JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE
J. Softw. Maint. Evol.: Res. Pract. 2006; **18**:133–151
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/smr.330

Practice

Integrated development and maintenance for the release, delivery, deployment, and customization of product software: a case study in mass-market ERP software

Slinger Jansen[1,*,†], Gerco Ballintijn[2], Sjaak Brinkkemper[1] and Arco van Nieuwland[1]

[1]*Institute of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB, Utrecht, The Netherlands*
[2]*Center for Mathematics and Computer Science, P.O. Box 94079, 1090 GB, Amsterdam, The Netherlands*

SUMMARY

The maintenance of enterprise application software at a customer site is a complex task for software vendors. This complexity results in a significant amount of work and risk. This article 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 resource planning software to serve a large number of customers with many different product configurations. Copyright © 2006 John Wiley & Sons, Ltd.

Received 26 September 2005; Revised 29 November 2005; Accepted 16 December 2005

KEY WORDS: product software; software configuration management; product data management; customer relationship management; software delivery; deployment; customizations; product release

*Correspondence to: Slinger Jansen, Institute of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB, Utrecht, The Netherlands.
†E-mail: slinger.jansen@cs.uu.nl

Contract/grant sponsor: NWO/Jacquard; contract/grant number: 638.001.202

1. INTEGRATED DEVELOPMENT AND MAINTENANCE

The complexity of the maintenance, release, and deployment processes of product software is a result of the enormous scale of the undertaking. There are many customers for the vendor to serve, who 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 evolve over time to answer the changing needs of customers. As a consequence, the release and deployment of these applications take a significant amount of effort and are time-consuming and error-prone processes.

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 alleviate these processes, we envision a software knowledge base (SKB) that contains facts about all product artifacts 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 [3]. The need for a distributed software knowledge base comes from the literature. Meyer was the first to introduce the concept of a centrally available software knowledge base [4]. Others, such as Klint and Verhoef [5] and Robillard [6] emphasize the need for explicit knowledge management during development and maintenance.

Exact Software (ES)[‡], a Dutch software manufacturer that serves 160 000 customers worldwide, has implemented a SKB to manage and improve its software maintenance, release, and deployment processes. The SKB used by ES is implemented in their own commercial product called e-Synergy. In this article we show that ES successfully supports its large customer base with an integrated product data management, software configuration management, and customer relationship management system, thereby alleviating the process of software product maintenance. The article 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 article also demonstrates how a central SKB, containing all of the relevant knowledge about software products, is implemented and used to support the processes of software maintenance. Finally, the article describes four principles employed by ES to deal with general complexities in the software-engineering discipline with respect to software maintenance.

Figure 1 displays the overall architecture of the integrated customer relationship management (CRM), product data management (PDM), and software configuration management (SCM) systems in e-Synergy. The integration of these systems enables efficient maintenance of software configurations at a customer site. The CRM 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 system are associated with their corresponding artifacts in the SCM system, which enables a customer to download the correct files that are required for a product update or deployment. Finally, the PDM system generates a list of files on the vendor side

[‡]Please note that this research took place in 2003 and 2004. Some of the presented results no longer represent daily ES practice.

