Similarity Analysis within Product Line Scoping: An Evaluation of a Semi-automatic Approach

Markus Nöbauer[1], Norbert Seyff[2], and Iris Groher[3]

[1] InsideAx GmbH, 4031 Linz, Austria
markus.noebauer@insideax.at
[2] University of Zurich, 8050 Zurich, Switzerland
seyff@ifi.uzh.ch
[3] Johannes Kepler University (JKU), 4020 Linz, Austria
iris.groher@jku.at

Abstract. Introducing a product line approach in an organization requires a systematic scoping phase to decide what products and features should be included. Product line scoping is a non-trivial activity and traditionally consumes a lot of time and resources. This issue highlights the need to complement traditional scoping activities with semi-automatic approaches that allow to initially estimate the potential for reuse with small efforts. In this paper we present an evaluation of a tool-supported approach that enables the semi-automatic analysis of existing products in order to calculate their similarity. This approach is tailored to be used within the configuration-based systems domain, where we have used it to identify similarity within two types of industrial standard software products. The results of this evaluation highlight that our approach provides accurate results and leads to time savings compared to manual similarity analysis.

1 Introduction

Adopting a product line (PL) approach supports companies to foster systematic reuse [1],[2]. However, the transition from single-system development to a PL approach is often non-trivial and costly [3]. PL scoping is an important activity in existing PL engineering processes [2]. Within PL scoping, companies investigate which existing products are potential candidates for a PL. Therefore, they have to identify the commonality that potential members of the PL share and their variations [4],[5]. Most existing scoping approaches require large upfront investments [6],[7]. Scoping highly depends on the development practices within the organizations, the architecture of the system, and the business domain. In [8] about 16 different scoping approaches and industrial applications have been identified. However, only 6 industrial case studies for these approaches exist. Those approaches aim to deliver a full PL scope while our goal is to deliver a first estimation if further scoping investments are justified.

According to our experience, small and medium sized companies (SME) would require more lightweight, automated and therefore time and cost-effective approaches for initially estimating the reuse potential within their existing products. This is also highlighted by current research in the agile software engineering domain [7],[9],[10]

M. Jarke et al. (Eds.): CAiSE 2014, LNCS 8484, pp. 165–179, 2014.

that emphasis the need for more lightweight approaches. However, agile approaches such as the scoping game [7] require the participation of various stakeholders which could also limit its applicability. Within our research we aim at addressing the bespoke issue by investigating semi-automatic similarity analysis to support PL scoping for configuration-based standard software products.

In this context we presented a first solution [11], which allows to semi-automatically identify key information on the reuse potential of existing standard software product configurations. Such configurations represent large software products such as Enterprise Resource Planning (ERP) systems. These kind of products aim to support a wide range of possible business activities in different industry domains. Therefore, standard software products are by their nature highly flexible and configurable software systems. However, the configuration of standard software products is a time consuming non-trivial task, similar to coding in traditional software development. The goal of our work is to provide a semi-automatic approach to estimate the variability within product configurations. We are aware of the fact that traditional scoping approaches [4],[12] include important activities such as domain assessment and risk analysis and consider product plans or organizational issues. Our automated approach is not meant to replace traditional scoping but to complement it with an initial similarity calculation that could trigger additional scoping measures.

We have conducted two industrial case studies in order to validate our approach. Within the first case study we have applied our approach to ERP system customizations for 5 different customers from different industry branches. The second case study has included more than 50 different business intelligence systems. Both case studies aimed at investigating whether the similarity calculation provides correct results and indicates the start of a product line for the products under comparison. In the first case study we have compared the calculated similarity values to a manual estimation performed by domain experts. Furthermore, we have evaluated the efficiency of our approach in terms of time savings by comparing the manual estimation to the tool-supported calculation. The second case study focused on the scalability of our approach on large sets of products and the extensibility to handle less standardized configurations. We also have evaluated the integration of domain knowledge by using synonym definitions to identify semantically equivalent settings.

The results of the case studies suggest that our approach is capable of semi-automatically calculating the similarity between existing standard software product configurations and thus supports a key scoping activity which is the analysis of existing systems. Domain experts clearly indicated that they would benefit from our tool-supported approach compared to manual methods to investigate the reuse potential within a software product portfolio. Apart from time-savings, our approach provides objective results on the similarity between the different products. This helps to prevent misjudgments and discussions between engineers, which are not based on facts but potentially false opinions.

