Applying Product Line Concepts in Small and Medium-Sized Companies

Because software product line engineering requires long-term planning, the companies that have used it successfully are large ones that can afford to take the long view. But smaller enterprises must be flexible and fast in reacting to customer requests. The authors have results and lessons learned to report on applying the Pulse product line method to six small to medium-sized companies.

Peter Knauber, Dirk Muthig, Klaus Schmid, and Tanya Widen
Fraunhofer Institute for Experimental Software Engineering

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 management sees that it can use the same idea (or slight variations of it) to develop a set of products. However, during this process most companies still have scarce financial resources, so they cannot hire more developers. They must make reuse a reality—reusing a core idea in the form of a single asset, a concept which is key to product line development. Changing the organization's original development mode from a one-product-at-a-time to a product line approach entails a fundamental shift. Unfortunately, the methodological underpinnings necessary to do this are often missing, making this phase a major stumbling block during a company's build-up phase.

Product line engineering aims to make large-scale reuse a commercial reality—to develop common or similar functionality explicitly for reuse in multiple systems and thus to distribute cost and effort. The core idea is to synchronize and organize the development of multiple products around a single reuse infrastructure. Pulse is a specific method for product line software engineering developed at the Fraunhofer Institute for Experimental Software Engineering.[1] From the start, the IESE designed this approach to be customizable for different kinds of organizations. Recognizing the special difficulties that small and medium-sized organizations face, we specifically wanted to make Pulse work for them. Here, the IESE's Software Variant-Building Project provided a major input. It overlapped with the development of the Pulse approach and to a large extent contributed to its customizability, because it raised the awareness of Pulse's developers to the large range of variability in companies.

The Software Variant-Building Project

So far, reports about successful applications of domain engineering approaches stem mostly from large-scale projects in large enterprises.[2,3] However, in Germany

0740-7459/00/$10.00 © 2000 IEEE

many software companies are small or medium-sized, as are most software development departments in larger companies. In 1997, the IESE initiated the Software Variant-Building Project to transfer product line engineering concepts to small and medium-sized enterprises (SMEs). The project lasted for two and a half years, applying and validating various product line methods that the IESE had adapted, evolved, or developed. Each company committed to contributing at least 18 person-months; the IESE spent about seven person-months per company spread over the project's duration. Clearly, we could not actually perform product line engineering but only initiate and coach the process in the participating companies.

It was recommended that the domains selected for exploration should be typical and well-understood but not critical to business success. The companies involved also had to be located in the state of Rhineland-Palatinate. From the original set of candidates, we selected six companies, all within driving distance of the IESE to ease communication through frequent meetings. Table 1 gives an overview of the participants, their products, and their motivations for joining the project.

Company A produces CAD solutions and other applications for architects. Its product line candidate was a product used for city planning. The company wanted to develop a new version from scratch with widely extended functionality and then release different versions over time. Modeling of a simple landscape with simple houses would be available initially; more complex house and roof shapes would be added later on, along with the functionality to draft variations of buildings within a given area. The last version planned would automatically check adherence to legal regulations and would support sophisticated tasks such as anticipating the population structure based on the kind and size of buildings planned in a given area. The technology from Software Variant-Building would ensure that assets developed for early versions of the product would still be usable in the later ones.

Company B develops software for statistical calculations used by civil engineers and architects in the domain of steel and composite (that is, steel and concrete) construction. The company has separate software products dealing with beams, columns, and slabs, but no current software solution to support the construction of whole buildings (or major parts of buildings) made of composite materials. The company wanted the Software Variant-Building Project to develop a software reference architecture enabling the integration of these products (the existing separate products would continue to exist).

Company C develops applications for desktop publishing and for the preliminary printing stage. Particular emphasis is on layout functionalities that satisfy the highest typographical demands. User groups with different backgrounds (from occasional users to professionals) need variants that support different scripting families, including roman, Japanese (written top down), and Hebrew (written from right to left). These families require not only special layouts but also special input facilities. The company wanted the Software Variant-Building Project to define a domain model and a common architecture for these variants.

Company D leads the German software market for stock market investment and information services for nonprofessionals. A couple of products covering several subdomains already exist. The company also plans

The core idea is to synchronize and organize development of multiple products around a single reuse infrastructure.

Table 1

The Companies Participating in the Software Variant-Building Project

Company	Application domain	Target market	No. of developers	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	To address different user groups
D	Stock and securities exchange	Standard software	8	Reintegration 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

The major premise of Pulse is that stock methods cannot be applied as is to new contexts.

a series of new applications, so it is critically important to reunify the software base. The company wanted the Software Variant-Building Project to provide the basic technology to do this.