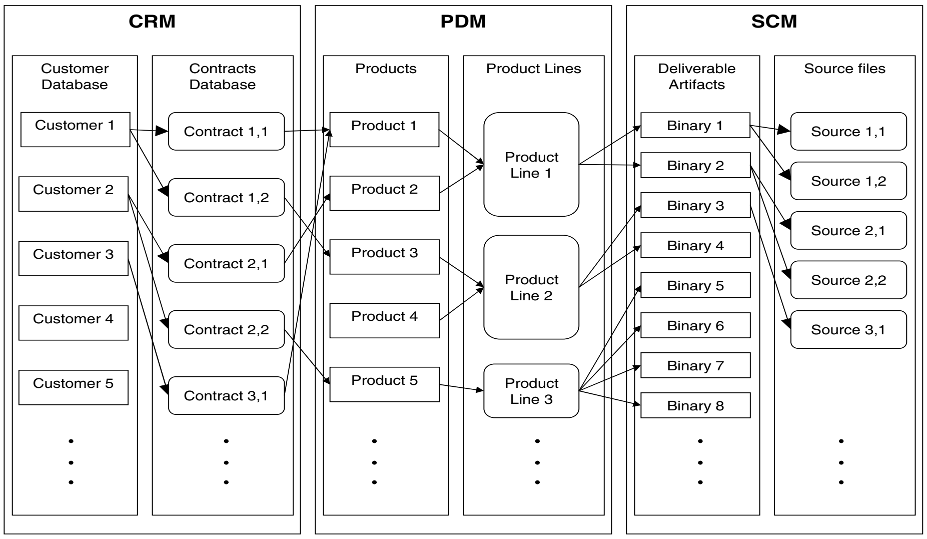

Figure 1. Integration of CRM, PDM, and SCM.

that is compared with the list of files on the customer system. When differences are encountered the required files are downloaded by the customer automatically to update the customer configuration.

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 enables traceability of changes on deliverables, thereby improving product quality. The integration of the SCM and PDM systems also allows 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 article is structured as follows. Section 2 describes the objective of our research at ES and 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 ES's SKB. Related work is presented in Section 6 and finally Section 7 concludes our article 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. These processes, however, have not been documented sufficiently in the literature. The SWEBOK[§], for instance, gives a generic description in the SCM chapter of the processes of release and delivery. The Capability Maturity Model (CMM) [7] also does not provide adequate descriptions for these processes, which is explained by the fact that the CMM does not focus on product software specifically. Attempts have been made in the release candidate of the IT Service CMM [8], although the IT Service CMM also does not provide an elaborate description for the processes of release, delivery, and deployment. Clearly, even though there is a need for process definitions, there are no adequate process descriptions available. The goal of our research is to simplify the software release and deployment effort. We propose to do so by managing all of the knowledge about a software product explicitly. 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 is used to create required upgrades [9]. Central to the maintenance activities we envision is the SKB. This SKB can be seen as an integrated SCM/PDM/CRM system that stores all information about all of the artifacts that are part of the applications lifecycle. The SKB stores the information of all available applications in all available versions at the vendor site, whereas at the customer site the SKB stores information about the installed applications, application settings, and configurations. Both the vendor and the customer can request or receive information from the configuration of the other party, such as regular updates, product information, usage and error reports, product knowledge, and licenses.

As part of our research, we are performing case studies [10,11] 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 processes of release and deployment and to function as a SKB, which partly validates the theory that a SKB 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 I). Twenty per cent of these employees are active in the development of software on several international locations with the largest part (180 employees) working in Kuala Lumpur.

[§]http://www.swebok.org

Table I. 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 and general management	223	11.0
Development	294	14.5
Quality assurance	96	4.7
Release and deployment	15	0.7
Total	2024	100

A typical application sold by ES is Exact Globe, a back-office application that integrates business processes, such as finances, logistics, PDM, and CRM. 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 small to medium enterprise market, ES enforces four 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, which 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 data need to be entered just 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 where a fully functional prototype is produced by a spearhead team first. Once the prototype is released the product enters a maintenance cycle and the product can then only be changed by the maintenance team through well-defined maintenance procedures. All procedures are monitored by a large quality assurance team, as seen in Table I. 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 [12]. This internal use provides the maintenance department with early bug reports and feedback.

These principles are enforced with one fact in mind. Revenue statistics from ES show that the largest stream of income has been coming from maintenance contracts since 2003, whereas previously money came for the larger part from license sales. This made ES management realize that investment in the maintenance process was required.

2.3. The case study

There were three reasons for performing the case study at ES [13]. To begin with, we wished to prove our hypothesis that explicit management of software knowledge on both the customer and vendor site can improve the CCU processes. Secondly, ES provided us an example SKB and showed how a SKB can be applied. They also allowed us to review the reasons for implementing a SKB to support its processes. Finally, ES gave us an opportunity to see the advantages and disadvantages of using a SKB in daily life. During the three-month case study, facts have been collected using 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 able to directly observe and document day-to-day operations.

