Abstract

Software Product Line Engineering has emerged as a software engineering strategy aimed at helping industry achieve business goals. Nevertheless, in order to ensure the return of investment with the Software Product Line (SPL) approach, a well-defined Product Derivation (PD) process is important. Without this process, the products are instantiated in an ad-hoc manner with success relying on the effort of a few individual members. This may increase the production costs and time-to-market.

Despite its importance, when compared to the vast amount of research on developing product lines, relatively little work has been dedicated to the process of product derivation. Additionally, there are few available reports about how software development organizations derive their products from a product line.

Thus, this study presents the findings gathered through to the case study methodology in order to enhance understanding of how product derivation is performed in industrial settings, including its key phases and activities in the product derivation process.

Keywords

Case study, Product derivation and Software Product Line Engineering.
1 Introduction

A growing number of software development organizations are adopting strategies that emphasize proactive reuse, interchangeable components, and multi-product planning cycles [1]. In this way, the Software Product Line (SPL) approach has emerged as a software reuse approach, in which reuse is planned, enabled, and enforced. It applies a strategy that plans the use of assets in multiple products rather than ad-hoc approaches that reuse assets only if they happen to be suitable [1].

The SPL approach makes a distinction between domain engineering (where a product is derived based on the platform components [2]) and application engineering (when individual products using the platform artefacts are constructed). The process of creating these individual products from a product line of software assets is known as product derivation [3].

An effective product derivation process may contribute to ensuring that the effort required to develop the platform assets is less than the benefits delivered through using these shared artefacts across the products within a product line [3]. However, despite the importance of product derivation, there are several difficulties associated with the process, such as: the process is slow and error prone [2], it has an inherent complexity [4], [5] and it is still a time-consuming and expensive activity in many organizations [3]. In addition, there are few reports [6], [3] available describing how SPL organizations derive products from a product line.

Due to these difficulties, we performed a case study in an industrial SPL project within the medical information management systems domain. The case study investigated the key phases and its activities in the product derivation process.

The remainder of this paper is organized as follows. Section 2 describes the case study background. Section 3 presents the research results, i.e. an overview of company product derivation process. In Section 4, the results are discussed. Section 5 discusses related work on product derivation. Finally, conclusions and futures directions are presented in Section 6.

2 Case Study Background

This section describes the case study conducted at MedicWare Informatic Systems (http://www.medicware.com.br) located in Salvador, Brazil. MedicWare Systems has been developing integrated management systems for the medical domain since 1994.

The MedicWare product line (SMART) is composed of 52 modules (sub-systems), including more than 918 features. It provides thousands of possible variations among its different features and enables the instantiation of customized products within the medical domain. Thus, a company customer can choose within SMART portfolio, the set of modules and features that satisfy their needs.

The products built on top of the SMART Platform of Core Assets are large and complex technical software systems with hundreds of features. Their infrastructure is composed of several parameter calls, which enables the selection of components and features during product derivation.

2.1 Research Question

The main goal of this study is to investigate and collect information about the MedicWare product derivation process. Evidences gathered in the organization were used to understand how the process is conducted. Hence, the research question of this study is stated as follow:

- RQ: What are the key phases and activities in the product derivation process? According to recent literature [7], [8], these issues are important aspects to be investigated in the area.
2.2 Case and Subjects Selection

The MedicWare systems LTDA was selected as our case study organization because it had a platform of reusable core assets, a product derivation process, and a considerable number of customers to which it provided customized products.

Regarding to the study subjects selection, a set of 15 subjects were selected from different units and areas (Development team, Product Customization Team, Analysis, Deployment and Business). It is important to involve different roles and personalities in a study like this one to get different and complementary point of views [12].

3 Product Derivation Process

The Medicware product derivation process is composed of five phases: Commercialization, Modelling, Analysis, Customization, and Deployment.

3.1 Commercialization

Commercialization consists of four activities: Define Scoping, Commercial Presentation of the Product, Technical Presentation of the Product and Prepare Proposal/Negotiation.

