Benchmarking the Customer Configuration Updating Practices of Product Software Vendors

Slinger Jansen Sjaak Brinkkemper Remko Helms
Information and Computing Sciences Institute
Utrecht University, Utrecht, The Netherlands
{s.jansen, s.brinkkemper, r.helms}@cs.uu.nl

Abstract

Product software vendors do not invest enough effort on release, delivery, deployment, and usage and activation of their software products. Not spending effort on these customer configuration updating processes leads to high overhead per customer, which impedes growth in customer numbers. This paper presents the results of a survey that provides product software vendors with an overview of their customer configuration updating processes and practices, and benchmarks theirs against competitors using similar technology, of the same size, and active in the same market. These benchmarks contain customized advice to the respondent company that can be used to strategically improve customer configuration updating processes to gain efficiency and effectiveness. The survey was held in the Netherlands, and 74 software vendors responded. Amongst other conclusions, a significant positive correlation was found between success of a software product and a vendor's recent investments into customer configuration updating.

1 Introduction

The product software industry in the Netherlands is flourishing. Computer games, ERP products, and navigation systems are just some examples of successful products, nationally and internationally. Though these businesses have a large body of knowledge available to them about generic software development and engineering, none of it is specific to product software development. One area that is specific to product software vendors is the fact that they have to release, deliver, and deploy their products on a wide range of systems, for a wide range of customers, in many variations. Furthermore, these applications constantly evolve, introducing versioning problems. This paper presents a benchmark survey into the customer configuration updating processes of 74 product software vendors.

To date product software is a packaged configuration of software components or a software-based service, with auxiliary materials, which is released for and traded in a specific market [22]. Customer configuration updating is defined as the combination of the vendor side release process, the product and update delivery process, the customer side deployment process, and the usage and activation process [13]. The release process is how products and updates are made available to testers, pilot customers, and customers. The delivery process consists of the method and frequency of update and knowledge delivery from vendor to customer *and* from customer to vendor. The deployment process is how a system or customer configuration evolves between component configurations due to the installation of products and updates. Finally, the activation and usage process concerns license activation and knowledge creation on the end-user side.

Vendors encounter many problems when attempting to improve customer configuration updating. To begin with, these processes are themselves highly complex considering vendors have to deal with multiple revisions, variable features, different deployment environments and architectures, different distribution media, and dependencies on external products [12]. Also, there are not many tools available that support the delivery and deployment of software product releases that are generic enough to accomplish these tasks for any product [14]. Finally, CCU is traditionally not seen as the core business of vendors, and seemingly does not add any value to the product, making vendors reluctant to improve CCU.

This paper presents the results from a survey of 74 product software vendors. The respondents are product managers for one product from one product software vendor. The object under study is the release, delivery, deployment, and usage and activation processes from each vendor for one of its products. These four CCU processes consist of two to four practices. Each practice consists of capabilities. A vendor's capabilities are measured using between three and five questions per capability. The aims of these scores are to establish a vendor's position compared to competitors. The overall goals of this research are to see what the CCU landscape in the Netherlands looks like, to establish whether CCU process scores are directly related to product success, and whether a survey can be used as a knowledge dissemination method.

In the following section the processes and practices of customer configuration updating are described in detail. In section 3 the research design, consisting of the hypothesis and research approach, is presented. In section 4 the survey results and respondents are presented, including the results for each process area and for the hypotheses. Finally, in section 6 we present our conclusions and discuss our future work in regards to CCU process improvement.

2 Customer Configuration Updating

The four process areas and their corresponding practices used in the benchmark survey are defined using the SPICE model for process assessment [2]. The SPICE model, which enables self analysis for vendors, defines a process as "a statement of purpose and an essential set of practices that address that purpose". These practices are software engineering or management activities that contribute to the creation of work products of a process or enhance the capability of a process. In this section CCU and practices are defined together with their relationship to the CCU model and current literature.

We define CCU as the release, delivery, deployment, and usage and activation processes of a software vendor. These processes consist of two to four practices, each with a number of elementary capabilities. For instance, the release process is made up of four practices. One of these practices is release frequency and quality. The capabilities falling under the release frequency practice are that a vendor must frequently release major, minor, and bug fix releases, that a vendor must synchronize these releases with customer demand, and that releases are tested by pilot customers before they are made publicly available.