The validity threats to our descriptive case study are construct, internal, external, and reliability [14] threats. With respect to construct validity, the protocol used for this study was applied to a number of case studies [10,11], which was guarded by closely peer reviewing the case study process and database. To create a complete and correct overview, the development and release, delivery, and deployment processes have been documented extensively. The internal validity was threatened by incorrect facts and incorrect results from the different sources of information. The interviews that were held consisted of two sessions, one to explore and elaborate, and one to cross-check documentation found in the document management system of e-Synergy and to confirm facts stated in other interviews.

With respect to external validity, a threat is that this case is not representative for the Dutch software vendor market and the ERP domain. This threat was dealt with by comparing general information from ES to other vendors that are active in the Platform for Productsoftware[¶], a national organization that aims to share knowledge between research institutes and software vendors, with over 100 members. The comparison shows that ES is, although being one of the largest ERP manufacturers in the country, similar to other ERP manufacturers. Finally, to defend reliability, the same results would be gathered if the case studies were redone, with one major proviso, which is that many of the improvements in the case study report, published after the case study, were implemented by ES.

3. THE SKB AND ITS USE WITHIN ES

ES uses its proprietary product e-Synergy, to support all of its business processes. ES uses all e-Synergy modules for its activities, such as document management, workflow management, and financial accounting. An implementation of e-Synergy provides four optional Internet portals (see Figure 2),

[¶]http://www.productsoftware.nl/

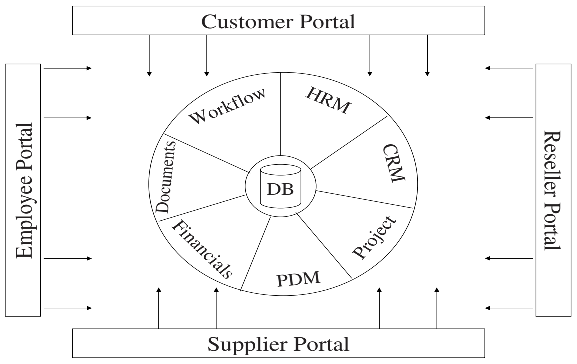

Figure 2. Abstract architecture of e-Synergy.

which are used to provide customers, employees, resellers, and suppliers with their specific views on the data. With respect to maintenance e-Synergy 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 products.

The SKB 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, as seen in 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 CRM module stores information about customers, and the project and workflow modules are used to distribute activities among personnel. Development workflow activities are classified as bug reports and change requests and can be attached to other workflow, documents, and deliverables. These attachments are 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. 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 24 hours a day. To solve these problems, ES implemented their own Web-based e-business product, e-Synergy.

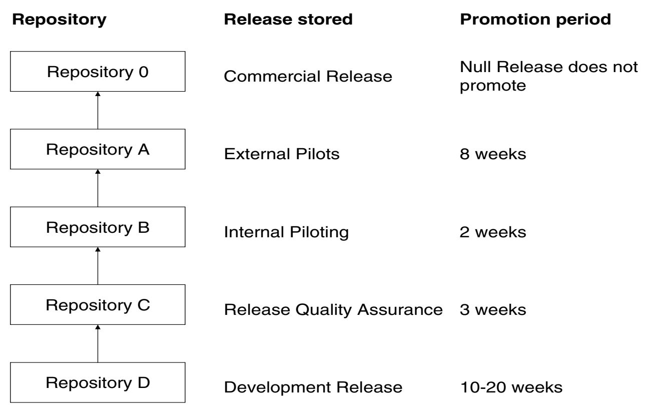

Figure 3. Repository promotion scheme and promotion periods.

3.1. SCM

The SCM system in e-Synergy 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 repository, the full repository is manually promoted (copied) from one repository to another.