Each activity is detailed in Table 1.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define Scoping</td>
<td>Information about the customer is elicited. An initial survey is performed to establish the reusability of features relevant to the customer.</td>
</tr>
<tr>
<td>Commercial Presentation of the Product</td>
<td>To determine the set of appropriate models and features to be instantiated and integrated to compose the Base Configuration. Director of Sales discusses and presents the Base Configuration and identifies customer’s needs that are not supported by the SMART Platform.</td>
</tr>
<tr>
<td>Technical Presentation of the Product</td>
<td>To present an overview of the features offered by each module and what should be instantiated for the specific customer. Before this presentation, the Product Expert configures assets variants via globally accessible database tables (TableINI) and Configuration Files. Thus, during the Technical Presentation of the Product, each module is described within the customer scope, i.e., each module is described as a base configuration.</td>
</tr>
<tr>
<td>Prepare/Negotiate Proposal</td>
<td>To present an overview of the features offered by each module and what should be instantiated for the specific customer. The customer confirms if the product and features to be implemented are aligned to their needs. With the review meeting, new features can be included, excluded or reprioritized in the feature list. Moreover, a module list that represents the Partial Product Configuration is defined.</td>
</tr>
</tbody>
</table>

Table 1. Commercialization Activities.

3.2 Modelling

The modelling phase involves two activities: Modelling and Kick Off Meeting.

Each activity is detailed in Table 2.
Session I: Session title will be inserted by editors

<table>
<thead>
<tr>
<th>Activity</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modelling</td>
<td>Map the customer requirements and workflow to define the level of reuse in the Base Configuration and create a Partial Product Configuration. The customer needs are elicited and relevant features are selected. The process of customer needs elicitation involves mapping the workflow of the medical unit (customer domain) in Medical Unit Workflow Diagrams which is compared with the Workflow of Medical Domain Diagrams (i.e. workflow supported by platform features) to identify which features will be integrated in the Partial Product Configuration. Deciding which features should be selected is supported by a questionnaire for each module and variation point resolution. At the end of this activity, the customer scope is defined. From this modelling, the core assets that will be instantiated to compose the Partial Product Configuration are defined. Additionally, the features defined will be customized (platform assets that will be adapted) and developed from scratch during the product customization phases.</td>
</tr>
<tr>
<td>Kick Off Meeting</td>
<td>In order to obtain agreement from the stakeholders on the feature list and product scope, the Partial Product Configuration is demonstrated by the Product Expert to key stakeholders. The Scoping Declaration and Deployment Chronogram are presented and the customer specific requirements which cannot be satisfied by the Partial Product Configuration are negotiated.</td>
</tr>
</tbody>
</table>

Table 2. Modelling Activities.

### 3.3 Analysis

The Analysis phase consists of two activities: Specify Requirements and Analyse New Features. This phase occurs when it is necessary to implement new features or adapt existing ones which cannot be satisfied through a configuration of the SMART Platform of Core Assets. When new features are identified, the New Features Analyst interacts with the Platform Architect, in order to analyse and approve feature development. The approved feature is then classified as: Specific Feature, Reactive Feature or Proactive Feature. Each activity is detailed in Table 3.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify</td>
<td>The new features for the customer’s product are specified and detailed. The New Features Analyst or Requirements Analyst can specify the new features during the Modelling activity or the Analysis phase. In order to do it, they make observations, collect documentation and interview potential system users.</td>
</tr>
<tr>
<td>Requirements</td>
<td></td>
</tr>
<tr>
<td>Analyse</td>
<td>Customers can request requirements which are not supported by the platform. When new features are identified, the New Features Analyst interacts with the Platform Architect, in order to analyse and approve the feature development. The approved feature is then classified as: Specific Feature, Reactive Feature or Proactive Feature. The features are classified according to: potential for reuse, its level of complexity.</td>
</tr>
<tr>
<td>New Features</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Analysis Activities.

### 3.4 Customization

During the Analysis phase, features are classified in categories to define their implementation form.
Thus, features classified as *Specific Feature* will be implemented by the *Product Development Team*. On the other hand, new features classified as either *Reactive Feature* or *Proactive Feature* are implemented by the Platform Development Team using *Reactive Product Customization* or *Platform Evolution*.

Figure 1 shows the relationship between features classification, product customization approach and parameters analysed.

![Figure 1. Relationship between feature classification, product customization approach and analysis parameters.](image)

In *Reactive Product Customization*, new features are implemented with incorporated configuration mechanisms that are configured only in the specific product. *Reactive Product Customization* results in platform evolution without impact to other customers. These configuration mechanisms enable the configuration of variants for instantiation of the reusable assets.