This paper is structured as follows: In Section 2 we present our tool-supported approach that allows semi-automatic product line scoping. Section 3 and 4 present the conducted case studies and the identified results. Section 5 discusses related work and Section 6 concludes the paper with lessons learned and an outlook on future work.

2 Lightweight Product Line Scoping

2.1 Conceptual Solution

Our approach is focused on standard software products (i.e. customer specific software configurations). These kinds of products are developed and maintained by large software vendors such as Microsoft or SAP and typically contain a rich set of features. Smaller partner companies sell these products to their customers and configure or extend the product to meet the customers' needs. This is mainly done by setting values in available configuration options. In [11] we presented an initial solution to perform tool-supported product line scoping for configuration-based standard software products in order to support these small partner companies. We foresee a semi-automatic approach where customer-specific product configurations are compared in order to identify commonalities and variations. The variability in our case lies in the differences between the configuration settings of the products under comparison. This means each product contains a set of features that are configured according to the needs of a customer. The comparison is based on whether a feature is part of a product and if yes how it is configured. An analyst can provide additional domain knowledge as input to our approach to calculate similarity. Our approach is based on the following steps in order to perform a similarity analysis for configuration-based products:

Step 1: The domain expert selects a set of products for analysis. The selected products are instances of a particular standard product (e.g. ERP system without customizations) that defines the schema of the available configuration settings. The result of this step is a list of products for further analysis.

Step 2: In this step the domain expert defines the scope of the analysis. Therefore the relevant configuration settings that should be compared are selected. Many software products contain settings that do not influence the behavior of the product and are thus not relevant for scoping (e.g. audit settings). Such settings can be excluded from analysis. Moreover, settings can also be grouped if they belong to a logically related feature (e.g. credit card and limit settings). During similarity calculation (cf. step 4) grouped settings are only considered similar if all individual setting values are similar. The output of step 2 is a new configuration schema which only includes relevant configuration settings and newly defined groups of settings.

Step 3: The domain expert can define how the similarity between the selected configuration settings is calculated. Checking on exact equality of settings might often be too strict. Therefore we support an approach that also allows a more fuzzy comparison. The domain expert can define so called *dictionaries* to deal with different naming conventions. A dictionary contains lists of synonyms to identify semantically equal configuration settings (e.g. "revenue", "rev." and "turnover"). The domain expert can also provide upper- and lower boundaries for numerical values of settings (e.g. credit card limits between 2900€ – 3100€ shall be considered similar). These *similarity ranges* are defined using mathematical formulas (e.g. a range of 10% around a default value). We not only support defining similarity ranges for specific configuration settings (e.g. credit card limit) but also for global data types (e.g. Floating Point). For example an analyst can decide to round all floating point numbers. The combination of 1.2, 1.4, 0.9 and 1.7 would be transformed to 1,1,1,2. The output of step 3 is a domain specific definition of similarity for the configuration settings selected in step 2.

Step 4: In this step the similarity analysis is performed by comparing the configuration setting values for the selected products. We compare the configuration values for each of the selected settings of all products to calculate a similarity percentage value per setting. We take the most frequent setting value among the selected products as basis and compare it with all other values. Values are identified as similar if they are identical or if they are within a similarity range defined in step 3. If there is a domain specific similarity defined for a specific setting (e.g. credit card limit between 2900€ – 3100€) and the data type of this setting also has a similarity definition in place (e.g. round floating points), the tool uses the more specific definition from the setting to calculate the similarity value. As an example let us consider the comparison of 4 products with the values *true*, *true*, *false*, *true* for the setting *credit card payment allowed.* We take the most frequent value (in this case the value *true*) as basis and compare all values with it. In our example 75% of the values (3 out of 4) are equal to the base value *true*. In addition we calculate an overall similarity value as the percentage of similar settings within all the products under analysis. For example, if a set of products is compared and out of the 100 settings defined in the schema, 20 are regarded as similar in all product configurations, the overall similarity value is 20%. The results of step 4 are a calculated similarity value for each setting or group of settings and a total similarity value for all products under analysis.

Step 5: Finally the domain expert can draw conclusions based on the calculated results. The analysis results guide the decision whether to invest in a product line.

2.2 Tool Support

We have developed an internally available tool prototype to perform a similarity analysis based on configuration settings. The tool supports importing product configurations from multiple source systems, defining similarity evaluation rules, and comparing the different product configurations. In contrast to the initial prototype described in [11], we have extended the tool's capability to process different types of configuration settings from XML files and SQL data sources.

