Applying Product Line Concepts in Small- and Medium-Sized Companies

Peter Knauber, Dirk Muthig, Klaus Schmid, and Tanya Widen

Fraunhofer Institute for
Experimental Software Engineering (IESE)
Sauerwiesen 6
D-67661 Kaiserslautern, Germany
+49 (0) 6301 707 251
{knauber, muthig, schmid, widen}@iese.fhg.de

ABSTRACT

Small and medium-sized enterprises (SMEs) work under heavy constraints: They need to be very flexible and fast in their reaction to customer requests, thus limiting their possibility for long-term planning.

Software product line engineering, on the other hand, usually requires this possibility for long-term planning, because investing in a product line infrastructure is investing in future reuse. Thus, so far successful product line engineering efforts have been shown mostly by large companies, often dominating their market.

In a partially publicly funded project[1] we have started to apply PuLSE™, the Product Line Software Engineering method developed at Fraunhofer IESE, in six SMEs addressing six different domains. The paper presents first experience and lessons learned from 24 months of project work including first results within the companies.

1 INTRODUCTION

New software companies usually start with a single idea — and thus with a single product. Over time, if the company is successful, the product matures and it becomes visible that the same idea, or slight variations of it may also be used to extend the single product into a set of products. However, during this process financial resources are usually still scarce in the company, thus, it is impossible to hire more developers for developing the new products. Consequently, in order to make this transition truly successful the company has to make reuse a reality in the sense that a core idea shall also be reused in the form of a single asset, a concept which is key to product line development. Changing the original development mode of the organization from an one-product-at-a-time to a product line mode forms a vital shift to the company. However, usually the

[1] The project “Software Variantbuilding” was partially funded by the Ministry of Economic Affairs, Transportation, Agriculture, and Viniculture of the state Rhineland-Palatinate, Germany.

methodological underpinnings necessary to perform such an organizational quantum leap are missing, making this phase one of the major stumbling blocks during the build-up phase of a company.

Product Line engineering is a recent concept in software engineering aiming at making large-scale reuse a commercial reality. The core idea is to synchronize and organize the development of multiple products around a single reuse infrastructure. The aim is to develop functionality which is common or similar for multiple systems explicitly for reuse and thus to share the effort among the different systems. PuLSE™ is a specific method for product line engineering that was developed at the Fraunhofer IESE [22]. Right from the start this approach was developed to be customizable to different organizations. In particular, recognizing the difficulties described above, we aimed at making it also tailorable to small organizations. Here, a major input was provided by the "Software Variantbuilding" project, which was set up in 1997. This project aimed at making product line technology available to SMEs. This project overlapped with the development of the PuLSE approach and to a large extent contributed to the customizability of the PuLSE approach as it raised the awareness of the method developers to the large range of variability that exists in companies.

The paper is structured as follows: Section 2 characterizes the "Software Variantbuilding" project, the companies participating and their domains. In section 3 the initial approach we started with is described. When PuLSE, on which an overview is given in section 4, became available, we decided to apply it instead. Section 5 summarizes our experience from this project and presents some major lessons learned.

2 CHARACTERIZATION OF THE PROJECT "SOFTWARE VARIANTBUILDING"

2.1 The Project

So far, reports about successful applications of domain engineering approaches stem mostly from large-scale projects in large enterprises ([5], [1]). However, in Germany there exist many small- to medium-sized software companies and most software development departments in larger companies are of that size too. Thus, there is a strong need to address the specific situation of these organizations. In 1997 the Fraunhofer Institute for Experimental Software Engineering (IESE) initiated the project "Software Variantbuilding" with the intention to transfer the concepts of domain or product line engineering to small and medium-sized enterprises (SMEs). Six SMEs are involved in the project, which in total lasts for two and a half year, to apply and validate respective product line methods adapted, evolved, or developed by IESE. Each company committed to contribute with at least 18 person months, IESE spends about 7 person months per company spread over the duration of the

project. From these numbers it becomes clear that we can not actually do the product line engineering but only initiate and coach the process in the companies.