The **Release process** is based on four release practices. The first practice is how often versions and updates of a product are released and how this is planned within the organization. The second practice is how releases are shared within the company and between customers and the vendor. Thirdly, all dependencies between components, be they products that have been built by the vendor or purchased from another, must be managed by making explicit dependencies between these products and components. Finally, versions of external components, such as components-off-the-shelf, must be managed explicitly to maintain high quality releases.

With regards to the **delivery process** there are two practices. The first practice prescribes that vendors must use every possible channel for the distribution of products and product updates [9]. The second practice states that every possible method for delivery must be applied, such as automatic push or pull.

There are four practices for the **deployment process** of a vendor. To begin with, a product must be removable without leaving any remnants of data on a system. This is required because a new installation preferably must not be contaminated with old data. Secondly, if issues are encountered during the deployment of a software product, automatic resolution must take place to resolve these issues. Such resolution mechanisms are capabilities such as automatic downloading of missing components, freeing up resources when required, or even automatic renewal of licenses. The third practice for the deployment process is that updates and installations must be able to cope with customizations made by customers or third parties. A vendor supports this practice when a special software architecture is in place that enables customizations. The fourth practice is deployment reliability, which is ensured by capabilities such as validity checks, feedback reports, and externalization of customer specific changes and data [7].

Finally, a vendor's **activation and usage process** is based on three practices. First, a vendor must (semi) automatically handle all license requests and distribute licenses with a maximum amount of flexibility within the organization. A vendor supports this practice once customers can explicitly manage their licenses, licenses expire, temporary licenses can be generated for sales and test purposes, and licenses are generated automatically once a sales contract is signed. Secondly, vendors must make use of feedback to gain as much knowledge about the product in the field as possible. A vendor is considered adequate for this practice once it makes use of both usage reports and error feedback. The third practice is that a vendor must be aware of its customers' configurations. A vendor scores for this practice when it is aware of the software and hardware components that are used by its customers.

Process	**Practice**	**Average score**	**Maximum score**
Release	Releases are frequent and of high quality [13]	6.08	17
	The vendor maintains an explicit release planning [12]	2.26	7
	a formalized release scenario that describes what happens on release day [13]	8.73	18
	manage external products, such as commercial-off-the-shelf components [10]	4.73	9
	Total	**21.8**	**51**
Delivery	Vendors must use every possible channel to stay into frequent contact with customers [9]	11.91	30
	every possible method for delivery must be applied [13]	11.28	38
	Total	**23.19**	**68**
Deployment	Explicit dependency management for correct deployment [6, 21]	3.35	7
	The product can be uninstalled and rolled back [14] [18]	2.7	4
	The vendor uses update tooling and manages customizations [8] [14]	11.8	26
	Update reliability and semi-automatic problem resolution [12]	14.49	32
	Total	**32.34**	**69**
Usage and activation	Licenses are can be renewed and trialled and activate components of the software [13]	6.97	7
	Licenses are explicitly managed within the organization and generated from contracts[13]	2.65	9
	Vendors know how customers use products and take advantage of this knowledge [23, 15]	9.24	14
	Total	**18.86**	**30**
All	**Cumulative Total**	**96.19**	**218**

Table 1. Scores for Practices and Processes

3 Research Design

This research has been conducted to find out more about the CCU practices and processes of product software vendors, to benchmark a product software vendor's practices, and to generalize some of the conclusions of earlier work we applied in a multiple case study [13]. The work has been conducted for two reasons. The first reason is to perform a benchmark for product software vendors about their release, delivery, deployment, and usage and activation practices. Secondly, we wished to prove or disprove the following hypotheses:

H1: A product software vendor's scores for the CCU processes are positively correlated to the age of the company, the age of the product, the number of natural languages in which the product is available, the number of developers working on the product, and the number of customers of the product. Many different tools are built around software products during their lifecycle. Furthermore, customers come and go, making the need for good CCU processes and tools even larger. Also, as a vendor's customer base grows, this needs increases further. Also, with more developers on board, more developers can work on CCU improvement. We therefor expect H1 to be true.

H2: A younger technology platform will result into lower scores for a product software vendor for the CCU processes. When a product software vendor starts using a new technology there are not many CCU support tools available for that technology. This implies that older technology platforms will have better CCU support, improving a vendor's CCU score.

H3: Recent changes in CCU processes are directly correlated to product success. When a product software vendor changes the CCU process it improves customer experience and enables a product software vendor to spend less resources per customer. This frees up resources to further develop the product or do more maintenance. This hypothesis is two sided, however, due to the fact that a more successful product software vendor will have more resources available to spend.

