessions, one to explore and elaborate, and one to cross-reference answers from other interviews. The second session was also used to cross-reference documentation found in the document management system of e-Synergy and confirm the facts stated in these documents. To ensure reliability, the case study report was reviewed by key informants. Besides these reviews we also created a case study protocol and a case study database. A second threat to the validity is the representativeness of ES for the software industry. Even though ES is a market leader, there is a risk that ES may not be representative for other software producing companies within the same problem domain and that some of the conclusions we have drawn cannot be generalized. A last threat to the validity is that developers tend to be reluctant to admit the existence of quality issues. Consequently, they may downplay relevant issues. However, during the case study we interviewed many experienced, senior developers and are confident that their answers were accurate.

3 The ISKB and its use within ES

ES uses its own product e-Synergy, to support all its business processes. ES uses all modules for its activities, such as document management, workflow management, and financial accounting. The document management module, for instance, is used by developers to share documents, work on concepts together, and to review designs. The workflow management module is used to share and manage activities among employees. An implementation of e-Synergy provides four optional Internet portals (see Figure 2), which can be used to provide customers, employees, resellers, and suppliers with their specific views on the data. The database management system, Microsoft SQL Server, is a prerequisite for installing all ES products. With respect to maintenance the e-Synergy product is used to support two forms of maintenance. On the one hand e-Synergy is used to support the maintenance department in performing the product composition, development, bug fixing, and workload division amongst developers. On the other hand e-Synergy is used to supply customers with an interface to the latest releases of the products a customer has purchased. The interface enables customers to update their products to their specific requirements and customisations, and maintain the correct versions of artefacts on the customers site.

The ISKB used by ES, e-Synergy, is a front office application that integrates seven modules: project management, workflow, human resource management, document management, CRM, logistics, and financial activities (see Figure 2). Through the One-X architecture, each module can use the data in other modules enabling users to easily navigate from one item to another. The logistics module of e-Synergy is a PDM module that manages conventional products, the customer relationship module stores information about customers, and the project and workflow modules are used to distribute activities among personnel. Workflow activities are classified as bug reports and change requests and can be attached to other workflow, documents, and deliverables. These attachments can be used to quickly produce reports on how many tasks are still attached to a deliverable or a document. Since all tasks have different defined levels of impact, projections can be made about the amount of work and the cost associated with that work, which enables status reporting.

Before e-Synergy was implemented, ES was utilizing different isolated systems for the processes of software maintenance, release, and delivery, such as daily build servers and conventional concurrent versioning management tools for SCM and its own product Globe, for CRM. ES experienced many problems within the setting of isolated systems. To begin with, many of the tasks performed included the duplicate entry of data into different systems. A second problem experienced by ES was that deliverables were not managed explicitly, delaying deadlines and often producing incomplete sets of deliverables for customers. The final problem relevant to this case study was that multiple worldwide departments needed access to the software repositories twenty four hours a day. To solve these problems ES applied their own product, e-Synergy, internally.

3.1 SCM

The SCM system consists of five repositories, in which five concurrent releases of all deliverables and corresponding source code are stored, as shown in Figure 3. Each of the five repositories contains a release of all the source files, help files, binary files, executables, resources, and SQL scripts, for one product, such as e-Synergy or Globe. Periodically, depending on the quality criteria for each reposi-

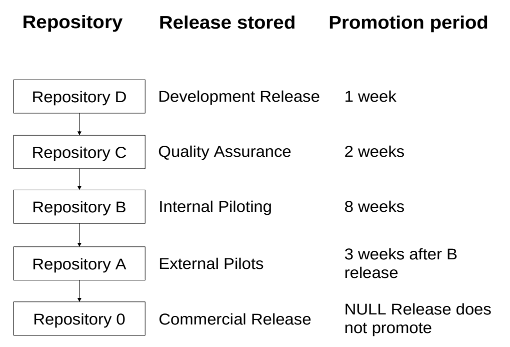

Figure 3. Repository Promotion Scheme and Promotion Periods

tory, the full repository is manually promoted (copied) from one repository to another.

All developers perform their operations, such as committing, on the development release stored in the D repository. When all uploaded bug fixes and new functionalities have been finished and checked by the programmers on the release stored in the D repository, that release is promoted to the C repository, overwriting the release previously stored in that repository. Once every two weeks, the release in the C repository, after being approved by quality assurance personnel, is copied to the B repository. The release stored in the B repository is, if possible, used internally by all ES personnel and is thereby thoroughly tested. This testing generates new bug reports and functionality requests again.