Setup. The setup of the tool consists of three steps. First the binaries are installed on a Microsoft Windows-based computer. This step requires almost no additional human input. The second step is to configure the connections to the products under analysis. In case where the products use an SQL database only a connection client is required (ODBC). Furthermore we provide add-ons for non-SQL products to export the configurations to XML, for example QlikView (see case study in section 4). In a final step the product configurations are loaded, either from an SQL data source or XML files. The effort to make the tool ready for analysis was on average half an hour for each case study. However, if the products under analysis cannot be accessed either by using SQL connections or one of our plugins for XML export, the tool cannot be used out of the box and additional development effort is required.

Data Management. Meta data is required to handle product configurations. XML schema definitions are used to describe how product configurations are structured. These schemas are either delivered with the standard software product itself, or deduced from the actual configurations. Moreover, domain knowledge about the customer's context is essential in order to tailor the analysis. For example, it can be

advised to compare all products for a specific industry branch. Therefore our tool contains master data about customers and the products in use. Every customer has at least one legal entity. Each legal entity belongs to one or more industry branches (e.g. retail, construction). A legal entity uses one or more products. Each product is built for one or more business areas (e.g. sales, procurement) and there can be multiple products built for one business area. Each product has at least one configuration. This product configuration is an instance of a schema definition. This structure is used to tailor the analysis scope (e.g. compare all product built for sales in retail industry).

2.3 Limitations

A main issue regarding our approach is that in its current form it can only be used for identifying the reuse potential within configuration-based systems (e.g. ERP systems). This means it cannot be applied without modifications in other software domains where the system behavior is not configured. Furthermore, the system behavior may also be influenced by its application data (e.g. the chart of ledger accounts in an ERP system) which is not considered by our approach. We only analyze existing product configurations. This does not reflect all possible product configurations and therefore cannot reveal the complete variability. Each feature has equal influence on the calculated similarity. Currently we do not support weighting of features (e.g. based on their granularity). Focusing on existing product configurations means that we currently do not support other typical product line scoping activities such as planning future products as members of the product line. The tool compares the values of each configuration setting and calculates the similarity. Therefore the effort increases with each additional product under analysis. However, we have conducted a case study on 54 products (see section 4) which can be seen as typical repository size for an SME and the tool was able to finish the calculations within seconds.

3 Industrial Evaluation I – Microsoft Dynamics AX

3.1 Case Study Setting

In order to get insights on our approach we conducted a case study at InsideAx, a small company that is a partner for Microsoft Dynamics AX. Dynamics AX is a business software solution from Microsoft for medium to large enterprises. Partners, such as InsideAx, configure, customize and sell the ERP product to customers.

Standard software, such as Dynamics AX has high reuse potential and would therefore suggest the application of software product lines. However, as partner companies do not own the software product, their influence on the evolution of the system is limited. In fact they have to cope with massive changes within short time periods. This volatile environment and the limited resources of a small company have so far prevented InsideAx to invest in PL engineering.

At InsideAx employees are basically aware of the benefits of PLs and some even believe that the introduction of PL engineering would be beneficial for their company. However, no detailed analysis of existing product configurations was conducted, which leaves the company in uncertainty about the reuse potential of their software.

Furthermore, initial informal discussions among domain experts at InsideAx have indicated that experts might have different opinions on the similarity and thus the reuse potential of particular software products.

In order to eliminate this uncertainty we analyzed their product configurations with the help of our tool. The results could then be used to support decisions on further steps regarding the introduction of PL engineering at the company.

3.2 Evaluation Method

The goal of this study was to provide first evidence that our approach provides correct similarity calculations. Within a small company we used our approach to calculate the similarity between different product configurations to deliver a first estimation if further scoping investments are justified. Particularly we were interested in providing initial answers to the following questions:

Q1: Will the calculated results indicate the need for introducing a PL approach?
Q2: Will the calculated results differ from domain expert estimates?
Q3: Will domain experts benefit from knowing the calculated results?

Steps 1 to 4 were conducted as described in Section 2.1 by one of the authors of this paper, who is also an employee of InsideAx. He first selected five customized products (step 1) which are all derived from the same base product but operating in different industry branches (construction, retail, and manufacturing). He selected the business areas *Purchase*, *Sales*, and *Inventory Management* (i.e. their configuration settings) for comparison as they reflect key business areas (step 2). Next, the domain expert provided similarity ranges (step 3) and the tool calculated a pairwise similarity between the products and the overall similarity value for all three business areas (step 4).