Company E develops technology and provides software solutions for case-based reasoning. CBR applications exist in helpdesk systems, tourist information and reservation systems, real-estate searching systems, and others. All of company E's applications are based on a retrieval kernel, which calculates similarities by manipulating various combinations of input data formats and quality, similarity models, and data structures. The company wanted Software Variant-Building to develop a reference architecture for the retrieval kernel.

Company F provides a system for simulating various aspects of the human body. Based on its model of the human body, the system can predict the posture a person would adopt given certain circumstances, while taking into account human comfort based on the statistical distribution of body measures correlated to gender, age, and origin. In the past, the system was primarily used for ergonomic analysis by the automotive industry; the company now wants to use it in very different application contexts—for example, to measure the body for producing tailor-made clothes. The current structure of the system, however, is difficult to adapt quickly to these new contexts. To overcome this problem, the company wanted to combine reengineering technology with product line methods from Software Variant-Building.[4]

The Initial Approach

At the start of the Pulse project, we assessed the predominant domain engineering technologies and chose Lucent's Commonality Analysis process as the method to begin with.[5] This process centers around eliciting and documenting the requirements for a family of systems in a so-called Commonality Analysis document. Requirements are informal textual descriptions, hierarchically structured to group-related concepts. The structuring is open, and the people creating the document can decide what structure to use. 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. They are aggregated in the decision model, which is used to support single-system development.

We chose the Commonality Analysis process because

- 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),
- the IESE has a good relationship with the method's developers at Lucent, and
- Lucent reported success using the method internally with small groups.

Because of other project pressures the participating six SMEs faced, we decided to meet with each project partner about one day every other week, holding group discussions with available experts. Other tasks were to be done between meetings.

Also due to existing project pressures, training had to be limited. We decided to try "training while doing," starting with a one-day introduction to the ideas of product line engineering. At the beginning of the process, we (the IESE) would be responsible for writing the documentation (the domain model) until 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 because of lessons learned and feedback from the development of the Pulse process—so much so that now we can say we are applying Pulse on the project.

However, the initial lessons learned also heavily influenced the development of Pulse. The major premise of Pulse is that stock methods cannot be applied as is to new contexts. Therefore, Pulse aims to provide customized processes that integrate aspects of existing methods but are tailored to match each specific situation.

Pulse

As we said earlier, the IESE developed Pulse as a customizable method to support the conception, construction, usage, and evolution of software product lines—that is, the complete product line life cycle. Figure 1 presents a decomposition of the three main Pulse elements:

1. The deployment phases represent the logical stages of a product line: Pulse is initialized to the particular organization, and the product line infrastructure is constructed and finally used. While the product line and the organization evolve over time, the initialization and construction phases must be frequently revisited.
2. The technical components provide the technical know-how to perform the activities required in each deployment phase. In Figure 1, the shade of a technical component corresponds to the shade of the deployment phase it supports. Pulse-BC describes how Pulse is initialized, that is, how the remaining technical components are customized according to the specific environment. Pulse-Eco, -CDA, and -DSSA support construction, Pulse-I supports usage, and Pulse-EM supports the evolution of the product line infrastructure.
3. The support components address precustomizations of Pulse and organizational aspects of the environment and enable us to evaluate the quality of a Pulse application in a specific environment.

The product line infrastructure construction phase, applied in the Software Variant-Building Project so far, makes use of three technical components:

- Pulse-Eco for product line scoping—lets us effectively scope the infrastructure by focusing on concrete product definitions;
- Pulse-CDA for product line modeling—lets us model the product characteristics found within the product line's scope and explicitly denote the family members; and
- Pulse-DSSA for product line architecting—lets us develop the reference architecture while maintaining traceability to the model.

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, one can analyze which features of the envisioned products will be supported in a reusable way based on the organization's goals.

Pulse-Eco does exactly this, by determining the specific opportunities for reuse that exist in the product line.[6] It establishes the scope of the reuse infrastructure—that is, it describes 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.

Pulse-Eco's basic approach is to systematically develop an overview of the various functionalities relevant to the product line's products and to determine which functionality shall exist in which products. This description is called a product map, due to the tabular notation used.

Then, based on the organization's goals for embarking on the product line approach, we define evaluation functions that describe to what extent having a specific functionality in the scope contributes toward these goals. Finally, we determine the values for the evaluation functions, usually 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.