All 294 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 checked in by the programmers on the release stored in the D repository, that release is promoted to the C repository on a weekly basis, overwriting the release previously stored there. The quality assurance department checks the release and reports errors back to the development department. Every 10–20 weeks quality assurance freezes the C repository for three weeks to check that release intensively. After approval from quality assurance, that release 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 or functionality requests again.

When the release stored in the B repository is deemed stable enough by primary internal 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 an experienced support employee. After the release in the A repository has been used for a minimum of eight 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 software artifacts. Versioning of files is therefore not possible, which is different from common practice, but one of the results of the KISS strategy of ES. New functionality releases occur approximately four times a year.

3.2. PDM

When ES started designing e-Synergy, ES decided that their commercial PDM system could just as well manage its software products and control its (software) product deliverables. PDM systems generally implement a classification of artifacts to support reuse [15]. The PDM system implemented in e-Synergy makes use of atomic entities called 'items' by ES. An item is used to represent any business item, such as a promotional sweatshirt, a printout of a manual, or a software product's executable. 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 is a service agreement, a manual, a software product (including its paper manual and 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 (such as the deliverables for a software product).
- *Deliverable items*. Deliverable items 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's 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 PDM 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 the point of view of the sales department it is not relevant how deliverables and sources look. For this view it is required to know what a product costs, what options there are for 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, is 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 [16,17].

Development view

Developers are concerned with deliverable files and source files. Developers always work in the context of a product and they know the complete structure of that product; however, developers usually

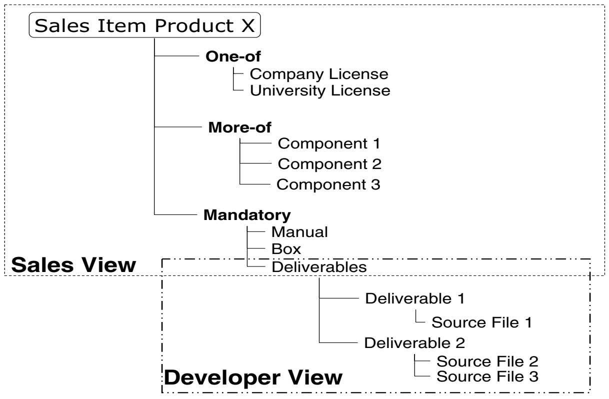

Figure 4. Sales and developer product view of an example product.

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.

The ES PDM system stores lists of all the mandatory deliverables for a product and these deliverables have sources as their children. This results in the fact that dependencies amongst deliverables are not explicitly stored. The ES Product Updater [18], a proprietary tool that is used to update all ES products, uses the list of deliverables for products to compare that with the list of installed files at the customer. Also, even though the PDM relationships such as more-of and one-of are used to deliver packages of only the purchased components to a customer, all deliverables for a product are delivered to a customer. The e-Synergy product is currently represented by approximately 8000 source items, tools, and deliverable items in ES's PDM system. The largest part consists of source files that are not delivered to customers. Other items are tools that are required to produce the product, but are not delivered to the customers, such as developer tools. The deliverable items are executables, dll files, sql definitions, manuals, boxes, CDs, and promotional material. ES's PDM system stores artifacts in product lines. Product lines are collections of products that share items (files) to support reuse.

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. ES believes that product (experience) improvement by intensive customer contact will retain more customers [19]. 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. As 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. e-Synergy's 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 24 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 SKB

4.1. Maintenance at the development site

Before the systems of SCM, PDM, and CRM were integrated, the concurrent versioning system RCS [20] was used to support development. In addition to 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 [21] 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*. As bandwidth and disk space are relatively cheap and development is expensive, ES concluded that installing the full set of deliverables for a product and activating purchased modules according to a license is more efficient than doing partial delivery. Also, to improve service and product quality, ES wanted an automated check of the validity of a product configuration before it is deployed.
- *Build support*. ES previously used separate build servers to build products 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 role based workflow for each task such as functional requests or bugs. First the development manager evaluates the task. Once evaluated the developer executes the task. Finally, quality assurance tests whether the task was successful and approves the task.