In parallel three domain experts at InsideAx were asked to estimate the similarity of the mentioned products and business areas. They used a scale from 0% to 100% (0% meaning that there is no overlap and 100% meaning that two products are identical). To support the estimation process, they had full access to the configured products, their documentations and requirements. The first expert was an ERP consultant with 3 years experience, the second expert was a data analysis consultant with 4 years experience and the third expert had 2 years experience in developing and customizing ERP products. We then compared the tool-calculated similarity with the domain experts' estimates. Finally, we discussed these results in a workshop with the domain experts.

3.3 Results

The author calculated the similarity for the selected products (P1 to P5) with the help of the tool and spent about 15 minutes in total to perform steps 1 to 4. Table 1 shows the pairwise calculated similarity values for the business activities Sales, Inventory and Purchase.

Table 1. Calculated Similarity for Products (P1-P5)

	Purchase				Inventory				Sales			
	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4
P2	60 %				48 %				58 %			
P3	60 %	40 %			52 %	48 %			54 %	46 %		
P4	53 %	68 %	53 %		52 %	45 %	52 %		42 %	56 %	35 %	
P5	47 %	60 %	47 %	93 %	48 %	48 %	48 %	59 %	39 %	50 %	31 %	65 %

For *Purchase* the pairwise similarity calculations resulted in a similarity range from 40% to 93% (58% on average). In addition to the pairwise comparison the tool also calculated the overall similarity value. Within *Purchase* 33% of all features are similar. The pairwise similarity comparison for *Inventory* provided a range from 48% to 59% (50% on average). In total 41% of all features are similar. For *Sales* the pairwise similarity comparison resulted in a range from 31% to 65%, while the average similarity for *Sales* is 47%. Within *Sales* the total similarity rate is 31%.

Furthermore, we received the estimates of the domain experts. However, one expert was not familiar with all products under analysis and therefore just provided a comparison for those he knew. Table 2 highlights the pairwise similarity estimates. We present the minimum and maximum estimates and the individual estimates for the three domain experts in brackets.

A comparison between the calculated values and the estimates revealed that the expert's opinion and tool calculations differ significantly with hardly any exact matches. To still enable a meaningful comparison, we defined a 20-percentage point interval around each automatically calculated similarity value as a more relaxed comparison criterion. For example, if the tool has calculated a 70% similarity, we defined an interval from 60% to 80%. We then counted the number of expert estimates that were within the defined ranges.

For the business area *Purchase,* eight out of ten estimates do not differ more than 20 percentage points on the scale form 0% to 100%. However, the two remaining estimates on purchase vary between 30 (P3-P5) and 70 (P4-P5) percentage points. For *Inventory* only 4 estimates are within the 20-percentage point limit. This leaves 6 estimates which differ significantly (40 percentage points at most). Also for *Sales* only 4 out of 10 estimates are within the 20-percentage point limit.

The first domain expert who only provided estimates for 18 out of 30 requested comparisons provided 3 estimates, which were within the corresponding intervals. All of them could be linked to the business area *Purchase*. Eight estimates were actually lower, leaving seven estimates to be higher than the calculations. The second domain expert performed best. 5 out of 10 estimates were within the corresponding intervals. All of them were within the business area *Purchase*. 7 out of 10 estimates were within the intervals for the business area *Inventory* and still 3 out of 10 for *Sales* were within the given range. 14 estimates were higher than the tool-calculated results. The third domain expert was able to provide three estimates within the given range. Only one estimate for each business area was within the interval. Again, most estimates were too optimistic (17 out of 23).

Table 2. Similarity Estimates by Domain Experts

	Purchase				Inventory				Sales			
	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4
P2	85 85,85,85				80-95 95,80,95				65-95 65,85,95			
P3	55-75 -,75,55	55-75 -,75,55			70-75 -,70,75	70-75 -,70,75			70-75 -,70,75	70-75 -,70,75		
P4	55-65 55,65,55	55-65 55,65,55	45-55 -,55,45		25-65 25,50,65	25-65 25,50,65	50-75 -,50,75		15-65 15,65,65	15-65 15,65,65	70-85 -,70,85	
P5	35-55 55,55,35	35-55 55,55,35	25-55 -,55,25	25-95 95,75,25	25-65 25,45,65	25-65 25,45,65	55-75 -,55,75	60-85 85,60,65	15-55 15,55,55	15-55 15,55,55	60-75 -,60,75	55-85 85,60,55

3.4 Findings

Will the calculated results indicate the need for introducing a PL approach? The case study was conducted for customized ERP products. While ERP software is commonly known as Standard-Software, we expected a high reuse potential. Reviewing the calculated similarity results, this assumption seems to be somewhat correct. In total, 31% to 41% of the features were similar in all the customized products, regardless of industry branch or business area. A pairwise comparison indicated even higher values. Although we have not defined a clear schema in order to decide which result indicates to introduce a PL, we expected a significantly higher overall similarity value than 50% for each business area. This assumption was based on the published scoping threshold in [6] where 50% is understood as break-even point. However, as we couldn't identify a similarity value higher than 41% we conclude that the results do not clearly indicate the need for introducing a PL approach.