Approximately every two months, when the release stored in the B repository is deemed stable enough by primary internal business users, such as the director of ES Finance and Administration, the release is copied to the A repository, which is open to external pilot customers who report their experiences back to a designated developer. After the release in the A repository has been used for a minimum of three weeks, the release is copied to the NULL repository containing the official product release, which is sent out to customers on CD-ROMs or through the Internet. To remove the complexities introduced by the concurrent versioning systems, ES now uses one single development version to manage the artefacts maintained by the maintenance department. Versioning of files is therefore not possible, which is different from common practice, but one of the results of the KISS strategy of ES.

3.2 PDM

PDM is the discipline of "governing the control of the product data and processes used during the entire life cycle of a product" [7]. When ES started designing e-Synergy, ES concluded that producing software is no different from producing a conventional product and that a PDM system can be used to manage software products.

The PDM system of ES is used to control its (software) product deliverables. PDM systems generally implement a classification of artefacts to support reuse [7]. The PDM implemented in e-Synergy makes use of atomic entities called "items" by ES. An item can be used to represent any business item, such as a promotional sweatshirt, a printout of a manual, but also an executable belonging to a software product. Items are categorized, of which the relevant categories for this case study are sales items, source items, and deliverable items.

- **Sales items -** ES uses sales items to encapsulate all sellable goods. A sales item can be a service agreement, a manual, a piece of software (including a paper manual and a CD-ROM), or any other good sold by ES. From each sales item a bill of materials can be generated, stating what items are necessary to complete the product. When a sales item is a software product the bill of materials includes all deliverable items, but no source items.
- **Deliverable items -** Deliverable items are deliverables that depend on source items, even if they are simply direct copies of those source items. Deliverable items include digital manuals, resource files, library files, and executable files.
- **Source items -** Source items are source files that are required to create a deliverable. Source items are source code files, resource files, etc. Companies producing conventional products use the source items to store their basic raw materials and resources with which they create their products.

ES products are represented in e-Synergy in different ways at the development sites of ES, whereas instantiation information for products at customers is stored in the contract management facility, the product data management facility, and locally at the customer's site. These two different views are based on the assumption that sales personnel do not need to know all the implementational details, whereas developers are not concerned with sales knowledge. An example is that the sales department sees products as decomposable modules that can be sold separately, whereas development sees the product as a large set of deliverables containing all modules, which are later activated or deactivated at runtime by the license file.

Figure 4 displays a generic product structure and the two different views of that structure:

Sales View - From a sales point of view, it is not relevant what deliverables and sources look like. For the sales view it is required to know what a product costs, what op-

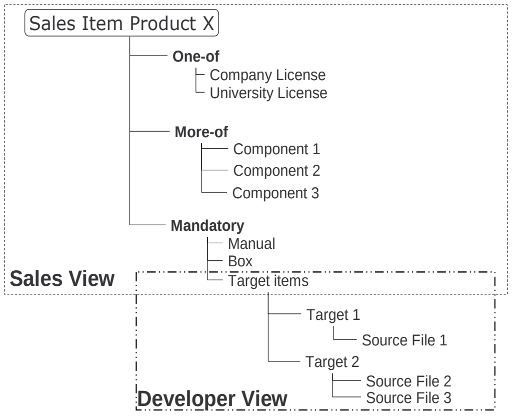

Figure 4. Sales and Developer Product View of an Example Product

tions there are to a product, what kind of sales agreements are possible, and what materials make up a product. Sales personnel thus share no interest in source files. A product, consisting of sales items connected by the one-of, more-of, mandatory, and optional relationships, can be instantiated by binding these relationships. The binding of these relationships corresponds with the product information stored in a license file. The relationships defined here are similar to the relationships defined for feature diagrams [11] [19].

Development View - Developers are concerned with deliverable files and source files. As developers always work in the context of a product, and they know the complete structure of that product, however, developers usually do not use sales items. A developer considers a product as a complete set of deliverables, therefore there are only two relationships available for the development view, the Sources relationship, meaning the file is a source file for some parent deliverable item, and the Deliverables relationship, meaning the file has corresponding sources.

Currently, the PDM is implemented to only store lists of all the mandatory deliverables for a product. The fact that mandatory deliverables are stored in a list, and their sources are children of these deliverables, means that the dependencies amongst deliverables are not explicitly stored. The ES Product Updater [10] uses the list of deliverables for products to compare that to the list of installed files at the customer. The solution implemented by ES, where one product is configured to serve different purposes, is cheaper than implementing a full product line at this scale [8].