3.1 Approach and Survey Design

The research hypotheses are proved or disproved using a web survey that establishes scores of CCU processes and different maturity indicators for product software vendors and their products. The benchmark survey consists of eleven parts. Each part contains between three to twelve questions. There are both closed and open ended questions in the benchmark survey. The closed questions are used to establish scores for practices of vendors and consist of yes/no questions and multiple choice questions. The open questions establish the generic or numeric information on the vendor, such as the vendor's name, the amount of customers of the prod-

Practice	Question or description	Average Score of all vendors	Highest possible score
A1	How often do you publish a major release of your product?		5
	How often do you publish a minor release of your product?		5
	How often do you publish a bugfix release of your product?		5
	Are releases timed in sync with customer requirements?		2
Total	**A1: The vendor addresses release package planning to maintain high quality releases[13]**	6.08	17
A2	Does your organisation use a formal release planning that states the times until the next major, minor, and bugfix releases?		5
	Is this planning published in such a way that all related personnel can access it?		1
	Is there a formal publishing policy towards the outside world in regards to this document?		1
Total	**A2: The vendor maintains an explicit release planning [12]**	2.26	7
A3	Is there a step-by-step description (release scenario) of what happens on the day of a release?		2
	Releases are stored in a versioned repository?		6
	All major, minor, and bugfix releases can be downloaded by all product stakeholders?		6
	The latest release can be downloaded by all stakeholders?		4
Total	**A3: a formalized release scenario that describes what happens on release day [13]**	8.73	18
A4	All tools that are built to support CCU are managed as if they are externally acquired products.		2
	All other tools (such as development tools) are managed explicitly as well?		2
	Are these components stored in a versioned repository?		5
Total	**A4: manage external products, such as commercial-off-the-shelf components [10]**	4.73	9
All	**Release Process**	**21.8**	**51**

Table 2. Scores for Release Practices and Processes

uct, the amount of users of a product, etc. Three open ended questions have been added to the end of the survey to find out in which parts of CCU the vendor will invest in the future and what tools they feel are missing in the range of CCU support tools.

Each of the practices stated in Section 2 has three to five questions that assesses the vendor's **practice score**. Each of the eleven benchmark survey parts concerns a maximum of three practices, as to create a coherent series of question subjects. The practices have been derived from the CCU model, as described in Section 2. When all practice scores for one process are added up we obtain the **process score**. As is suggested by Saywell [20], methods that force your public to evaluate their own process actively by participation (in a benchmark survey, for instance) are a valid method for knowledge dissemination. When sending out the benchmark report we attached a small paper survey, with a stamped return envelope, to evaluate Saywells claim. We received 26 responses of which only one was negative. All others responded positively to the usefulness of the benchmark report as both a knowledge dissemination tool and as being useful to use in future improvement projects. We define the following validity threats:

Construct validity - In this study concepts come directly from literature and from our earlier practical case study work. The questionnaire was pre-tested by a small working group of four software vendors and by five researchers. Their comments lead to approximately 10% of the survey having been changed. These four software vendors were excluded from participation. **Reliability -** To stimulate correct answers the submitters were all promised a full report, complete with customized advice to specifically fit their product. Furthermore, we promised the submitters that the results would only be published anonymously. **External validity -** We strongly believe that the results of this survey are generalizable to other countries. The OECD states that the Netherlands is the fourth largest exporter of product software in the world. It must be noted that three of a handful of larger software vendors in the Netherlands replied. Even though we estimate that our dataset covers 6 percent of the total number of software vendors, we expect this number to be much higher when looking at the number of employees active in the product software industry.

3.2 Sample Selection

The respondents have been selected based on 2 criteria: First the submitter must be a product manager or a development manager who is close to the process and knows answers to each question. Secondly, the software product must specifically be a software product that is delivered to customers and executed at the customer's site. These requirements are specified in the invitation e-mail and at the beginning of the benchmark survey.

The vendors have been selected through the Platform for Product Software [4], the yellow pages, and the Netherlands business index for ICT [5]. The potential respondents have been approached by e-mail twice. No two respondents belonged to the same product software company.