Will the calculated results differ from domain expert estimates? The results highlight that tool calculations and expert estimates vary strongly. Therefore, we performed a more detailed manual analysis of the product configuration settings together with the domain experts. The tool's calculations were validated and for selected settings the corresponding product configurations were analyzed. Although no overall similarity was calculated manually, the experts agreed that the tool's calculations were correct and can be seen as the ground truth for this comparison. Comparing calculated and estimated similarity shows that 33% of the estimates were within a 20 % points interval around the calculated results. In general the domain experts estimated a higher similarity than tool calculations revealed. The discussion with the domain experts in the debriefing session revealed the reasons. In many cases new projects are based on existing customized products from past projects. The domain experts know this and somehow assume that these products show high similarity. However, in reality, these products undergo major changes in most cases and diverge. Table 2 indicates this effect comparing product P1 and product P2, which was initially built based on P1. Although the product has undergone major changes, domain experts still assumed high similarity between the products. In general the domain experts make their estimates based on their experience. None of the domain experts performed a detailed comparison of all product configuration settings, which would not have been possible in a reasonable amount of time.

Will domain experts benefit from knowing the calculated results? Having a different view on the overall situation might lead to conflicts between domain experts. In the debriefing workshop we told domain experts about the tool results and compared them to their estimates. Most experts did not expect that the products were tailored to individual customer needs to that extent. They liked the fact that they could compare their estimates with the calculated values. They argued that the calculated values provide guidance and support in resolving conflicts between domain experts.

3.5 Threats to Validity

Conducting one single case study does not allow computing any statistical significance regarding the correctness of the calculated similarity values, which is a threat to *conclusion validity*. Discussions with domain experts and a conducted manual analysis revealed the appropriateness of the calculated similarity in this case.

One of the authors used our tool to calculate the similarity values within this study, which can be seen as a threat to *internal validity*. However, the debriefing meeting with the consultants revealed the correctness of the input and the configuration.

Our similarity calculation mechanism was developed for the domain of configurable software products. However, our study only focuses on Dynamics AX. This can be seen as a threat to *construct validity.* Also, we did not utilize the dictionary concept in the first case study.

The size and the number of products used in our study were limited, which is a threat to *external validity*. We focus on configuration-based systems and do not provide support for a broader system range. Input from a skilled domain expert, who is familiar with the tool is needed. Not yet investigating the tool's usability in more detail can be seen as threat.

4 Industrial Evaluation II – QlikView

4.1 Case Study Setting

The second case study at InsideAx was focused on a product for data analysis and business intelligence. Domain experts at InsideAx use a product named QlikView to build data analysis applications for their customers. The process of building these applications contains three key steps. In a first step QlikView extracts, transforms, and loads data from multiple source systems such as ERP systems. A domain expert defines so-called measures in a second step. A measure in QlikViev is a formula to calculate a business relevant value (e.g. contribution margin). In a last step the measures are visualized and set in relation to other measures (e.g. revenue in contrast to contribution margin). These analysis results are used to lead a company and optimize a departments' work.

Business intelligence applications such as QlikView are built in tight collaboration with the customers' decision makers, and therefore are more individual than a standard software product. Still, domain experts at InsideAx believe there is a common set of recurring QlikView measures on most business areas and industry branches. However, no in depth analysis has been conducted so far.

Within this study we analyzed a set of customer specific products in order to identify similar QlikView measures. The products are characterized by business area and the customers' industry branch.

4.2 Evaluation Method

The goal of this study was to show that our similarity calculation approach works with a larger example (scalability of the approach). Furthermore, the dictionary concept was applied for the first time. In contrast to the first case study, the size of the system under analysis does not allow comprehensive manual calculations or estimates in a reasonable amount of time. This is why we did not include domain experts in this case study. We have framed our evaluation goal in three research questions (RQs):

Q1: Will the calculated results indicate the need for introducing a PL approach?
Q2: Will the dictionary concept influence the calculated results?
Q3: Will the approach be scalable to the given problem size?