The companies that should benefit from the funded project had, of course, to be located in the state of Rhineland-Palatinate. According to these restrictions six companies were selected from the original set of candidates. All of them are located within driving distance to ease communication through frequent meetings. These companies and their main business domains are characterized in the next section.

It was recommended that the domains selected for exploration in the project would be well-known by the companies; they should be typical but not critical for their business success.

2.2 The Companies

Table 1 gives an overview on the companies participating in the project "Software Variantbuilding". A short profile of each of the companies and their products is presented together with their motivation to join the project.

Company	Application domain	Target market	Number of software developers in the domain	Main motivation for product lines
A	Architecture CAD systems	standard software	4	consistent versions over time
B	Civil engineering	standard software	2	common architecture for separate products
C	Desktop publishing	standard software	6	address different user groups
D	Stock and securities exchange	standard software	8	re-integration of diverged products
E	Intelligent information systems	individual customers	11	common architecture for application kernel
F	Human comfort modeling	individual customers	5	diversification of existing product

Table 1. Overview on SMEs in the project "Software Variantbuilding"

Company A produces CAD solutions and other applications for architects. The candidate for a future product line was a product used for city planning. An older version of such a product exists already but a newer one with widely extended functionality would be developed from scratch. Variants of this product would mostly exist in versions over time: Initially only modeling of a simple landscape with simple houses would be available. Later on more complex house and roof shapes would be added and there would be functionality to draft possible variations of buildings within a given area. In the last version planned, the adherence to legal regulations would be checked automatically and there would be more high-level functionality to support sophisticated tasks like the anticipation of the population structure, based on the kind and size of buildings planned, in a given area would be available. The technology from "Software Variantbuilding" would ensure that assets developed for early versions of the product would still be usable in the later ones.

Company B develops software for statical calculations used by civil engineers and architects in the domain of steel and com-

posite (i.e., steel and concrete) construction. For projects where composite materials are used basically no software solutions exist that support the construction of whole buildings or major parts of these. So far, company B provides separate products dealing with beams, columns, and slabs. Within the project “Software Variantbuilding” it was intended to develop a software reference architecture which allows for the integration of these products while the existing applications continue to exists based on the same software assets.

Company C develops applications for desktop publishing and for the preliminary printing stage. Particular emphasis within the products is on the layout functionality which satisfies highest typographical demands. Variants are needed to support user groups with different background (from occasional users to professionals) and different scripting families: Supported are not only roman scripting families but also families like Japanese (written top-down) and Hebrew (written from right to left). These different scripting families not only require special layout but also special input facilities. Motivation for “Software Variantbuilding” was to define a domain model and a common architecture for these variants.

Company D is leading in the market of software for stock market investment and information services for non-professionals in Germany. There exist already a couple of products covering several subdomains. These products were developed in an evolutionary way, that is, they have to be maintained and evolved in parallel with the respective amount of effort. Additional to the existing products a series of new applications is planned. Given that situation it is of critical importance to re-unify the software basis. The basic technology to do this would be provided by “Software Variantbuilding”.

Company E develops technology and provides software solutions for case based reasoning (CBR). CBR applications exist already in helpdesk systems, tourist information and reservation systems, real estate searching systems, and others. Common basis for all applications of company E is the retrieval kernel. That kernel has to deal with various combinations of input data formats and quality, similarity models, and data structures for the similarity calculation depending on the amount of data to be handled. Within “Software Variantbuilding” a reference architecture for the retrieval kernel would be developed.

Among others company F provides a system for the simulation of various aspects of the human body. It provides a model of the human body and is able to predict the posture a human would adopt given certain circumstances, while taking into account human comfort feeling based on the statistical distribution of body measures correlated to gender, age, and origin. Whereas, in the past, it has been primarily used for ergonomic analysis by the automotive industry, there are plans now to use it in very different application contexts, for example, body measuring for the production of tailor-made clothes. The current structure of the system, however, makes it difficult to quickly adapt it to these new contexts. Reengineering technology was

combined with product line methods from “Software Variantbuilding” to overcome these problems (see [3]).