ES has development sites in multiple time zones covering 24 hours, which eliminates the option to perform nightly builds. This caused the decision for building the end product on the developer's workspace. The solution where developers upload their compiled deliverables creates a repository that always contains the most recent build of the software, removing 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.

- *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, how the variabilities of a product would look. 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 had often been confronted with complex manual update procedures.

To manage the deliverables and ease the maintenance process at the customer site, e-Synergy's PDM module was employed. The PDM functionalities were implemented as a central system to the process of maintenance. e-Synergy connects the PDM system 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. As all deliverables are currently stored explicitly in the PDM system, the Product Updater can automatically retrieve a list of deliverables for a product from the PDM system and install them if necessary. The fact that all deliverables are retrieved in this way eases the process of software delivery to customers.

Combined SCM and PDM support is provided by the logistics module of e-Synergy because 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 fairly similar to building physical products and can be done using a commercial and generic 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 system 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 artifacts that are part of the products the customers have purchased.

ES uses e-Synergy to manage all tasks with the concepts of tasks and projects. A task has four states, being open, approved (in progress), realized, and processed. These states can be changed by both the assigner and the assignee when appropriate. Projects, which are collections of tasks and subprojects, enable a project manager to assign tasks within a project and view the status of a project by looking at task overviews. A typical task is setup when a bug report is processed by quality assurance to a bug fix request. The request will be assigned by the quality assurance team member to a developer, such that the task comes back to the quality assurance team member after the developer finishes the task.

A typical recurring project is the promotion of a product from the B release to the A release. The release to the A repository requires special care due to the fact that the A repository is open to pilot customers. This requires the product and all its artifacts to be available and up to date, before a release can actually be promoted. A release project thus consists of different subprojects, such as approval steps, artifact testing, and quality checking projects. An interesting aspect of a product release is internationalization. ES uses generic replaceable terms for its applications, which are replaced by dictionaries for each language in which a product is available. Obviously, these dictionary artifacts need to be complete and available at release time. One product release subproject thus is the translation of the dictionaries. Also, since manuals are delivered with the product, the PDM system must contain a valid manual for each language the product is released in.

4.2. Maintenance at the customer

Figure 1 depicts a small subset of the information stored in the integrated system. The CRM system stores information on customers and their contracts. The contracts are used to generate licenses and store what product(s) a customer presently has purchased. The products, as stored in the PDM system, are linked to the artifacts that make up that product and that must be deployed at the customer site when the customer owns the rights to that product. Finally, the artifacts are linked to source artifacts that are used to build the deliverables. The SKB 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 relatively cheap. In addition to 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.

Before e-Synergy was implemented, ES only used a proprietary 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. These lead to the following non-generic requirements, which were met by e-Synergy.

- *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 the 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 realized that 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.

4.2.1. Custom solutions

Ever since ES produced standard out-of-the-box applications, a custom solutions department has been needed to create specific solutions for customers. To do so in e-Synergy, a specific messaging architecture has been created to enable the addition of extra components. Custom solutions are created using a dedicated e-Synergy SDK. Custom solutions produces two types of customizations, being building blocks and customer-specific solutions. Building blocks are standard customizations that are applicable to specific market niches, such as equipment rental companies or educational organizations. Customer specific solutions are requested by customers and are not generalizable to other customers. Finally, customers can purchase the option to create customizations themselves.

All three types of customizations are built using the same SDK and are facilitated by a messaging architecture. The messaging architecture created by ES is kept as stable as possible, such that custom solutions that worked with older versions of e-Synergy do not break down due to an update. If a customization changes a file that belongs to the product itself, such that an update would remove the customization, the customization developer can mark the file as unfit for update. The Product Updater will then simply skip the file during the update process. Obviously, it is impossible to guarantee that the product remains fully functional after an update when changes have been made to the product itself and therefore ES focuses on communicating regularly with its customers that implement customizations.