In step 1 we selected 54 different products from 12 customers in 6 different industry branches. All of these 54 products use the data analysis capabilities of QlikView. In contrast to the first case study these products are not derived from a common base product. Each of these 54 products can be seen as individually developed product. QlikView only defines how measures are created but does not include a predefined list of measures (such as a configuration schema in an ERP system). Therefore we defined an initial dictionary containing synonym lists of measures to allow a domain specific comparison. For example, the measures *Profit Margin* and *Contribution Margin* are identical and should therefore be identified as similar during the similarity calculation. The dictionary containing the synonym lists was created by one of the authors of this paper who has 6 years experience in developing business applications.

In step 2 we grouped the products by business area (e.g. Sales, Finance, and Project).

In step 3 we incrementally refined the dictionary while importing product configurations into our tool. We assigned newly imported measures to existing synonym lists if they were semantically similar. For example, the imported measure *Profit Contribution* was added to the synonym list that already contains the measures *Profit Margin* and *Contribution Margin*.

In a next step we performed a similarity analysis on the groups of products built for a specific business area (step 4). The tool identified the number of measures occurring in all products within this group and calculated a similarity value.

4.3 Results

In total, we analyzed 54 different products from 12 customers in 6 different industry branches. These products were grouped into 7 business areas. In total all products together contained 930 measures. We defined a dictionary containing 27 synonym lists to group semantically equal measures. 109 measures were highly individual and could not be grouped with others.

The author spent about 40 minutes to create the initial dictionary (step 1) and 5 minutes to group the products into business areas (step 2). Step 3, the refinement of the synonym lists, took again about 45 minutes. The calculation itself has been performed within seconds (step 4).

Table 3 shows the results from the analysis based on the product's business area. The second column shows the number of products per business area. The number of products varies between two for *Project* up to 17 for *Sales*. The third column contains the number of customers running a product for this business area. For example there are two customers running a product to analyze *Project* but 10 customers analyzing *Sales*. The next column shows the total number of identified measures within the products per business area. For the business area *Project*, 29 measures were identified while for the business area *Sales* we found 460 measures. The fifth column shows the number of synonym lists used to group the measures. This means that the comparison of measures can be reduced from the number of measures to the number of synonym lists. For example, for the business area *Sales* the tool had to compare 64 measures instead of 460 measures because they were identified as semantically equal in the dictionary. The last column shows the calculated similarity of all products in the given business area. For the business area *Project*, the two compared products are 63% similar. This means 63% of identified measures are used in both products. In the business area *Sales*, only 8% of the identified measures (or their synonyms) are present in all 17 products. In general, the calculated similarity decreases with the number of compared products.

Table 3. Calculated similarity for products grouped by business area

Business Area	# Products	# Customers	# Measures	# Synonym lists	Similarity
Project	2	2	29	15	63%
Production	3	2	46	18	50%
CRM	6	3	55	19	25%
Procurement	6	4	71	27	23%
Finance	9	7	118	26	16%
Warehouse	11	7	151	34	14%
Sales	17	10	460	64	8 %

4.4 Findings

Will the calculated results indicate the need for introducing a PL approach? Providing a clear answer to this question is difficult in this case. Comparing the calculated similarity to a threshold of 50% like published in [6], the results suggest that no PL approach is needed and a single-system development approach can also be perused in the future. In case where the calculated results reach the threshold value (see business areas *Project* and *Production* in Table 3) only two products were compared. In all other cases the calculated similarity was significantly lower than the desired threshold. However, our study also indicates that the different granularity of the products under investigation had a significant impact on the results. For example we found five very specialized *Sales* applications built for one customer. A way to address this issue

could be to merge these specialized applications and compare more general applications in a next evaluation. In order to at least partly assure the validity of the calculated similarity values, we performed a manual similarity analysis for 6 products with 71 measures from the business area *Procurement,* which is about 10% of total products under analysis. This manual analysis revealed the correctness of the calculations.

Will the dictionary concept influence the calculated results? Solutions developed with QlikView are in general harder to compare as less standardization is available. Therefore, we introduced the dictionary concept. We observed that synonym lists within the dictionary have a significant impact on the calculated results. We initially provided a predefined set of synonym lists. Therefore many terms were mapped to these synonyms and the reported similarity value was high. With ongoing analysis we refined the set of synonym lists by splitting existing synonym lists into smaller more precise lists. As a result the calculated similarity decreased. This also means that the calculated similarity strongly depends on the analyst's domain knowledge and ability to define adequate synonym lists for domain specific terms.