3 THE INITIAL APPROACH

At the start of the project, the predominant domain engineering technologies were assessed. Lucent Inc.’s Commonality Analysis process was chosen as the method to begin with in the companies [8].

The Commonality Analysis process is a centered around eliciting and documenting the requirements for a family of systems in the Commonality Analysis document — where requirements are informal textual descriptions. The requirements are hierarchically structured to group related concepts. The structuring is open and can be decided upon by the people creating the document. The requirements are categorized as either common requirements that hold for all members of the product line, or variable requirements that capture the anticipated differences among systems. Variable requirements come in three forms: optional, alternative, and range. The variable requirements are aggregated in the decision model, which is used to support single system development.

The reasons for choosing the Commonality Analysis process include the following:

- the project members had experience with the method
- the method is well documented
- the method is straightforward to learn
- the method is text-based (no special tool support required)
- there is a good relationship between IESE and developers of the method at Lucent
- Lucent reported success using the method with small groups internally

Because of the other project pressures the SMEs faced, the initial plans for this project included meetings about one day every other week with each project partner, which would consist of group discussion meetings with available experts. Additionally, tasks were to be done between meetings.

Also due to existing project pressures, training had to be limited. Therefore, we decided to try “training while doing”. This consisted of a one day introduction to the ideas of product line engineering prior to starting analysis process. At the beginning of the process we, IESE, would be responsible for writing the documentation (the domain model), up until a point when it represented a good example and could be taken over by the company’s experts. After this we would still consult on the process and results.

This initial approach evolved as we went along, mainly from our lessons learned and feedback from the development of the PuLSE process.The initial approach evolved to such a degree that now it can be said that we are applying PuLSE on the project.

However, it should also be noted that the initial lessons learned also heavily influenced the development of PuLSE. The major premise of PuLSE is that stock methods cannot be as is applied in new contexts. Therefore, PuLSE aims to provide customized processes that integrate aspects of existing methods but tailor the process to match each specific situation.

4 PULSE – PRODUCT LINE SOFTWARE ENGINEERING

PuLSE™ was developed at Fraunhofer IESE as a customizable method to support the conception, construction, usage, and evolution of software product lines, that is, the complete product line lifecycle. Figure 1 presents a decomposition of the three main PuLSE elements:

- The PuLSE deployment phases describe the activities performed to set up and use the product line. These phases are initialization of PuLSE, as well as construction, usage, and evolution of the product line infrastructure.
- The technical components comprise the second main element of PuLSE. They provide the technical know-how needed to implement a product line. During PuLSE initialization the components are customized according to the specific environment where PuLSE is applied. Then they are applied throughout the other deployment phases.
- The third main element of PuLSE, the support components, address pre-customizations of PuLSE and organizational aspects of the environment, and allow to evaluate the quality of a PuLSE application in a specific environment. They are used by the other elements.

The following subsections describe the three technical components applied in the product line infrastructure construction phase of PuLSE in more details. Mainly these three components are applied in “Software Variantbuilding” so far. The components are:

- PuLSE-Eco for product line scoping: how to effectively scope the infrastructure focussing on concrete product definitions
- PuLSE-CDA for product line modeling: how to model the product characteristics found within the scope of the product line and explicitly denote the product family members
- PuLSE-DSSA for product line architecting: how to develop the reference architecture while maintaining the traceability to the model