Pulse-CDA

After Pulse-Eco has defined the set of products that belong to the product line, we must document the requirements for all these systems within a single model: the domain, or product line, model. Pulse-CDA helps us develop this requirements docu-

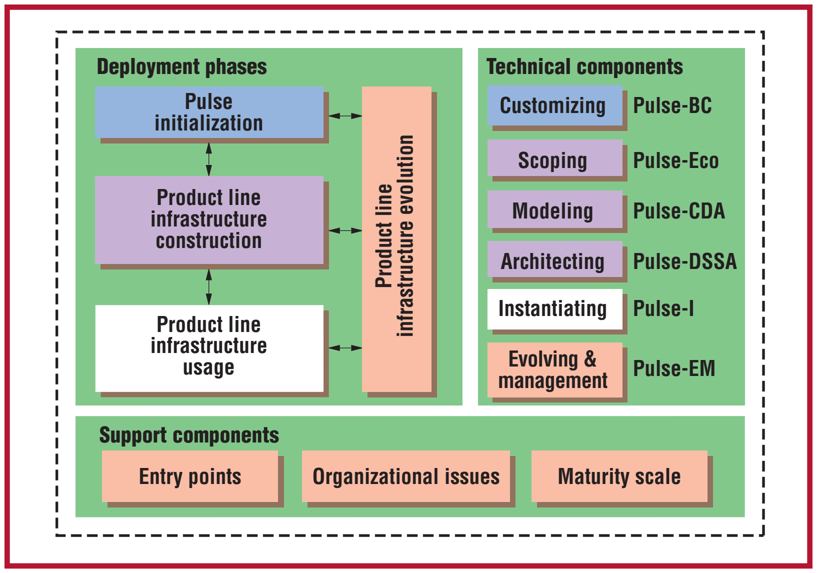


Figure 1. Pulse overview. Colors indicate which technical and support components describe how to perform a particular deployment phase.

If the employees are not used to following a process, you cannot just introduce new technology formally; it must be explained in detail using examples.

ment and enables the instantiation of the product line model for individual systems. The domain analysis process and the products created by Pulse-CDA are customizable to the application context.

A product line model consists of multiple work products that capture different views of a domain. Each view focuses on particular information types and relations among them. In these work products, we model common and distinguishing aspects of the products. Thus, commonalities are requirements that hold for all product family members, and variabilities are requirements that might differ between members. The variabilities 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; it guides the deployment of the product line model and therefore plays a major role in application engineering.[7]

Pulse-DSSA

At the core of a product line infrastructure is the reference (or domain-specific) architecture. This architecture is common to 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 such as increased quality, cost reduction, decreased time-to-market, and so on. 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 developing product line reference architectures.[8]

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. First, we develop generic scenarios using information from the product line model developed in Pulse-CDA. We then sort these scenarios according to their architectural significance and then use a basic set of them to build an initial architecture. Next, we apply the remaining scenarios 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 we have applied all generic scenarios.

Lessons Learned

We learned from this project on different levels. Some issues deal with technology transfer to SMEs, others deal with the introduction of product line engineering into SMEs, and still others concern specific components of the Pulse technology selected in this project. The following discussion covers the most important lessons we learned in the project.

Technology Transfer

We found that one major factor to consider is the influence of a few key individuals—perhaps the company founder, or those who play key roles because of their very strong skills. We found it absolutely necessary to convince these people of the value of the technology we wanted to introduce, because they then pushed other employees to adopt it. Otherwise, the project would likely fail because they were important enough to block the transfer process.

Another factor typical for the six companies prior to our project was the absence of an explicitly defined development process. Two implications arise: First, if the employees are not used to following a process, you cannot just introduce new technology formally; it must 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 must take place to determine current development practice 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, the process definition will probably be ignored as soon as pressure builds in the project.

Introducing Product Line Engineering

Typical for SMEs is their close cooperation with customers. This offers them a marketing advantage over larger competitors because they not only know early about their customers' actual and potential needs but they are also able to react more flexibly to these needs. With respect to product lines, however, 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 domain stability. If these requirements are not fulfilled, the investment in setting up a reusable infrastructure for the product line might not pay. Successful product line development involves some change in behavior: Reacting flexibly to customers' needs gains fewer benefits than actively steering these needs in a direction where they would be supported by their own product line. Pulse-Eco proved to be a particularly good means to explicitly model business goals and to identify possible effects when the goals change.

Scoping

SMEs usually have no precise vision of the future products they will develop. Although it is difficult to apply Pulse-Eco in the first place, once an organization has accepted it, the approach actually helps to shape the company's vision and enables it to find new opportunities.