3.3 CRM

With respect to customers ES has attempted to increase customer contact, scale down support, reduce the complexity of the delivery process of software products, and reduce piracy of its products, all with e-Synergy. ES believes that product (experience) improvement by intensive customer contact will retain more customers [12]. Customers log into the ES portal to see their contract information, to see the status of their support calls or bug reports, and to download (renewed) license files. Because customers are expected to visit the ES customer portal regularly, customers are notified of new NULL releases and other products through the ES e-Synergy customer portal.

Customers can use the information portal of e-Synergy to access the CRM system and see the status of their contracts, see their support questions, find new products, and find where customers can download license files to activate their purchased products. The CRM module contains a contract management facility which was built to meet Exact's requirements. The contract management facility stores a link to the customer information, purchased product information, the license file for each product, the version number of the latest sent out version, and a link to customer support calls and service status. The purchased product information lists what variants of products have been purchased. The corresponding license files are only available for download by customers. License files are generated every twenty four hours, depending on whether a new contract has become available or needs to be renewed. The license file is published on the customer portal of e-Synergy so that customers can download the periodically renewed license file.

4 Maintenance and the ISKB

4.1 Maintenance at the Development Site

Before the systems of SCM, PDM, and CRM were integrated, the concurrent versioning system RCS [17] was used to support development. Besides RCS, ES used daily build servers, so that the quality assurance department could check the work of the day before. During the implementation of e-Synergy, ES drew up standard requirements for change control, team support, status reporting, process control, and audit control. Aside from these standard requirements, ES had the following non-generic requirements [14] in the area of SCM:

- **Version control -** In the past too many resources were absorbed by legacy support and customers were confronted with complex upgrades and bug fixes. As a consequence of the KISS principle, ES decided to reduce complexity for the customer and development by no longer supporting and storing multiple versions of a product.

- **Configuration support -** Because bandwidth and disk space are cheap and development is expensive, ES concluded that sending each customer the full set of deliverables for a product would be most efficient. Wanting to keep its software as simple as possible, ES decided to support only runtime variabilities. Also, to improve service and product quality, ES wished an automated check of the validity of a product configuration before it is delivered.
- **Build support -** ES previously used separate build servers to build the complete product overnight. That build was tested the next day by quality assurance. As ES grew larger internationally and developers were working on code 24 hours a day, there was no down time left for servers to build the software. A new way of partially building was required to facilitate these needs.

ES promotes some key starting points for its development process. To begin with, developers have private ownership of deliverables and source code and they commit their compiled deliverables with their source code. This introduces pessimistic locking, and enables management to assign responsibility for deliverables to one developer specifically. ES uses very strict maintenance procedures and rolebased workflow for each task such as functional requests or bugs. First the product manager evaluates the task. Once evaluated the developer executes the task. Finally, quality assurance tests whether the task was successful and changes the status of the task.

The decision for building the end product on the developer's workspace is deliberate since in the past ES experienced too many resources were spent on the build process. The solution where developers upload their compiled deliverables creates a repository that always contains the most recent build of the software, which removes the need for nightly builds. This solution also enables ES to have a latest running version available 24 hours a day at all departments worldwide. In the case of ES, which has development sites in Kuala Lumpur, the United States, and the Netherlands, such a system allows developers to work on the products 24 hours per day without any interruption from build servers.

- **Manage deliverables -** Previously, source files were the focus of management instead of deliverables. A compiled version of the software created on the build server and a manual from the manual department were the only deliverables. As product complexity grew, however, ES desired to be able to manage the deliverables individually and attach workflow and documents to deliverables to increase its traceability. Also, previously developers determined, depending on the requirements, what the variabilities of a product would look like. ES decided that sales departments should be able to influence at what level a developer introduces variability.
- **Ease of delivery -** ES wished to automatically update its products with an evolving set of deliverables because ES consultants were often confronted with complex manual update procedures in the past.

To manage the deliverables and ease the maintenance process at the customer site, e-Synergy's PDM module in was employed. The PDM functionalities were implemented as a central system to the process of maintenance. e-Synergy connects the PDM and the human resources management system to enable ES to assign deliverables to specific developers and hold developers responsible for the quality of their deliverables. Before e-Synergy was introduced, when deliverables were mostly unmanaged, automatically reusing modules for different products was impossible. Since deliverables can now be linked together to form new products, ES can create new products from a standard set of components.