3.3 Survey Tool and Benchmarks

The open source PHP/MySql tool used for this benchmark survey is called PHP Surveyor [3]. The tool is still under development and the fact that it is open source has enabled some final customizations, such as the addition of question types and custom data analysis methods. The survey tool has been selected specifically because the submitter can save the survey results up to a certain point, to allow him or her to find answers to questions he or she does not know at the time of asking. The previous answers can then be reloaded at a later point.

The benchmark survey is an initiative of the Platform for Product Software [4], a scientific initiative that attempts to unite Dutch product software vendors to improve the industry by disseminating knowledge from the academic world. The benchmark was created to serve both our academic interests and the interest of the platform. To be of use to the platform the results from the survey are used to automatically generate a customized benchmark report for each of the respondents. Such a report includes comparisons of a respondent's practices to other product software vendors using similar development technology or are active in the same market. Besides providing a dataset for research, the report spreads knowledge and provides new insights to product software vendors. We are currently unaware of any comparable initiatives and benchmarks that focus specifically on product software development.

One question asks the software vendor to prioritize six reasons for CCU improvement. These CCU improvement priorities were to serve more customers, serve customers more cost-efficiently, reduce deployment problems, reduce the time in which bugs are found, shorten release cycles, and apply a more flexible pricing model. Furthermore, the customized benchmark reports include specific advice for each of the product software vendors. The advice, such as "You must introduce a formal release scenario" included a full description (1 paragraph) and a relevance indicator. This relevance indicator (*low*, *medium*, and *high*) is based on the above mentioned prioritization question. For example, if the respondent put "Shorten release cycles" as a top priority, the release scenario advise example received a *high* relevance rating. Each advice was assigned to between one and three CCU improvement priorities, based on the CCU evaluation model.

To evaluate whether the benchmark reports are a useful tool for knowledge dissemination, a second survey was sent with the benchmark report. This second survey contains questions such as "Has the report provided you with new insights into the release process?"

4 Results and Respondents

In table 1 the average process and practice scores are listed for CCU. The first column listst the process for which this practice is applicable. The second column describes each practice and provides references to other work in which these practices are described and discussed. In the third column the average score is provided for each practice, obtained by the vendors. The final column lists the maximum score that could have been earnt. The processes and practices of the CCU evaluation are based on the CCU model [13], the SOFA model [19], and some capabilities of the Software Dock [9] and have been recorded into the SPICE based evaluation model. Please note that the total scores for each process cannot be compared to the scores of other processes. We believe that the four processes are equally important. In table 2 the average scores for the release practices and process are listed. The table describes the questions that are posed to determine each practice score for the processes. The scores that can be earned per question have been determined by a committee of five representatives from four product software vendors. The scores for each practice are determined by between two and five questions each. For a full score listing, we refer to [11].

74 product managers submitted the survey, from software vendors ranging from 1 to 460 employees (see table 3). Three were excluded from the dataset on the grounds of being from the wrong country or for being a web service provider. Statistics Netherlands estimates that there are 1400 software vendors in the Netherlands, giving a 5 percent coverage of the total product software industry. Furthermore, when asked the product managers to categorize their business by checking at least two categories (see table 3). By far the largest part of product software vendors is building business productivity tools and enterprise resource planning systems. The product managers were also asked to provide the development technologies they used (results in table 3). The Netherlands has many one-man software companies (16 benchmarked in this survey). This partly distorts the results of the overall survey. Currently we are gathering

Distribution of respondents					
# Employees		Product Categories		Development Technologies	
# Employees	# Respondents	Category	# Respondents	Development Technology	# Respondents
1	16	Business productivity	51	Java	18
2-4	10	Internet	36	C++	14
5-7	8	Health care	5	dotNet	14
8-10	6	Home user	5	Pascal	14
11-20	9	Financial	4	PHP	12
21-50	10	Embedded	4	ASP	9
51-75	8	Extensions	3	C	8
76-100	3	Media	2	Basic	8
101-300	2	Games	1	Clarion	4
301-600	2			Perl	3
				Lisp	1

Table 3. Distribution of Respondents

more results to draw conclusions about this group by applying a size classification.

4.1 Hypotheses Results

H1: A product software vendor's scores for the CCU processes are positively correlated to the age of the company, the age of the product, the number of natural languages in which the product is available, the number of developers working on the product, and the number of customers of the product. The results of the survey have been used to test this hypothesis.