Will the approach be scalable to the given problem size? The study has shown that our approach can be used with a larger number of products. In contrast to the first case study where 5 products were compared in 3 business areas we managed to analyze 54 products in 7 business areas. We could not identify any performance problems within this case study as the tool calculated the similarity within seconds. However, the manual definition of the dictionary (step 1) and the iterative refinement (step 3) consumed a significant amount of time (about 1,5 hours in total).

4.5 Threats to Validity

Although the example is larger than the one in the previous case study and we can draw first conclusions regarding the scalability of our approach, we still cannot compute any statistical significance regarding the correctness of the calculated results. This can be seen as a threat to *conclusion validity*. More case studies would be needed to further evaluate the correctness of the similarity calculation in different settings. The number of products in some of the business areas (such as *Production* and *Project*) was very limited. Manual analysis revealed the correctness of the calculations for a limited set of products in the business area *Procurement*.

As one of the authors of this paper was also the developer of the tool similar threats to internal validity as described for the first case study (cf. Section 3.5) apply. Particularly, the dictionary had a significant influence on the similarity calculation in this case study. Although the dictionary was built iteratively, which allows for validation and correction, it was created and validated by one person only.

Similar to the first case study, we focused on a specific product (QlikView), which is a threat to *construct validity*. Although we applied the dictionary concept, we did not use similarity range definitions.

As discussed in the first case study, or approach focuses on configuration-based systems. This could mean that it underrepresents *external validity*. A domain expert who is familiar with the tool is needed. Although in this study we successfully used our approach in a larger setting, we did not investigate the tool's performance in detail. However, we expect the problem size of the second study to be typical for SMEs.

5 Related Work

Schmid and Schank [6], PuLSE-BEAT is introduced, a tool for supporting the product line scoping approach called PuLSE-Eco presented in [13]. To identify the optimal scope, a product map is used which is a matrix with the product characteristics (features) on one axis and the products on the other axis. The product characteristics are elicited from stakeholders, existing systems, and the product plan. In our approach, product characteristics (we call them feature definitions) are derived from existing systems. In PuLSE-Eco the benefit analysis step decides what to develop for reuse and what not. Benefit functions describe the benefit of having a certain characteristic inside the scope. In our approach we analyze the similarity of the different products and calculate a similarity value to decide what should be inside the scope.

John [8] describes the CAVE approach. It utilizes existing documentation in order to identify communalities and variability. CAVE foresees the steps; Preparation, Analysis and Validation. In the first step a domain expert selects user documentation for analysis. In the second step a domain expert browses the selected documents and tags elements based on extraction patterns. The results of the second step are product line artifacts that are validated by a group of domain experts.

Scoping based on source code is presented in de Medeiros et al. [14]. The authors present a tool-based approach containing three modules. The feature identification module receives legacy systems source code and outputs the features composing the legacy system. The similarity comparison module identifies copied source code from legacy systems and calculates the similarity of feature implementations. The third module visualizes the results for domain experts. Duszynski et al. [15] present an approach that analyzes the source code of multiple variants for commonalities to support migration towards a product line. The reuse potential of system parts is assessed using occurrence matrices. Instead of a pair-wise comparison of existing variants, the authors propose to describe the similarity between a set of variants in a matrix. The matrix contains the different elements of the variants that are compared, the variants and the occurrence of the elements in the variants. The similarity rate is categorized as core (element occurs in all variants), shared (element occurs in some variants), and unique (element occurs in only one variant). In contrast to [14] and [15] we focus on configuration settings of standard software products rather than on source code. In the second case study we calculate similarity of products based on measures defined in these products. We compare formula definitions in this case but do not analyze the source code for similarity. Although many measures realize the same business concept (e.g. calculate the contribution margin) the implementation itself greatly differs because the information sources for this measure are different for most of the systems (e.g. different tables from different ERP systems are used to retrieve the input data for the calculation). Code scoping techniques would thus reveal that the implementations of these products are highly different.

6 Conclusion and Lessons Learned

We presented a tool-supported approach that enables companies to semi-automatically perform a similarity analysis of existing product configurations. We

consider it as a lightweight approach which complements existing approaches and supports companies to initially answer the question if a product line approach would suit their needs. The presented solution enables SMEs to estimate the similarity in standard software products. The case studies have shown that our approach can be applied to different types of products (ERP, data analysis) and also scales to a larger number of products. The similarity calculation is automated and reveals quick insight on the existing product portfolio. It can be repeated and refined without major changes on the tool.