In case the custom solutions department creates a customization for a customer, that customization is stored in a Custom Solutions SCM. The dedicated SCM system enables customers to update a custom solution automatically when the Product Updater is run. When a product for which a customization is built is updated, custom solutions cannot cost effectively test the customization that was made for one specific customer in each possible updated configuration and only building blocks are tested for each new (maintenance) release. Due to the fact that updates do not break compatibility in the SDK however, customer specific customizations generally do not break down after an update.

4.2.2. 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 and license file are stored. The data file contains a list of all the deployed deliverables and the license file states all the modules that have been 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 is 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.

In an attempt to reduce piracy, a license-checking mechanism was implemented. Periodically the license file is renewed and must be downloaded again to keep the product active. Currently there are no data available to support whether piracy has effectively been reduced. ES is unique because they provide customers with 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 use of license files to encapsulate product instantiation information is common in the software industry.

5. DISCUSSION

The customer base of ES has shown 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 is 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 has enabled ES to attach workflow to them, thereby providing a software maintenance process that 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*. Secondly, ES teaches us that by mapping maintenance of configurations at customers onto the published deliverable repositories simplifies customer configuration updating, because this process is reduced to comparing the list of local artifacts against the list of released artifacts.
- *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. 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 of 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 keeps track of dependencies among source modules only by adding textual notes to sources, which are hard to maintain. 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 system to simplify the maintenance process, thus restricting concurrent development.

ES's implementation of e-Synergy can be seen as a SKB. The SKB consists of (A) a vendor-side SKB that stores all product and component knowledge and (B) a local SKB 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 SKB that has allowed ES to expand its customer base. Before the implementation of the SKB, 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 that this software knowledge is effectively used in other processes, such as workflow management, human resource management, and customer support. As such, the ES case shows that a simplified instance of the concept of a SKB improves a company's ability to handle large numbers of customers.

ES manages six large product lines, with each product line containing approximately ten solution areas (that, in turn, can contain large amounts of products and product modules), consisting of a total of one million lines of source code. The source code is managed by 294 software developers, who commit their sources a number of times per day. Customers connect to the release repository of ES approximately 250 000 times per year to download updates. We thus strongly believe that the integration of the PDM, CRM, and SCM system is the best way to automatically manage an ever-growing number of customers in need of updates, licenses, and support.

6. RELATED WORK

The techniques applied by ES to integrate SCM and PDM are similar to those described by the book on the integration of PDM and SCM systems in Finnish industry [15]. The execution of case studies performed by Tiihonen *et al.* [22], Bosch and Högström [23], and Davenport *et al.* [24] are similar to the way in which the ES case study was performed. All three describe one or more case studies with qualitative, rather than quantitative results for software products and development processes. The work presented by Tiihonen *et al.* and Bosch Högström is different from our research because it focuses on the state of the practice of software product configuration in software product lines and because both apply a problem-focused approach. Davenport *et al.* described a model of knowledge management that is later put into practice at Siemens, and differs from our work in that their model is first compared with the current practices and then implemented to provide extra grounds for evaluation.

The aim of the presented research is to provide qualitative results to the current body of knowledge of knowledge management and case studies in product software companies in software maintenance in general. The techniques applied by ES are clearly centered around the SCM system, and have been integrated upwards with the CRM and PDM systems. With respect to the SCM system, we have positioned it in the framework created by Conradi and Westfechtel [25]. The SCM system under study, by their definition, is a state-based toolkit that applies a database to store course-grained extensional product compositions for any problem domain. The product configurator applies functional rules to provide an interactional model to its users. Finally, the case studies performed by us [10] and Ballintijn [11] are of the same format as the work presented here. These two case studies show, similar to the ES case but to a lesser extent, that explicit software knowledge management can improve the CCU processes. The main differences between the ES case and these cases is that they describe much smaller organizations (160 and 100 employees, respectively) and that they have not yet integrated PDM, SCM, and CRM to such an extent as ES.