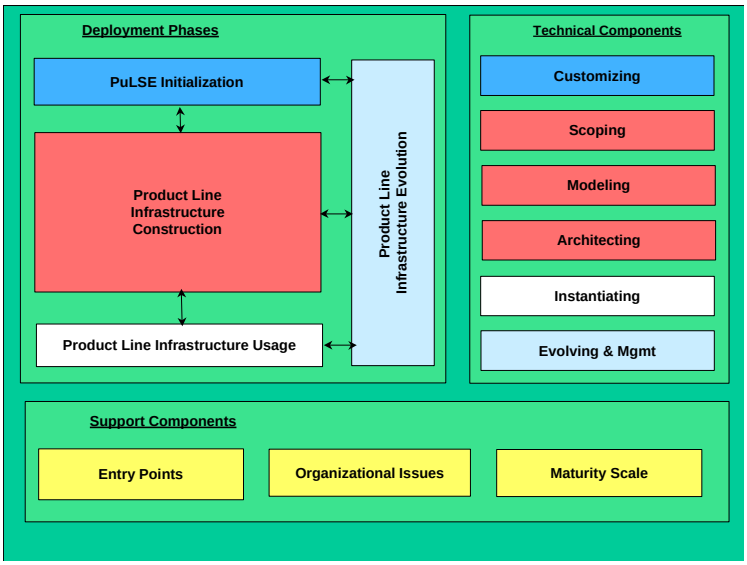

Figure 1. PuLSE Overview

Section 5 presents experience from the application of these components in “Software Variantbuilding”.

4.1 PuLSE-Eco

A major distinction between software product lines and other reuse approaches is that the product line is based on a clear vision of which future products will be developed. Based on this an analysis can be performed on which features of the envisioned products shall be supported in a reusable manner based on the goals of the organization.

PuLSE-Eco aims at performing exactly this activity [7]. In particular, PuLSE-Eco aims at determining the specific opportunities for reuse that exist in the product line. As a result of this activity the scope of the reuse infrastructure is determined, that is, a description of what functionality should be supported in a general manner for the whole product line and what functionality should be regarded as specific to a single system.

The basic approach followed by PuLSE-Eco is to develop in a systematic manner an overview of the various functionality relevant to the products in the product line and to determine which functionality shall exist in which products; this description is then called a product map, due to the tabular notation used for representing this information.

Then, based on the goals of the organization for embarking on product line approach, evaluation functions are defined that describe to which extent having a specific functionality in the scope contributes towards these goals. Finally, the values for the evaluation functions are determined. This is usually done by interviewing stakeholders in the specific company. This leads to a rating of the features with respect to the benefit of including them in the scope.

4.2 PuLSE-CDA

When the set of products that are determined as member of the product line has been defined by PuLSE-Eco, the requirements for all these systems are documented within a single model: the domain, or product line, model. The purpose of PuLSE-CDA is to develop this requirements document, as well as to enable the instantiation of the product line model for single systems. The domain analysis process and the products created by PuLSE-CDA are customizable to the context of its application.

A product line model consists of multiple workproducts that capture different views of a domain. Each view focuses on particular information types and relations among them. In the workproducts, common and distinguishing aspects of the products, commonalities and variabilities, are modelled. Thereby, commonalities are requirements that hold for all members of a product family. Variabilities are requirements that may differ between members of the product family. They are connected to decisions that, when resolved, specify (via instantiating the product line model) a particular system, a member of the product line. The decisions are at different levels of abstraction and are hierarchically structured based on constraints among them. The decision structure is the decision model for the product line model, which guides the deployment of the product line model and, therefore, plays a major role during application engineering [4].

4.3 PuLSE-DSSA

At the core of a product line infrastructure is the reference (or domain-specific) architecture. This architecture is common for all the systems in the product line, ensuring their conceptual integrity. Developing systems based on instances of the reference architecture implies a high potential for reuse and related benefits like increased quality, cost reduction, decreased time-to-market, etc. However, due to the required degree of flexibility, product line architectures are even more difficult to conceptualize than those for individual systems. PuLSE-DSSA (domain-specific software architecture) is a method for the development of product line reference architectures [6].