Some characteristics of the Pulse-Eco approach make it adaptable to a wide range of situations, in particular to SMEs:

- The approach need not be applied to the complete product line; it can be restricted to the subdomain that promises the highest pay-off.
- The level of analysis (granularity of features) is adaptable, so we can adapt to the specific needs that exist.
- The goals are based on the specific organization. For instance, in the case of company D, an approach that built in goal weighting would fail to arrive at the right conclusions, because one goal was clearly dominating to an unusual extent.

A major problem to be expected in all small companies is the lack of sufficient data (for example, on effort or errors). To some extent this is alleviated by the fact that the evaluation functions are derived specifically for the organization based on its goals. Consequently, the functions can be defined in such a way that the stakeholders are able to provide appropriate data. Again, the flexibility built into the approach enables it to be adapted to specific environments.

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 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 little effort on the part of the company. Furthermore, the use of universally available low-tech tools (in our case Excel) let the company perform data acquisition offline.

Product Line Modeling

What we learned about product line modeling covers both general topics and topics specific to SMEs.

Because of the noncritical nature of the projects and the characteristics of SMEs (project pressure with few people to move around), we encountered problems with resources and management expectations within the companies. From this we learned that management must set aside and enforce time for the developers to work on the product line project, and they must show their support. Otherwise, working on the product line project becomes a spare-time activity, with normal project demands taking over the domain experts' time. We found that the domain experts worked on the project only when we had meetings, and these were hard to schedule. Therefore, our initial plan of having meetings one day every other week was not producing results very quickly. Changing to larger blocks of meetings (for example, 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 mostly focused on producing products (code). Therefore, we had to introduce both systematic development and product line engineering at once. This a large change in work habits, so we had to minimize the impacts felt by this change and provide a lot of guidance on this new way of working.

Group meetings among the domain experts were good for sharing and checking

Management must set aside and enforce time for the developers to work on the product line project, and they must show their support.

Our training approach enabled the domain experts to simultaneously learn the approach and produce results.

knowledge. But meeting size was an issue: having discussions with more than five experts could be a problem. The meetings also needed to be focused, with concrete goals (in terms of what domain topics would be covered). Otherwise, there was very little preparation and the discussion might span a variety of topics. Different interest groups (management vs. developers, or academics vs. industry participants) would try to push the topics covering their interests. Of course, focused meetings alone cannot solve this problem. 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 consumer of time and resources. Therefore, not everything should be discussed in meetings. We found that much modeling can be done offline between meetings and then reviewed with the group. Also, issues that need more input can be captured and assessed offline.

Our training approach—a one-day training session and then training while doing (along with examples of how to do it)—enabled the domain experts to simultaneously learn the approach and produce results. In general, the companies liked to talk about their domain, but they all had trouble writing down results at the required abstraction level. Providing examples in their own domain helped them learn how to abstract and document. The experts all felt that the different viewpoint offered by the external domain analyst—which was our role in the project—improved their understanding of their systems and domain.

However, there are some drawbacks. First, 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 transfer modeling responsibility to the domain experts as soon as possible. The company should take responsibility for the sample documents as soon as possible so that it owns them, gets used to maintaining them, and finally uses them.

We found that the Commonality Analysis process is okay as a starting point but that we needed more focus and different models. Each product line raises different concerns, and special models might exist that are most appropriate to capture this information. You can start with standard models, but be on the lookout for others that capture important information. Although Pulse advocates customizing prior to modeling, you cannot determine the appropriate set of models before modeling starts due to the limited resources of SMEs. 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, behavioral models can typically be used in requirements capture.

Tool support is necessary to some degree. First, 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 the method must be customized as you go, you need tool support that can adapt to this.

Architecting

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

The factor most influencing the development of a reference architecture—that is, an architecture common to all systems of a product line—is the architecture of existing systems. Due to the size of SMEs, usually only a few architects are available to develop the reference architecture. In the companies in the Software Variant-Building Project (the same six we chose to work with Pulse), these architects are also responsible for the architecture of the existing systems. Of course, they tend to resist changes to their architectures for two reasons: first, they simply want to avoid as much rework as possible, and second, they hesitate to admit that there could be better solutions than the ones they developed on their own. We found it hard to overcome these problems; the only way to achieve cooperation is to motivate them to come up with their own suggestions for the architecture and to just guide the process.

To successfully transfer product line concepts to small and medium-sized enterprises, you must have a product-oriented approach and show early results—for example, the approach should support modeling, architecting, and the implementation of a subdomain instead of a complete domain. Also, the process must be optimally suited to the environment, because the companies cannot afford extra overhead.