Because currently all deliverables are stored explicitly in the PDM system, the Product Updater can automatically retrieve a list of deliverables for a product from the PDM and install them if necessary. The fact that all deliverables can be retrieved this way eases the process of software delivery to customers.

Combined SCM and PDM support is provided by the logistics module of e-Synergy since it stores the product data, the deliverables, and the source code. ES has combined the PDM and SCM systems in e-Synergy because ES believes that building a software product is no different from building physical products and can be done using a general PDM system. Developers see the SCM system primarily as one repository in which both the source and their corresponding deliverables, such as executables, are stored. For sales personnel the PDM primarily consists of physical objects and software objects, to which documents, software deliverables, and workflow activities can be attached. Finally, the PDM system supplies customers with a link to the published repository from which they can download the most recent versions of the artefacts that are part of the products the customers have purchased.

4.2 Maintenance at the Customer

Figure 1 depicts a small subset of the information stored in the integrated system. The CRM stores information on customers and their contracts. The contracts are used to generate licenses and store what product(s) a customer presently has rights to. The products, as stored in the PDM, are linked to the artefacts that make up that product and that must be deployed at the customer site when the customer owns the rights to that product. Whenever a customer logs in to the system to update a product, the license is first checked. Once the license is approved, the product the customer wishes to update is looked up and a list of current artefacts for that product is generated. The list is sent to the customer for comparison to the local artefacts. The update tool then decides what files need to be updated. The ISKB implemented by ES consists of the SCM, PDM, and CRM modules in e-Synergy and are integrated through the One-X architecture.

ES has chosen to deliver the full set of deliverables for a product to a customer, abolishing the need for elaborate dependency information among software modules. The reasons for this approach are that development costs for a partial delivery system are high while disk space and bandwidth are cheap. Besides e-Synergy, there is one external tool that performs actions on the repositories. The Product Updater downloads and installs all deliverables for a release from the repository the user chooses (and has permission to). The Product Updater establishes what deliverables are present at the client site and downloads (new versions of) the required files. The Product Updater also has some scripting capabilities to install the application and create and transform the tables in the database. The Product Updater is not part of e-Synergy because it deploys deliverables at the client, which e-Synergy, a web application, cannot do.

Before e-Synergy had been implemented, ES only used its own product Globe for CRM. However, when maintenance matters had become too complex the overall goals became to reduce complexity of delivery, intensify customer contact, and reduce the cost of the support department leading to the following non-generic requirements:

- **Facilitate custom solutions -** The ES customer base still depends for a noteworthy part on custom solutions to extend current functionality in ES products. They wanted to reduce complexity of delivery, yet still facilitate custom solutions and extensions to its products, and ES wanted to remove the expensive need for consultants at the customer site to perform an update of the product.
- **Unify licensing and CRM -** ES realised its software was being copied and distributed illegally. To trace back illegal licenses ES wished to link a license directly to a customer, whereas in the past its licensing was done through license numbers provided with the distribution CD-ROM.

ES uses the CRM module in e-Synergy, which is based on these requirements combined with the functional specifications of the conventional CRM system implemented in Globe.

4.2.1 Contracts and License Files

Some of the results of the integration of the PDM and CRM systems are the contract management functionalities and the license files. The version number that is stored in the contract management facility for each customer's product is changed automatically, when a customer downloads an update, or manually, when a CD-ROM is sent out to a customer. The version number is used for support purposes, telling the support department what version a customer is currently using. The link to the customer support calls and service status is used to see how many calls a customer has made, and whether a customer is still allowed support. Using the support information leads to a stricter way of dealing with support and results in less support calls.

At the customer site a data file contains a list of all the deployed deliverables and a license file contains a coded version of all the bound relationships for a product, stating all modules that were purchased by a customer. The license file also contains information on the expiration date of the license, since licenses need to be refreshed periodically (yearly for most products). Finally the license contains, if available, a pointer to a download location for a custom solution. The download location can be used by the Product Updater to update the custom solution for a customer. The deployed data file stores the version number and the install location for each deliverable. The data file is used by the Product Updater when updating, by comparing the deployed data file to the list of available deliverables for a product. After an update the data file is updated to contain all the newest information. The local settings for a product are stored in the database of the client.

Updates are destructive due to file overwrites and the fact that destructive database conversions cannot be rolled back. In an attempt to reduce piracy a license checking mechanism was implemented. Each time the customer starts the software, the license file must be decoded and checked. Also, periodically, the license file is renewed and must be downloaded again to keep the product active. Currently there is no data available to support whether piracy has effectively been reduced.