Unexpectedly, there is hardly any correlation between product age and process scores. Though the scores vary greatly, older products do not necessarily have higher CCU scores. The product age (ranging from 0 to 22 years) only showed a slight upward trend for the deployment process scores, possibly meaning that only the deployment process score is influenced by the age of a product. There exists a strong positive correlation between the number of customers a product software vendor has and its CCU scores, for all CCU processes. The strongest positive correlation is found between deployment process scores and the number of customers. Clearly, the higher the number of customers, the more likely a vendor is to spend time on minimizing overhead by managing and improving CCU processes. In regards to FTE developers there exists a strong positive correlation between all processes except usage and activation (where there is a hint of a negative correlation). An explanation for this phenomenon has not yet been found. The hypothesis thus only holds partly true.

H2: A younger technology platform will result into lower scores for a product software vendor for the CCU processes. To prove or disprove this hypothesis, we have listed the technologies, the average CCU scores, the average separate scores per process, and the technology age.

First and foremost there does not seem to be a relationship between the age of a chosen development technology and the combined CCU process score. Some technologies do obtain much higher scores than others, however. At the top of the list are Visual FoxPro, C#, Visual Basic, and Java. At the bottom of the list are C, C++, and the scripting languages PHP and ASP. Furthermore, there is a strong negative correlation between development technology age and the deployment process score, suggesting that newer technologies have better deployment support.

H3: Recent changes in CCU processes are directly correlated to product success. A series of three questions was asked to establish a causal relationship between CCU improvements and product success over the last two years. To the first question, whether the product was more successful, 62 answered they were more successful, confirming our belief that product software is booming business. Of course there are many more factors that influence the success of a software product. To remove this variable the respondent was asked to value the relationship between product success and recent CCU improvements, from "of no influence at all" to "mostly caused by". We found a direct relationship between the success of a software vendor and its investment into the CCU process over the last two years ($r = 0.26, p \leq 0.05$), taking into account the answers to the question on the relationship between product success and CCU improvements.

4.2 Open Questions

To the open question "Into what area of CCU will your organization soon invest" most respondents answered they would invest into delivery methods of updates (40%) and into update tools and methods (35%). In regards to delivery the respondents explained they wished to deliver through different media such as FTP sites and portals and with different methods, such as scheduled automatic updates. With regards to update tools some answered they were investing into Microsoft Windows Vista and some that they were using commercially available installers such as InstallShield and Wix. The third problem that was discussed frequently was release planning and scheduling (12%).

When asked how recent CCU improvements had affected the product most respondents answered that *higher product quality had been reached* (23%) and that the product had become *more reliable* (21%). A close third and fourth were *less deployment problems* (16%) and *lower development cost* (16%). The less popular choices were *less time to find bugs* (11%), *more knowledge about customers* (5%) and *shorter release cycles* (9%). The submitters answered questions on how many people were active in the areas of CCU (release managers, version system managers, etc) and development (quality assurance, programmers, etc). When dividing the number of employees active with CCU by the number of development personnel, an average of 16% is found. This tells us that research contributions in CCU can potentially affect 16% of development personnel on average at product software vendors.

According to this survey the most important reasons for CCU improvement are to serve more customers and to serve them more cost-efficiently on a shared first place. The runner-up was to reduce deployment problems. In contrast, these three were at the first place of software vendors' lists 19 times each, whereas the other three ended up at the top of the list only approximately six times each. The scores are computed by awarding 6 points for priority 1, 5 points for priority 2, etc.

4.3 Suggestions for CCU Improvement

Each of the 74 product software vendors received a personalized and customized report, with up to eight CCU improvement suggestions per process area. These improvement suggestions were then annotated with a relevance rating, based upon the ordering of a product software vendor's reasons why they would improve CCU. The submitters received an average of thirteen improvement suggestions per survey report.

For the release process vendors generally do not use a formal release planning. Furthermore, many product software vendors set shortening release cycles or reduce the time in which bugs are found as a prime priority. This led to the advice "Introduce a formal release planning and share this within the organization" being issued to 47 of the 74 product software vendors. The respondents received the advice "Store external components and development tools in a versioned repository with the product" least often (13 times out of 74) due to the fact that most of the vendors already do this.