The basic idea of PuLSE-DSSA is to incrementally develop a reference architecture guided by generic scenarios that are applied in decreasing order of architectural significance. At first, generic scenarios are developed using information from the product line model developed by PuLSE-CDA (see section 4.2). These scenarios are then sorted according to their architectural significance. A basic set of them is used to build an initial architecture. After that, the remaining scenarios are applied one by one to the current architecture candidate to refine or extend it. This leads to new candidates that are analyzed and ranked based on functional coverage and coverage of property-related scenarios. The best one(s) serve(s) as input for the next iteration step. This iteration stops after all generic scenarios have been applied.

5 LESSONS LEARNED IN APPLYING PULSE IN SMES

We learned from this project on different levels. First, there are issues regarding technology transfer to SMEs in general. Then there are factors that have great influence on the introduction of product line engineering into SMEs. And last, there are issues concerning specific components of the PuLSE technology selected in this project. Some of the most important lessons we learned in this project are discussed in the following subsections.

5.1 Lessons Learned in Technology Transfer to SMEs

One major factor to consider we found in SMEs is the influence of few key individuals. They may be the founder of the company or have very strong skills that enable them to play a key role there. We found it absolutely necessary to convince these persons of the technology we wanted to introduce. If they can be convinced they would push the other employees to adopt it, if they do not like it, the project would be very likely to fail because they are important enough to block the transfer process.

Another factor typical for all companies in the project is the absence of an explicitly defined development process. From this fact arise two implications: First, the employees are not used to follow a process, meaning, the new technology can not be communicated and introduced formally only; it has to be explained in detail using examples. Second, it is hard to change a process or introduce additional tasks when there is no reference process to relate to. Either an extensive baselining takes place to determine current development practise within the company or very detailed training material customized to the specific environment is needed to illustrate the new working mode. And even when the opportunity of the transfer project is used to introduce a more formal process in a company it is not unlikely that the process definition will be ignored as soon as there is some project pressure.

5.2 Lessons Learned in Introducing Product Line Engineering

Typical for SMEs is their close cooperation with customers. This of course offers them a marketing advantage over larger competitors because they not only know early about actual and potential needs of their customers but they are also able to react more flexible to these than larger organizations. With respect to product lines, this factor proves to be a disadvantage. Basic requirements for successful product line development are a clear vision about the future evolution of the company's applications and some stability of the domain. If these requirements are not fulfilled, the investment in setting up a reusable infrastructure for the product line may not pay. Successful product line development involves some change in behavior: Reacting flexibly to customers' needs gains less benefits than actively creating and/or steering these needs (up to a certain level) in a direction where they would be supported by the own product line. Especially PuLSE-Eco proved to be a good

means to explicitly model business goals and to identify possible effects when these goals are changing.

5.3 Lessons Learned In Scoping

The first observation with SMEs from a planning point of view is that they usually have no precise vision of the future products they will develop. While it is basically a problem to applying the approach in the first place, once the organization has accepted it, the approach actually helps to shape their vision and enables them to find new opportunities.

There are some characteristics of the PuLSE-Eco approach, that make it actually adaptable to a wide range of situations, in particular, in small- and medium-sized companies:

- The approach need not be applied to the complete product line, but can just as well be restricted to a sub-domain, which promises the highest pay-off. This was actually done in “Software Variantbuilding”.
- The level of the analysis (i.e., the granularity of features) is adaptable, which allows to adapt to the specific needs that exist.
- The goals are based on the specific organization. Actually, in case of company D, one goal was clearly dominating to an unusual extent. Thus, an approach where the weighting of the goals would have been built in, would fail to arrive at the right conclusions.

A major problem, which can actually be expected in all small companies is the inavailability of sufficient data (e.g., on effort, errors, etc.). To some extent this could be alleviated by the fact that the evaluation functions are specifically derived for the organization based on their goals. Consequently, the functions could be defined in such a way that the stakeholders could provide appropriate data. Again, the flexibility built into the approach allowed to adapt it to the specific environment.