Similarly, in the *Platform Evolution*, new features with incorporated configuration mechanisms are implemented and integrated into the *SMART Platform of Core Assets*. These features are then available to other customers as new updates. For *Specific Product Customization*, new features are implemented and integrated in a specific product without configuration mechanisms and therefore cannot be integrated within products of other customers.

Each activity is described in Table 4.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Specific Product Customization</strong></td>
<td>To perform specific product customization including testing and integration of components. During the process, the <em>Product Developer</em> is responsible for component customization. This includes development of new components or adaption of existing platform components. Typically, this type of customization is necessary only for a specific customer and therefore is not integrated with the platform. During <strong>Specific Product Customization</strong>, developers select components which will be adapted to the customer requirements and adjustments are made to each component to accommodate customer’s needs. After customization, the components are tested and integrated to compose the customized product.</td>
</tr>
<tr>
<td>Reactive Product Customization and Platform Evolution</td>
<td>Implement or adapt reusable software assets based on the customer’s needs. Both <em>Reactive Product Customization</em> and the <em>Platform Evolution</em> activities are performed by the <em>Platform Development Team</em>. The customization process is similar for these two activities with exception of <em>the Integration of Components</em>. Once implemented, tested and documented, the new features will be compiled and integrated. However, for the <em>Reactive Product Customization</em>, the parameters for configuration, created during component implementation, will be enabled in this specific product. During component customization, the <em>Platform Development Team</em> interacts with the <em>Platform Architect</em> to obtain detailed information about core asset evolution and the...</td>
</tr>
</tbody>
</table>
constraints of the Platform Architecture. The Platform Development Team implements new features based on customer requirements. In order to implement reusable new features, the component code incorporates a parameterization mechanism. This enables the selection of variants that enable the reusable components.

After implementation and testing, the new feature is documented in the Core Assets Documentation. This includes a description, characteristics of its use, and configuration parameters necessary to enable it. Finally, a release of the module with the new feature is generated. It is integrated in the SMART Platform of Core Assets, and released to a specific customer (if the request has been classified as a Reactive Feature) or to other company's customers (if the request has been classified as a Proactive Feature).

Table 4. Customization Activities.

3.5 Deployment

The Deployment phase occurs incrementally, as each customer feature supporting a module is selected, it is installed and configured. This phase involves three activities: Instantiate Database, Configure Product and Simulation.

Each activity is described in Table 5.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiate Database</td>
<td>A database is selected from the Database Templates. The SMART Platform of Core Assets has five Databases Templates, where each model supports a specific medical speciality and can be customized according to the customer’s needs. With the mapping complete (Modelling activity), the Database Analyst selects the database, which best fits the customer domain. After this selection, the Database Analyst adjusts the base model to support customer needs. Then, the database is installed and available for data entry.</td>
</tr>
<tr>
<td>Configure Product</td>
<td>The product is assembled from reusable assets, which are built by reusing existing platform assets, implementation of non-existing assets in the platform, or implementation of product-specific assets. Iteratively and incrementally, as each module is configured, it is installed and configured. This process continues until all modules which compose the Specific Product have been configured and integrated. The Technical Deployment instantiates each module and configures the correspondent parameterizations. Thus, from this configuration, the variabilities are resolved and a Specific Product is derived based on the customer’s requirements.</td>
</tr>
<tr>
<td>Simulation</td>
<td>Certifies the data and detects whether there are non-conformity within the product derived. During this activity, the users run the system using as parameters the workflows mapped. Thus, the final product is validated.</td>
</tr>
</tbody>
</table>

Table 5. Deployment Activities.

4 Discussion

The paper reports the results of the case study methodology application to elicit the PD process and understand how it is performed within an industrial setting. In order to maximize the benefits from the available sources of evidence, this study followed three principles as defined by Yin [11]: (i) use of multiple sources of evidence; (ii) creation of a case study database; and, (iii) maintenance of a chain of evidence. In this way, we applied three different data collection methods: interviews, documentation, and participant-observation [11]. These data collection methods allowed us to triangulate the evidence, increasing the precision of the empirical research. The approach allowed us to looking at
the studies outcomes in different ways, capturing a set of interesting insights and issues.