7. CONCLUSION

If anything can be learned from this research, it is that software vendors must integrate their CRM, PDM, and SCM systems to automate the processes related to CCU. Such automation provides more efficient methods to perform repetitive tasks such as license creation, license renewal, product updating, error reporting, usage reporting, product release, and manual configuration tasks, such as backups. The second main lesson is that use of feedback reports supplies software vendors with the largest test bed imaginable, and therefore deserves more attention. The presented research can be used as a guideline for software vendors or for the development of a software manufacturing and software PDM system. In our search for tools that can provide the key practices presented in this paper, an undiscovered product niche was encountered. It seems that there are no (other than ES's e-Synergy) commercial PDM systems that explicitly manage licenses, software products, their fixes, and their patches, in such a way that customers can log in and download them.

This article describes a case study of a SKB at ES. The case study helps to provide evidence that the complex maintenance tasks of enterprise application software for a vendor is best managed with a SKB. 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 also use the results of this case study in comparisons to other case studies we are performing at other software manufacturers [2] and we will use the results to build prototype tools related to SKBs in cooperation with the industry.

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 Heijden, and Nenad Borota for their extensive help. Also, the authors thank Marcel van de Sandt and Marco van Dijk from ES for carefully reviewing and correcting important details. Finally, thanks go to Tijs van der Storm, Vedran Bilanovic, and Paul Klint for their discussions and preliminary reviews of our article.

REFERENCES

1. Xu L, Brinkkemper S. Concepts of product software: Paving the road for urgently needed research. *Proceedings of the 1st International Workshop on Philosophical Foundations of Information Systems Engineering* (*Lecture Notes in Computer Science*). Springer: Berlin, 2005; 523–528.
2. Jansen S, Brinkkemper S. Definition and validation of the key process areas of release, delivery and deployment for product software vendors: Turning the ugly duckling into a swan. *Technical Report UU-CS-2005-041*, Utrecht University, Utrecht, The Netherlands, 2005.
3. Jansen S, Brinkkemper S. Modelling deployment using feature descriptions and state models for component-based software product families. *Proceedings of the 3rd International Working Conference on Component Deployment (CD 2005)* (*Lecture Notes in Computer Science*, vol. 3789). Springer: Berlin, 2005; 119–133.