Another lesson for applying PuLSE-Eco stems from the problem that small organizations are usually extremely short on project resources. Consequently, they will not devote their time to a consulting project unless they are pushed to do so. Here, it proved very helpful that the approach — although it delivers a rather detailed analysis — requires only little effort on the side of the company. Further, the fact that the use of low-tech tools that are everywhere available (in our case Excel) was instrumental in enabling the company to perform the data acquisition off-line.

5.4 Lessons Learned In Product Line Modeling

The things we learned cover both general topics in product line modeling and topics specific for SMEs. Here we summarize the main lessons learned.

Because of the nature of the project (non-critical domain), and the characteristics of SMEs (project pressure with few people

to move around) we encountered some problems with resources and management expectations within the companies. From this we learned that the company management must set aside and enforce time for the project members to work on the product line project, and they must show that they support the project. Otherwise working on the product line project becomes a spare time activity, with the normal project demands taking over the domain experts time. We experienced that the domain experts were only working on the project when we had meetings and these were hard to schedule. Therefore, our initial work plan of having meetings one day every other week was not producing results very quickly. We found that changing to larger blocks of meetings (e.g. three days in a row) produced more results even if these meetings occurred less frequently.

The companies had no systematic development processes in place, they are mostly focused on producing products (code). Therefore, there was a need to introduce both systematic development and product line engineering at once. This a large change in the method of working and there is a need to minimize the impacts felt by this change and provide much guidance on the new way of working.

We found that group meetings were good for sharing and checking knowledge among the domain experts. Some things that we encountered were that the size of the meetings is an issue: having discussions with more than five domain experts could be a problem. We found that the meetings really needed to be focused, with concrete goals (in terms of what domain topics will be covered). Otherwise there was very little preparation and the meetings could cover all types of topics. Another common problem with unfocused meetings was that different interest groups (management vs. developers, or academic vs. industry) would try to push the topics to cover their interests. Of course this problem cannot only be solved by focused meetings. We sometimes found it beneficial to act as impartial moderators to facilitate the alignment of these interest groups.

One disadvantage of meetings is that they are a tremendous time/resource consumer. Therefore, not everything should be discussed in meetings. We found that much modeling can be done off-line between meetings and then reviewed with the group. Also, issues that arose which needed more input should be captured and assessed off-line.

Our training approach, which consisted of a one day training session and then training while doing (where initially we documented to give them examples of how to do it), did enable the domain experts to learn the approach and produce results at the same time. In general, all small companies liked to talk about their domain but they all had difficulties to write down results at the required abstraction level. We found that providing the starting examples in their own domain helped them learn how to abstract and document. We also found that the experts all feel that the different view point on their domain added by the

external domain analyst, which is our role in the project, improves their understanding of their systems and their domain.

However there are some drawbacks. One thing is that we were responsible for documenting the models, so that we could produce good examples based on discussions with experts. This was a problem because we had to learn a lot, as we are not domain experts. Also, we could not model anything that was not discussed in meetings. Therefore, it is important to get example of the documents early and then transfer the responsibility for modeling to the domain experts as soon as possible. It is important to move the responsibility for these documents as soon as possible to the small company so that they own these documents, get used to maintain them, and finally to use them.

Our experience with the Commonality Analysis process was that it is ok as a starting point, but that we needed more focus and different models. We found that there are different concerns in each product line and that there may be special models that are most appropriate to capture this information. Our conclusion is that one can start with standard models, but should be on the lookout for others that capture important information. PuLSE advocated customizing prior to modeling, however, due to the limited resources of SME's, the appropriate set of models cannot be determined all at once before the modeling starts. We adapted the models as we learned about the different concepts that needed to be captured. Our goal is to determine categories of models that can be used as a starting point. For example, there are many types of information models, and behavioral models that are typically used when capturing requirements.

Tool support is necessary to some degree. First of all we saw that the companies really wanted tools. Second, because of the amount of information and the many relationships and dependencies tool support helps with intellectual control of the information. When then method has to be customized as you go, then you need tool support that can adapt to this.