ES is unique because they provide customers a direct link to their individual contracts and their license files. ES is also unique because e-Synergy stores the version number of a product deployed at the customer, improving support. Finally, the usage of license files to encapsulate product instantiation information is common in the software industry.

5 Discussion

The customer base of ES has shown a constant growth over the last 15 years. Related to the area of maintenance and development ES has dealt with this growth by integrating its CRM, its PDM, and its SCM systems. The solution implemented by ES teaches us three lessons.

- **Integrated support systems for maintenance -** The first lesson is that integration of these three systems can be highly profitable. The integration has resulted in a reduction of effort required for the processes of maintenance. Explicit management of deliverables with the PDM system have enabled ES to attach workflow to them, thereby providing a software maintenance process which is easy to manage and enables quicker releases. The integration of the CRM contracts facilities and PDM enable ES to quickly and automatically manage the delivery of software and licenses to customers through the Internet.
- **Integrated maintenance of customer configurations and published releases -** ES teaches us a second lesson, that by mapping maintenance of configurations at customers onto the published deliverable repositories, the processes of release, delivery, and deployment become less complex.
- **Development simplification -** The third lesson lies in the fact that ES attempts to simplify all processes, thereby eliminating complexities that would normally result in more effort. The simplification approach used, has resulted in some striking changes. Their decision to deploy the full set of deliverables has removed the complexities of partial delivery and removed the need for dependency tracking among modules, enabling ES to focus more on the delivery process.

In our current experiences, delivering the full set of deliverables is common practice for software developers practicing KISS. Another influential decision is the decision to remove build servers that would perform daily builds, and introduce the concept of developers committing deliverables with their sources. ES has also chosen to build all its products on one universal data model, the One-X architecture, enabling applications to share data among each other. Finally, ES uses a maintenance cycle instead of a development cycle to improve its software. There are two advantages to the maintenance cycle. First, it reduces complexity for developers and quality assurance because developers do their activities in a maintenance cycle with predefined workflow. Secondly, the workflow module stores the processed workflow, making activities traceable.

There are downsides to the strategies employed by ES as well. ES does not keep track of dependencies among modules, simply because they are not required when deploying the full set of deliverables at a customer. Without these dependencies it is impossible to employ a product line approach, where dependencies among modules are required to guarantee a customer that a subset of modules is complete. Another downside of the simplification strategy is that ES performs destructive updates, disabling customers to move back to older versions of the software. Finally, ES does not allow developers to branch development in its SCM software to simplify the maintenance process, thus restricting concurrent development.

ES's implementation of e-Synergy can be seen as an intelligent software knowledge base (ISKB). The ISKB consists of (A) a vendor side ISKB that stores all product and component knowledge and (B) a local software knowledge base consisting of the data file containing all deployed deliverables, the configuration information of the tools stored in the database, and the license file storing a list of all activated modules and licensing information. It is the ISKB that has allowed ES to expand its customer base, since before the implementation of the ISKB, too many resources were used up by maintenance tasks to allow growth to 160.000 customers.

The case study has influenced our research in the following ways. First, the ES solution is not easily applicable to products in domains that do not wish to send out all deliverables, wish to provide incremental updates that can be rolled back, or wish to maintain a high level of reusability among products. In contrast, we wish to create a generalizable solution. Secondly, the case study shows us that integration of software knowledge, as we suspected, is a powerful tool in the maintenance of software. ES, however, also showed us this software knowledge can be effectively used in other processes, such as workflow management, human resource management, and customer support. Finally, we observe that the simplification techniques applied by ES demonstrate that solutions coming from Academia conflict with the KISS principle and are generally too complex to be easily adopted by the industry. However, the ES case does show that a simplified instance of the concept of an ISKB improves a company's ability to handle large amounts of customers.

6 Related Work

During our research we encountered some related work in the areas of software knowledge bases, integration of processes, product configurations, and deployment tools, schema, and languages. The need for a distributed intelligent software knowledge base, such as at ES, has already been mentioned in literature. Meyer is the first one to introduce the concept of a centrally available software knowledge base [15]. Others, such as [13] and [16] emphasize the need for explicit knowledge management during development and maintenance. The process and evaluation of software product updating have been recorded in [10], including an evaluation of Exact's Product Updater.