For delivery the advice issued most frequently (47 times) was "Seek contact with your customers more often through alternative channels", especially to those vendors who set "Serving more customers" and "Reduce the time in which bugs are found" as their first and second priority. This advice was especially provided to those who do not use the product itself for knowledge delivery, through pop-ups, for instance. In regards to deployment the advice "Explicitly manage all relationships between external products and yours" was issued 46 times, especially to those vendors who set "Reduce deployment problems" and "Serve customers more cost-effectively" as their top priorities. Another improvement suggestion was 46 times as well, but with a lower average relevance rating, was "Enable your product update tool to facilitate custom solutions by customers and partners" because many product vendors do not leverage the advantages of a software supply network yet [17, 16].

Finally, in regards to usage and activation "Send automatic error reports" was advised 53 times to those software vendors with the top priorities "Reduce deployment problems" and "Serve customers more cost-effectively". Furthermore, improvement suggestions often concerned licenses, such as "Enable the customer to renew their license without your manual intervention" and "Generate licenses automatically after entering a sales contract" (43 and 44 times). With the report sent back to the product software vendors a small survey is included that is used to evaluate the advice.

We received 19 short surveys about the benchmark reports. All short surveys reported that the benchmark had changed their view of the CCU processes and that they increasingly saw the CCU processes as interconnected. Of the 19 surveys two reported that they planned no new changes due to the benchmark report, two reported that they would make large changes to their CCU processes, and the other 15 reported they would make small changes due to the report.

4.4 Exploring Relationships

The benchmark survey delivered us with a large set of data, consisting of answers to yes/no questions, multiple choice questions, and a large amount of open ended questions. We attempted to find new relationships between yes/no questions using two-sided Pearson's goodness of fit chi-square tests. We used an algorithm to find whether relationships existed between all yes/no questions. We (obviously) found strong relationships between conditional questions, where the submitter could only provide an answer if the submitter had answered yes or no to the previous question.

Many obvious relationships surfaced: first, there exists a relationship between whether a product's licenses expire and whether a product software vendor can provide temporary licenses ($r = 0.411, p \leq 0.0002$). Furthermore, if a vendor is aware of how a customer configures the product, it is also aware of what hardware the customer uses ($r = 0.605, p \leq 0.001$) and of the customer's customisations ($r = 0.455, p \leq 0.001$). Also, the number of users is related to the number of languages in which the product is released ($r = 0.338, p \leq 0.003$).

We have defined two major practices, the use of a concrete release planning that states release dates for major, minor, and bugfix releases, and the practice to have a release scenario that states exactly what needs to happen on the day of a release. These two practices are related to eachother ($r = 0.293, p \leq 0.011$). Furthermore we find that release planning is related to the explicit management of development tools ($r = 0.328, p \leq 0.04$), the explicit management of prerequisite components ($r = 0.311, p \leq 0.07$), and the use of an update tool ($r = 0.237, p \leq 0.042$).We find theses relationships easy to explain. As the software supply network surrounding a software product grows, so do its dependencies [16]. These dependencies must be managed explicitly, including a specific planning of when the software product will support newer versions.

With regards to development we found that as there are more product developers, releases ($r = -0.251, p \leq 0.031$) and bugfixes ($r = -0.286, p \leq 0.007$) become less frequent. The same holds for customer contact ($r = -0.25, p \leq 0.048$). If a vendor uses a product update tool, it is probably also aware of the customer specific solutions that have been created ($r = 0.432, p \leq 0.001$). Also, as customer data, settings, and product adjustments are stored externally from the product, it becomes easier to deploy the product in a development, test, acceptation, and production environment ($r = 0.234, p \leq 0.037$).

5 CCU Practice Results

Release - Typically, software vendors will have their major releases checked by an average of 5 pilot customers. Furthermore, bug fixes are published every one month to six months. 10 respondents publish daily bug fixes. Minor releases are published every two months to one year. Major releases are published every year to three years. Please note that only seven respondents did not publish all three types of releases. We can safely assume that the major, minor, and bug fix release schedule is universal. 37% of organizations use a formal release planning with dates attached. Of this 37%, 70% has a formal publishing policy towards this planning.

Only 52% of the respondents have a formal release scenario or plan that describes the steps taken for a release. Releases are stored most of the time on a shared network drive within the organization, with a versioned repository as a close second. Frequently respondents answered that their old releases are available to customers. Tools that have been built by the respondent's organization are only managed as if they were commercially purchased tools in 55% of the cases. 69% of the respondents use components-off-the-shelf integrated into their products. In 75% of the cases these products are saved alongside the product in a release repository, ensuring version compatibility.