4. Meyer B. The software knowledge base. *Proceedings International Conference on Software Engineering*. IEEE Computer Society Press: Los Alamitos CA, 1985; 158–165.
5. Klint P, Verhoef C. Enabling the creation of knowledge about software assets. *Data Knowledge and Engineering* 2002; **41**(2–3):141–158.
6. Robillard PN. The role of knowledge in software development. *Communications of the ACM* 1999; **42**(1):87–92.
7. Paulk MC, Curtis B, Chrissis MB, Weber CV. *The Capability Maturity Model: Guidelines for Improving the Software Process* (*SEI Series in Software Engineering*). Addison-Wesley: Reading MA, 1995.
8. Niessink F, Clerc V. *IT Service CMM: A Pocket Guide*. van Haren: Zalbommel, The Netherlands, 2004; 107 pp.
9. van der Storm T. Continuous release and upgrade of component-based software. *Proceedings of the 12th International Workshop on Software Configuration Management (SCM-12)*. ACM Press: New York NY, 2005; 43–57.
10. Jansen S. Software release and deployment at Planon: A case study report. *Technical Report SEN-E0504*, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands, 2005; 21 pp.
11. Ballintijn G. A case study of the release management of a health-care information system. *Proceedings IEEE International Conference on Software Maintenance (ICSM2005), Industrial Applications Track*. IEEE Computer Society Press: Los Alamitos CA, 2005; 34–43.
12. Cusumano MA, Selby RW. *Microsoft Secrets*. Free Press: New York NY, 1995; 550 pp.
13. Jansen S, Ballintijn G, Brinkkemper S. Software release and deployment at exact: A case study report. *Technical Report SEN-E0414*, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands, 2004; 45 pp.
14. Yin RK. *Case Study Research—Design and Methods* (3rd edn). SAGE Publications: London, 2003; 192 pp.
15. Crnkovic UAI, Dahlqvist AP. *Implementing and Integrating Product Data Management and Software Configuration Management*. Artech House: Norwood MA, 2003; 366 pp.
16. Kang K, Cohen S, Hess J, Novak W, Peterson A. Feature-oriented domain analysis (FODA) feasibility study. *Technical Report CMU/SEI-90-TR-21*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh PA, November 1990; 45 pp.
17. van der Storm T. Variability and component composition. *Software Reuse: Methods, Techniques and Tools: 8th International Conference (ICSR-8)* (*Lecture Notes in Computer Science*). Springer: Berlin, 2004; 86–100.
18. Jansen S, Brinkkemper S, Ballintijn G. A process framework and typology for software product updaters. *Proceedings 9th European Conference on Software Maintenance and Reengineering*. IEEE Computer Society Press: Los Alamitos CA, 2005; 265–274.
19. Kim DJ, Ferrin DL, Rao HR. A study of the effect of consumer trust. *Proceedings of the International Conference on E-commerce*. ACM Press: New York NY, 2003; 310–315.
20. Tichy WF. RCS a system for version control. *Software—Practice and Experience,* 1985; **15**(7):637–654.
21. Mei H, Zhang L, Yang F. A software configuration management model for supporting component-based software development. *SIGSOFT Software Engineering Notes* 2001; **26**(2):53–58.
22. Tiihonen J, Soininen T, Mannisto T, Sulonen R. State of the practice in product configuration—a survey of 10 cases in the finnish industry. *Knowledge Intensive CAD* (1st edn). Kluwer Academic: Norwell MA, 1996; 21–40.
23. Bosch J, Högström M. *Product Instantiation in Software Product Lines: A Case Study* (*Lecture Notes in Computer Science*, vol. 2177). Springer: Berlin, 2001; 147 pp.
24. Davenport TH, Probst GJB. von Pierer H. Best practices in Siemens. *Proceedings of the Knowledge Management Case Book* (2nd edn). Wiley-VCH: New York NY, 2002; 336 pp.
25. Conradi R, Westfechtel B. Version models for software configuration management. *ACM Computer Surveys* 1998; **30**(2):232–282.

AUTHORS' BIOGRAPHIES

Slinger Jansen is a researcher at the University of Utrecht in the Organization and Information Department and a member of the Deliver team. His research is directed at strategies of companies to obtain and retain customers. Currently he is researching the processes of release and delivery of software to customers, and the problems associated with these processes, by looking at development, maintenance, release, delivery, and deployment. His strong ties with industry provide a background in practical processes and methods.

Gerco Ballintijn completed his PhD on locating objects in wide-area distributed systems at the Vrije Universiteit in Amsterdam, the Netherlands. He currently works as a scientific staff member in the SEN1 group at the Centrum voor Wiskunde en Informatica (CWI) in Amsterdam. His research focuses on configuration management problems during the development, release, and deployment of software, particularly those problems caused by software evolution. His research is part of the Deliver project.

Sjaak Brinkkemper is professor of Organization and Information at the Institute of Information and Computing Sciences of the Utrecht University, the Netherlands. Before he was a consultant at the Vanenburg Group and a Chief Architect at Baan. Before Baan, he held academic positions at the University of Twente and the University of Nijmegen, both in the Netherlands. He holds a BSc of the University of Amsterdam, a MSc and a PhD of the University of Nijmegen. He has published five books and more than 100 papers on his research interests: software product development, information systems methodology, meta-modelling, and method engineering. He is a member of the ACM and the IEEE Computer Society.

Arco van Nieuwland served in his latest position as Commercial Director e-business for Exact Software. As such, he had ultimate responsibility for e-business within all areas of the company, including operations, marketing, and business development. He co-founded Exact Software in 1984. Exact was taken public on Euronext Amsterdam in June 1999. Van Nieuwland served many positions within Exact, mostly related to starting new activities and new products. In the early days Van Nieuwland has been hands-on programming various ERP solutions. Today Exact has subsidiaries in 40 countries and over 160 000 customers in 50 countries. Van Nieuwland is still highly involved as shareholder of the company and operates as independent entrepreneur in IT.