In order to evaluate our approach we have conducted two industrial case studies in the field of business software. Starting a product line from existing solutions is recognized as an option to improve productivity and quality. In the initial case study we have shown that the tool can be used by domain experts and reveals valuable results. Furthermore the evaluation shows that domain experts give different estimates on the reuse potential that not only differ from the calculated value, but also from one another. If a company is planning to introduce software product lines they should not rely only on their domain experts estimates, but take (semi-)automatic scoping into consideration. Although the calculated results did not clearly indicate the need for introducing a product line approach, it provoked an intense discussion in the company.

In the second case study we started with a large set of existing products, built for different customers and purpose. In contrast to the first case study these products were less standardized. Therefore we intensively used the dictionary concept and defined many synonym lists to identify semantically equivalent configuration settings. This case study revealed the importance of domain knowledge when performing a scoping analysis. The dictionary concept was proven to be a valuable way to make domain knowledge explicit and reusable for automatic processing. Moreover, the second case study has shown that our approach scales also to a large number of less standardized products. Also, the more products under analysis are domain specific the more domain knowledge is required. We have learned that less standardized software products like QlikView in case study two, are harder to compare than strictly standardized software products like the ERP systems in case study one.

Future work will include extending the tools functionality to implement a more sophisticated similarity calculation method in order to identify clusters of similar parts. Moreover, we will work on the dictionary concept. The second evaluation has shown that we need to address the different granularity of dictionaries and synonym definitions in order to handle products of different granularity for analysis.

References

1. van der Linden, F., Schmid, K., Rommes, E.: Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. Springer (2007)
2. Clements, P., Northrop, L.M.: Software Product Lines:Practices and Patterns. Addison-Wesley (2007)
3. Krueger, C.: Easing the transition to software mass customization. In: 4th Int. Workshop on Software Product-Family Engineering (PFE), pp. 282–293 (2002)
4. Schmid, K.: A comprehensive product line scoping approach and its validation. In: 22nd Int. Conf. on Software Engineering (ICSE), pp. 593–603 (2002)

5. Schmid, K.: Scoping Software Product Lines: An Analysis of an Emerging Technology. In: 1st Conference on Software Product Lines (SPLC), USA, (2000)
6. Schmid, K., Schank, M.: PuLSE-BEAT – A Decision Support Tool for Scoping Product Lines. In: van der Linden, F.J. (ed.) IW-SAPF 2000. LNCS, vol. 1951, pp. 65–75. Springer, Heidelberg (2000)
7. Dalgarno, M.: The Scoping Game at miniSPA2007, http://blog.software-acumen.com/2007/07/19/the-scoping-game-at-minispa2007/ (September 2, 2013)
8. John, I.: Using Documentation for Product Line Scoping. IEEE Software 27(3), 42–47 (2010)
9. da Silva, I.F.: An Agile Approach for Software Product Lines Scoping. In: Proceedings of the 16th Int. Software Product Line Conference, Brazil (2012)
10. Leitner, A., Kreiner, C.: Software product lines – an agile success factor? In: O'Connor, R.V., Pries-Heje, J., Messnarz, R. (eds.) EuroSPI 2011. CCIS, vol. 172, pp. 203–214. Springer, Heidelberg (2011)
11. Noebauer, M., Seyff, N., Groher, I., Dhungana, D.: A Lightweight Approach for Product Line Scoping. In: 38th EuromicroConference on Software Engineering and Advanced Applications (SEAA), Turkey, pp. 105–108 (2012)
12. John, I., Knodel, J., Lehner, T., Muthig, D.: A Practical Guide to Product Line Scoping. In: 10th Int. SPL Conference (SPLC), USA, pp. 3–12 (2006)
13. DeBaud, J.M., Schmid, K.: A systematic approach to derive the scope of software product lines. In: 21st Int. Conf. on Software Eng., USA, pp. 34–43 (1999)
14. de Medeiros, T.F.L., LemosMeira, S.R., Almeida, E.S.: CodeScoping: A Source Code Based Tool to Software Product Lines Scoping. In: 38th Euromicro Conf. on Software Eng. and Advanced Applications, Turkey, pp. 101–104 (2012)
15. Duszynski, S., Knodel, J., Becker, M.: Analyzing the Source Code of Multiple Software Variants for Reuse Potential. In: 18th Working Conference on Reverse Engineering (WCRE), Ireland, pp. 303–307 (2011)