Since SMEs typically have no measures characterizing their development processes, the only way to keep product line engineering going is to motivate the participants, especially the main stakeholders, and you do that by giving them positive results. The following typical comments illustrate what we achieved with the application of Pulse:

- 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 of quality assurance, company A: "We consider the method as time-consuming but detailed. It helps eliminat[e] 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 experiences, we changed our approach (from Commonality Analysis to Pulse) and our mode of working (holding fewer but longer meetings). Since this transition, we expect to find fewer problems and more success in the future.

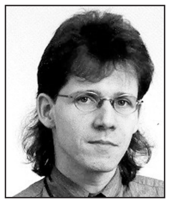
Acknowledgment

The Software Variant-Building Project was partially funded by the Ministry of Economic Affairs, Transportation, Agriculture, and Viniculture of Germany's Rhineland-Palatinate state.

References

1. J. Bayer et al., "Pulse: A Methodology to Develop Software Product Lines," *Symp. Software Reusability '99* (SSR'99), ACM Press, New York, 1999, pp. 122–131.
2. T. Bardo et al., "CORE: A Product Line Success Story," *CrossTalk: J. Defense Software Eng.*, Vol. 9, No. 3, 1996.
3. L. Brownsword and P. Clements, *A Case Study in Successful Product Line Development*, Tech. Report CMU/SEI-96-TR-016, Carnegie Mellon Univ., Pittsburgh, Pa., 1996.
4. J. Bayer et al., "Transitioning Legacy Assets to a Product Line Architecture," *Proc. Seventh European Software Eng. Conf./Seventh ACM SigSoft Symp. Foundations of Software Eng.*, Springer Verlag, Berlin, 1999, pp. 446–463.
5. D. Weiss and C. Lai, *Software Product-Line Engineering: A Family-Based Software Development Process*, Addison-Wesley, Reading, Mass., 1999.
6. J.-M. DeBaud and K. Schmid, "A Systematic Approach to Derive the Scope of Software Product Lines," *Proc. 21st Int'l Conf. Software Eng.*, ACM Press, New York, 1999.
7. J. Bayer, D. Muthig, and T. Widen, "Customizable Domain Analysis," *Proc. First Int'l Symp. Generative and Component-Based Software Eng.*, to be published.
8. J. Bayer, O. Flege, and C. Gacek, "Creating Product Line Architectures," *Proc. Third Int'l Workshop Software Architectures for Product Families* (IWSAPF-3), 2000, pp. 197–203.

About the Authors



Peter Knauber is department head for Software Product Lines at Fraunhofer Institute for Experimental Software Engineering (IESE), Germany, where he is responsible for several projects dealing with software architectures and product lines. He received a Diploma in computer science and a PhD in natural sciences from the University of Kaiserslautern. Contact him at the Fraunhofer Institute for Experimental Software Engineering (IESE), Sauerwiesen 6, D-67661 Kaiserslautern, Germany; knauber@iese.fhg.de.

Dirk Muthig is an applied researcher at the Fraunhofer IESE. He has been involved in several projects transferring product line engineering concepts into industrial environments. He helped develop the customizable Pulse method as well as the Kobra method, which integrates component-based software development with product line engineering. Muthig is a PhD candidate focusing on the institutionalization of product line engineering. He received an MS in computer science from the University of Kaiserslautern. Contact him at the Fraunhofer Institute for Experimental Software Engineering (IESE), Sauerwiesen 6, D-67661 Kaiserslautern, Germany; muthig@iese.fhg.de.

Klaus Schmid is an applied researcher at the Fraunhofer IESE, where he has been involved in several projects transferring product line engineering concepts into industrial environments and in the development of the Pulse method. He is a PhD candidate focusing on economical scoping of product line infrastructures. Schmid received an MS in computer science from the University of Kaiserslautern. Contact him at the Fraunhofer Institute for Experimental Software Engineering (IESE), Sauerwiesen 6, D-67661 Kaiserslautern, Germany; schmid@iese.fhg.de.

Tanya Widen is a software engineer at Market Maker Software AG. Prior to this, she was a member of the Software Product Lines department at the Fraunhofer IESE, where she helped develop the Pulse method for product line engineering. Through a large European project, she still collaborates with IESE to transfer product line engineering methods to Market Maker. She started studying domain engineering while working on her masters degree at Oregon Graduate Institute. Contact her at Market Maker Software AG, Karl Marx Straße 13, D-67655 Kaiserslautern, Germany; t.widen@market-maker.de.