Based on the number of customers, features reused and tests performed, the company platform was considered stable. However, regarding to the organization maturity level, none specific CMMI (Capability Maturity Model Integration) [10] process area was found. The process provides an iterative form of product derivation which enables product assembly to occur incrementally. This is one of the important characteristics for a product derivation approach as highlighted by Rabiser et al. [8].

The case study process proposes some interesting insights for product derivation, such as: the product derivation process begins during the sales process, the use of workflow mapping for elicitation of customer requirements, the analysis and implementation of new features in a SPL environment, the use of incremental deployment, and the role of training as part of the product derivation process.

Finally, although the process is deemed effective with the case study, we identified some issues. Firstly, the process is not formally described which can lead to confusion as to individual responsibilities. Secondly, there is no standardization of tools across the domain and application engineering teams. This can make it difficult to re-allocate staff occurring to organizational needs. Finally, the process is heavily dependent on expert knowledge particularly for control of dependences among core assets, traceability, and variability management. There is a high risk of losing important process and technical knowledge if these experts leave the company.

From a comparative analysis with [8], we observed that the MedicWare process provides full or partial support for the key activities in product derivation. The analysis we conducted showed that the details provided by the process and how each activity is performed can be used as a basis for building or improving existing product derivation approaches.

5 Related work

Rabiser et al. [7] identified that there is a growing interest by researchers and practitioners in product derivation. However, there is a lack of research reporting how software development organizations derive their products from a product line and the associated problems [8],[13].

Two relevant industrial case study reports on product derivation were published [6] and [3]. In the first, the authors present five problems and three issues associated with product instantiation. In the second one, the goal was to investigate the source of problems associated with the derivation of individual products from shared software assets.

These studies can be considered good sources of information in the area. However, important aspects associated with the industrial product derivation process and practice [20] were not covered.

Our study presents the results of a case study performed in industrial environment, describing how a product derivation process occurs and what practices are used. The study definition and reporting was structured based on [9] and [11], according to well-defined guidelines which allows the study replication and extension.

6 Conclusion

This paper presents the results of an exploratory case study on a SPL company working in the healthcare domain. We investigate industrial product derivation practices and document our findings. In particular, this paper provides further knowledge of the product derivation area. We present knowledge of industry product derivation. This paper provides empirical findings to demonstrate industrial product derivation practices.

The case study process proposes industrial insights on product derivation, such as: the product derivation process begins during the sales process, the use of workflow mapping for elicitation of customer requirements, the analysis and implementation of new features in a SPL environment, the use of incremental deployment, and the role of training as part of the product derivation process. Finally,
although the process is deemed effective with the case study, we identified some issues.

The findings presented can serve as a comparison for product derivation reporting. Researchers can use this work as a basis for defining, adapting or evaluating their product derivation approaches. Moreover, we expect that other researchers can use our work as a starting point for new industry reports, presenting their experiences with product derivation.

Acknowledgements

This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES[1]), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08 and CNPq grants 305968/2010-6, 559997/2010-8, 474766/2010-1 and FAPESB and by Science Foundation Ireland grant 10/CE/I1855.

7 Literature

8 Author CVs

Leandro Oliveira de Souza
M.Sc. Leandro Oliveira de Souza is an assistant professor at Bahia Federal Institute of Education, Science and Technology (IFBA) and researcher at the Reuse in Software Engineering (RISE) group based in the Federal University of Bahia (UFBA). He has a M.Sc. in Computer Science from Federal University of Pernambuco (UFPE), Brazil. Contact him at los2@cin.ufpe.br.

Pádraig O’Leary
Dr. Pádraig O’Leary is a Research Fellow at the University of Limerick, Ireland. He is a member of Lero - the Irish Software Engineering Research Centre and ARCH – the Centre for Applied Research in Connected Health. Contact him at padraig.oleary@ul.ie

Eduardo Santana de Almeida
Dr. Eduardo Santana de Almeida is an assistant professor at Federal University of Bahia and head of the Reuse in Software Engineering Labs. Contact him at esa@dcc.ufba.br.

Silvio Romero de Lemos Meira
Dr. Silvio Romero de Lemos Meira is a full professor at Federal University of Pernambuco and chief scientist at the Recife Center for Advanced Studies and Systems. Contact him at silvio@cesar.org.br.