Delivery - Product customers are contacted most frequently by e-mail and the software vendor's website. Only 12 of the 74 respondents inform customers using their own product. Bugs are reported most frequently through e-mail, by phone, and by an on-line bug system. Furthermore, 15 of the vendors send automatic feedback reports when the product encounters an error. Customers are kept up to date yearly, every three months, or monthly in regards to product information. Software is mostly delivered through a website, with CD-Rom as a close second. Furthermore, 12 of the respondent's products are delivered by USB stick. 47 of the respondents' products are delivered when a customer manually pulls it from the vendor. Only 12 of the 74 respondents have developed their product to pull updates automatically from their servers on a regular basis. In 55% of the cases software products can be downloaded by the customer from any location, instead of a preset site, encouraging customers to only download an update once from the vendor for any amount of workstations. 75% of the products are sent to customers in full, where later on parts are (de-)activated by a license file. The other 25% deliver the product to customers on a *get what you paid for* basis.

Deployment - Respondents estimate that on average 4.26% of deployments fail and require extra assistance. Some respondents, however, report up to 35% of failed deployments. Approximately one fourth of the respondents use InstallShield to deploy their products. Furthermore, MSI (19%) and ZIP (24%) are popular formats for delivery. Only 3 of the cases deliver their products as open source. 52% of the respondents uses an update tool to update their customers' configurations. Customer data and content is separated from product files by 68% of respondents. Furthermore, 40% of respondents has functionality for correctness checking of a customer's configuration. Product update and installation tools check for harddisk space most often, with the operating system and already present customer data as a close second and third. The least is checked whether the end-user is using the right hardware (6%). 21 respondents report that their deployment tool, if any, cannot automatically (attempt to) resolve deployment problems. For those tools that can resolve deployment problems, the detection and use of data from previous installations is most common.

In 49% of the respondents' products, the product update tool can deal with customizations. Update tools are most commonly used for major and minor updates. The tools are less commonly used for patch updates and content updates. In 34% of the cases the product update tool can update the product at runtime. Furthermore, 85% of the tools can be deinstalled without complex manual steps. Furthermore, in 51% of the cases minor updates can be undone. Finally, in 84% of the cases the product can be installed in a Development-Test-Acceptation-Production environment.

Usage and Activation - 95 percent of the respondents use license files, containing information on the purchased modules, the number of users, and the customer name and address. Customers pay per floating user or per user name most often. Pay per usage is quite popular as well, with 19 of the 74 respondents using it to bill their customers. 45% of the cases uses licenses that expire. 13 product software vendors have mechanisms in place that enables customers to renew a license without the intervention of the vendor. 65% of the cases provides temporary licenses on a regular basis. In 20% of the cases licenses are automatically generated from contracts. 28% of the vendors makes use of usage feedback reports. 78% of the vendors is aware of all customizations that have been built on top of their products. In 30% of the cases software vendors send back error reports in case of a crash or error.

The average scores per practice are high, which explains that product software vendors perform well when it comes to license management. Furthermore, they provide many different reporting features in their products (which are sent back to the vendor in only 30% of the cases, mind). Product software vendors perform well when it comes to license management and product usage reporting. This does not necessarily mean that this area does not deserve attention, though. We believe there is still much research to be done when it comes to the mining of data from feedback reports.

6 Conclusions and Discussion

This paper presents the results of a survey of 74 product software vendors in the Netherlands. The survey was created using the CCU process evaluation model, which is presented in detail. The survey allows us to generalize conclusions from earlier research that CCU processes of product software companies are generally implemented to a very limited degree. Furthermore, the results from the survey show that CCU improvements have a significant positive effect on product success. Finally, the results demonstrate weak points in CCU processes, such as a lack of CCU tools. In earlier work the CCU evaluation model was presented [13], and validated at nine product software vendors. This work, however, could not easily be generalized to be applicable to a larger group of product software vendors. The survey enabled a further detailing of the model, adding weights to practices and capabilities using an expert panel. Furthermore, the survey enables us to validate our hypotheses from the nine case studies and to show that CCU is an area that urgently requires more research.

The survey results show that CCU is an underdeveloped area that requires more attention and tooling. The areas that deserve most attention are the release, delivery, and deployment processes, due to the fact that product software vendors on average only implement between one third and one half of the capabilities of each practice. The main results of such research will improve customer retention rates for product software vendors, product success, and update and deployment reliability.