5.5 Lessons Learned In Architecting

A general problem we faced was the lack of explicit documentation. Typically there exists little or even no documentation on the architectural level or the documents are not current. In companies D and F it was necessary to do some architecture recovery first in order to get a better impression of the existing systems.

The factor most influencing the development of a reference architecture, that is, an architecture common for all systems of a product line, is the architecture of existing systems. Due to the size of SMEs, there are usually only few architects available for the development of a reference architecture. In the companies of "Software Variantbuilding" these architects are also responsible for the architecture of the existing system(s). Of course, they tend to resists against changes of their architecture for two reasons: First, they simply want to avoid too much (re-) work when revising the existing software according to the

new reference architecture. Second, they hesitate to admit that there could be better solutions than they developed on their own. We found it hard to overcome these problems; the only way to achieve cooperation is to motivate them in a way they come up with their own suggestions for the architecture and to just guide the process.

6 CONCLUSIONS

In the “Software Variantbuilding” project we started to transfer product line concepts to small- and medium-sized enterprises. As presented in the paper, limited resources in terms of personal and money are main influencing factors. In order to be successful in that environment an approach has to be product-oriented and being able to show early results, for example, support modeling, architecting, and implementing of a subdomain instead of a complete domain. Also, the process has to be optimally suited to the environment, because the companies can not afford the resources for extra overhead.

Since SMEs typically have no measures characterizing their development process, the only way to keep them going with the process is to motivate the people involved in the project, especially the main stakeholders and to give them positive results. The following typical comments may illustrate that we achieved this with the application of PuLSE in our project:

- Manager, company B: “… I expect a more efficient and goal-oriented way to develop new software components. From experience so far we expect positive results from the project.”
- Manager Quality Assurance, company A: “… We consider the method as time-consuming but detailed. It helps eliminating misunderstandings. … The employees involved in the project like and support it. …”
- Developers, company C: “… We consider it positive to address things that have been neglected in the past. … Everybody now gets detailed insight into the product. … We would like to speed up the process, for example, having meetings more often.”

Based on our experience we changed our approach (from Commonality Analysis to PuLSE) and our mode of working (have fewer but longer meetings). Since this transition we are now in a situation where we expect to find less problems and more success in the future.

REFERENCES

1. Tim Bardo, Dave Elliot, Tony Krysak, Mike Morgan, Rebecca Shuey, Will Tracz. CORE: A Product Line Success Story. *Crosstalk: The Journal of Defense Software Engineering*, Vol. 9, No. 3, 1996
2. A. Bayer, O. Flege, P. Knauber, R. Laqua, D. Muthig, K. Schmid, T. Widen, and J.-M. DeBaud. PuLSE: A methodology to develop software product lines. In *Symposium on Software Reusability ’99 (SSR’99)*, pages 122 –

131, May 1999.

3. J. Bayer, J.-F. Girard, M. Würthner, J.-M. DeBaud, M. Apel, Transitioning Legacy Assets to a Product Line Architecture, In *Proceedings of 7th European Software Engineering Conference (ESEC) / 7th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE)*, 1999
4. J. Bayer, D. Muthig, and T. Widen. Customizable Domain Analysis. In *Proceedings of the First International Symposium on Generative and Component-Based Software Engineering (GCSE)*, Erfurt, Germany 1999
5. L. Brownsword, P. Clements. A Case Study in Successful Product Line Development. *Technical Report CMU/SEI-96-TR-016*, Carnegie Mellon, 1996
6. Jean-Marc DeBaud, Oliver Flege, Peter Knauber. A Method for the Development of Software Reference Architectures. *Proceeding of the 3nd International Software Architecture Workshop (ISAW-3)*, Orlando, FL, 1998
7. J.-M. DeBaud and K. Schmid. A systematic approach to derive the scope of software product lines. In *Proceedings of the 21st International Conference on Software Engineering*, 1999.
8. David Weiss. Defining Families: The Commonality Analysis. Submitted to IEEE Transactions on Software Engineering