In the Finnish industry the study of integration of PDM and SCM systems in [7] and the case studies [18] of the use of PDM tools and SCM tools are reported. The techniques applied by ES to integrate SCM and PDM are similar to those described by these two studies.

For instantiations of configurable sets of products we found that much of our research is related to the work performed by [1]. Conradi and Westfechtel [2] have given an extensive description of SCM technology and we have positioned the strategies of ES within their presented evaluation.

The need for a system that uses an ISKB to support deployment of software has resulted in a deployment tool called the Software Dock and a case study regarding this tool [6]. The same group has published a relevant study of deployment schema and languages [5]. The deployment languages and schema of ES are not as advanced as those of the Open Software Description (OSD) [20] and the Desktop Management Taskforce (DMTF) [4], since these can, contrary to the ES product descriptions, describe component dependencies.

7 Conclusion

This paper describes a case study of an intelligent software knowledge base at Exact Software. The case study helps to provide evidence that the complex maintenance tasks of enterprise application software for a vendor is best managed with an ISKB. Our contribution is twofold. First, we showed that explicitly managing software knowledge improves the processes of release and deployment for a software vendor selling different enterprise resource planning software products. Secondly, we showed that integrating the knowledge with other systems, such as PDM and CRM systems optimizes the processes of maintenance and delivery, and enables vendors to serve a large customer base. We will use the results of this case study in comparisons with other case studies we are performing at other software manufacturers and we will use the results to build prototype tools related to ISKBs in cooperation with the industry.

References

[1] J. Bosch and M. Högström. Product instantiation in software product lines: A case study. *Lecture Notes in Computer Science*, 2177:147, 2001.

[2] R. Conradi and B. Westfechtel. Version models for software configuration management. *ACM Comput. Surv.*, 30(2):232–282, 1998.

[3] M. A. Cusumano and R. W. Selby. Microsoft secrets. Free Press, 1995.

[4] Desktop Management Task Force. Software standard groups definition, version 2.0. 1995.

[5] R. S. Hall, D. Heimbigner, and A. Wolf. Evaluating software deployment languages and schema. In *Proceedings of the International Conference on Software Maintenance*, pages 177–187, 1998.

[6] R. S. Hall, D. Heimbigner, and A. L. Wolf. A cooperative approach to support software deployment using the software dock. In *Proceedings of the International Conference on Software Engineering*, pages 174–183, 1999.

[7] U. A. Ivica Crnkovic and A. P. Dahlqvist. Implementing and integrating product data management and software configuration management. Artech House Publishers, 2003.

[8] A. Jaaksi. Developing mobile browsers in a product line. *IEEE Softw.*, 19(4):73–80, 2002.

[9] S. Jansen, G. Ballintijn, and S. Brinkkemper. Software Release and Deployment at Exact: a case study report. CWI technical report SEN-E0414, 2004.

[10] 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*, pages 265–274. IEEE, 2005.

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

[12] D. J. Kim, D. L. Ferrin, and H. R. Rao. A study of the effect of consumer trust. In *5th international conference on e-commerce*, pages 310–315. ACM Press, 2003.

[13] P. Klint and C. Verhoef. Enabling the creation of knowledge about software assets. *Data Knowl. Eng.*, 41(2-3):141–158, 2002.

[14] H. Mei, L. Zhang, and F. Yang. A software configuration management model for supporting component-based software development. *SIGSOFT Softw. Eng. Notes*, 26(2):53–58, 2001.

[15] B. Meyer. The software knowledge base. In *Proceedings of the International Conference on Software Engineering*, pages 158–165. IEEE Computer Society Press, 1985.

[16] P. N. Robillard. The role of knowledge in software development. *Commun. ACM*, 42(1):87–92, 1999.

[17] W. F. Tichy. RCS a system for version control. *Software Practice and Experience*, 15(7):637–654, 1985.

[18] 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.

[19] T. van der Storm. Variability and component composition. In *Software Reuse: Methods, Techniques and Tools: 8th International Conference (ICSR-8)*, Lecture Notes in Computer Science, pages 86–100. Springer, June 2004.

[20] A. van Hoff, H. Partovi, and T. Thai. The open software description format. w3.org, 1997.

Acknowledgements

The authors would like to thank Exact Software and its employees for participating in the case study. Specifically the authors would like to thank Sebastian Toet, Jeroen van der Heide, and Nenad Borota for their extensive help. Finally, thanks go to Tijs van der Storm, Vedran Bilanovic, and Paul Klint for their discussions and preliminary reviews of our paper.