CCU research and tools should be geared towards release planning, knowledge delivery, component updating, and feedback reports. This is indicated by four facts. Only 37% of the software vendors uses a formal release planning. Secondly, only 20% of the software vendors use their own product to share knowledge with customers, such as product knowledge and product news. In the area of deployment many product software vendors do not perform any configuration correctness checking or provide any checks before installation of a product. finally, only 30% uses automatic error reporting from customers for the usage and activation process. One of the most important contributions of the survey is that CCU improvements show a significant relationship with software product success. When also taking into account that 16% of product software development personnel is involved in CCU related processes, CCU proves to be a fruitful area for research and evaluation.

The fact that many software vendors underperform in crucial CCU processes and the many suggested products by vendors that are currently unavailable on the market, suggest that CCU is a developing process area of software engineering. When looking at software products currently available to support CCU processes it becomes clear that especially in the area of knowledge interaction between customers and vendors there is a lack of support tools. As a result of this we have started working on the Pheme [1] prototype communication infrastructure for knowledge delivery. This infrastructure consists of small server applications that are installed on different nodes (for instance customers, vendors, and release servers) that share different knowledge packages, such as updates, product information, product feedback, etc. We hope to implement the tool at a number of the respondents of the survey.

References

[1] http://www.cgbs.nl/Pheme.

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

[3] http://www.limesurvey.org.

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

[5] http://www.sdu.nl.

[6] A. Carzaniga, A. Fuggetta, R. Hall, A. van der Hoek, D. Heimbigner, and A. Wolf. A characterization framework for software deployment technologies. In *Technical Report CU-CS-857-98, Dept. of Computer Science, University of Colorado*, 1998.

[7] E. Dolstra. Integrating software construction and software deployment. In B. Westfechtel and A. van der Hoek, editors, *11th International Workshop on Software Configuration Management (SCM-11)*, volume 2649 of *LNCS*, pages 102–117, Portland, Oregon, USA, 2003. Springer-Verlag.

[8] E. Dolstra, E. Visser, and M. de Jonge. Imposing a memory management discipline on software deployment. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 583–592. IEEE, 2004.

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

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

[11] S. Jansen. Customer configuration updating in a software supply network. PhD Thesis, Utrecht University, 2007.

[12] S. Jansen and S. Brinkkemper. Modelling deployment using feature descriptions and state models for component-based software product families. In *3rd International Working Conference on Component Deployment (CD 2005)*, LNCS. Springer–Verlag, 2005.

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

[14] 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.

[15] S. Jansen, S. Brinkkemper, G. Ballintijn, and A. van Nieuwland. Integrated development and maintenance of software products to support efficient updating of customer configurations: A case study in mass market erp software. In *Proceedings of the 21st International Conference on Software Maintenance*. IEEE, 2005.

[16] S. Jansen, A. Finkelstein, and S. Brinkkemper. Providing transparency in the business of software: A modelling technique for software supply networks. In *Proceedings of the 8th IFIP Working Conference on Virtual Enterprises*, 2007.

[17] S. Jansen and W. Rijsemus. Balancing total cost of ownership and cost of maintenance within a software supply network. In *proceedings of the IEEE International Conference on Software Maintenance, Philadelphia, PA, USA, September*, 2006.

[18] Object Management Group. Deployment and Configuration of Component-based Distributed Applications Specification. 2003.

[19] F. Plsil, D. Blek, and R. Janecek. Sofa/dcup: Architecture for component trading and dynamic updating. In *Proceedings of the International Conference on Configurable Distributed Systems*, page 43, Washington, DC, USA, 1998. IEEE.

[20] D. Saywell and A. Cotton. Spreading the word: Practical guidelines for research dissemination strategies. WEDC publishers, UK, 1999.

[21] A. van der Hoek. Design-time product line architectures for any-time variability. In *Science of Computer Programming, special issue on Software Variability Management, to appear*, 2004.

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

[23] A. Zeller. Yesterday, my program worked. today, it does not. why? volume 24, pages 253–267, New York, NY, USA, 1999. ACM Press.

Acknowledgements

We are very grateful to the representatives of the software vendors that participated in the DELIVER working group of the Platform for Productsoftware and contributed to the benchmark survey. Furthermore, the authors thank Merel van Geest, Vedran Bilanovic, Ronald Batenburg, Lidwien van de Wijngaert, and Tijs van der Storm for their many inspiring ideas that contributed to this paper.