

Exploring Communities of Practice for Product Family Engineering

Tor Erlend Fægri¹, Björn Decker², Torgeir Dingsøy¹, Letizia Jaccheri³, Patricia Lago⁴, Dirk Muthig², Hans van Vliet⁴

¹SINTEF Information and Communication Technology, Trondheim, Norway

²Fraunhofer Institute for Experimental Software Engineering,
Kaiserslautern, Germany

³Dept. of Computer and Information Science, Norwegian University of Science and
Technology, Trondheim, Norway

⁴Vrije Universiteit, Amsterdam, The Netherlands

Abstract. Product Family Engineering (PFE) is an approach to software engineering that seeks to reduce the global effort in producing multiple software products by actively promoting and governing the reuse of assets between the family members. However, PFE is highly demanding, putting stringent demands on careful planning and management in the organization. To leverage the full opportunities offered by PFE, its introduction and use requires effective coordination of people and organizational units. This paper presents a solution to support this coordination by exploring concepts from the area of communities of practices for PFE. The expected benefit is that communities offer a more adaptive approach supporting the transition towards PFE compared to formal organizational restructuring. In this paper, we will describe our considerations about the building principles for organizations using a shared platform for the family members and a study proposal that will examine the effects of knowledge brokering among Communities of Practice as a means to assist PFE.¹

1. Introduction

Software engineering is a knowledge intensive activity. The demand for increasingly complex and sophisticated products that must be delivered faster and at lower cost, fuels the search for more efficient techniques, processes and methods in the field. Modularization and component-based software engineering techniques are practices that have been widely adopted by many organizations producing software products. They allow for a certain amount of flexibility in the development process that helps to address these challenges. Modules can be developed in parallel or may be supplied by

¹ Parts of the reported work has been funded by the project “Virtual Office of the future”, “Stiftung Innovation Rheinland-Pfalz”. Other parts of the reported work have been funded by the project Software Process Improvement through Knowledge Engineering (SPIKE) project, a research/industry collaboration partly funded by the Norwegian research Council.

external organizations. Simultaneously, as a consequence of component-orientation and increased complexity, developers lose their holistic view on systems and become specialists of certain subsystems only.

This often causes problems when organizations want to exploit the potential in component-based software engineering of producing multiple products based on common assets. We call this Product Family Engineering (PFE)². PFE seeks to optimize the resource usage in this production process through management of business, architecture, process and organization [1]. Therefore, PFE adds another level of complexity to software engineering processes: individuals are faced with challenges of managing profound and deep knowledge related to performing their own tasks. Specialization increases the boundaries between the individuals, hindering creative collaboration among the groups and may ultimately lead to reduction in job satisfaction.

Communities of Practice (CoP) [2] promise to solve parts of this problem. On the one hand, they enable concurrent and somewhat independent development of products. On the other, they effectively stimulate knowledge sharing among groups of people. As little research is done in the field of knowledge management in PFE organizations so far, we will use a rather open approach in our investigations. We want to examine four different questions to explore CoP in a PFE context: 1) Are there particular aspects of PFE that can benefit from applying communities as organizing principle? 2) Can brokering activities from CoP support PFE organizations in ensuring good knowledge distribution among the workers? 3) How should responsibilities be assigned within the communities? 4) How should we deal with the evolution of assets?

The remainder of the paper is structured as follows: First we introduce PFE and some key problems related to knowledge management. Then we describe Communities of Practice and explain why we see it as an interesting approach to address the problems. In Sections 4 and 5 we explain our proposed techniques; firstly the set of responsibilities (roles) that will benefit the organization of the communities and subsequently a proposed scheme for supporting change management of evolving assets. In Section 6 we describe our research framework in which our investigations will be conducted. Finally we give an overview of related work and an outlook to future research.

2. Product Family Engineering

As the complexity and size of software systems increases people need methods, processes and principles to cope with the scale. One commonly used principle is *divide and conquer*. This may manifest itself as recursive modularization and component-based development. That is, at a certain level of complexity, the software system is partitioned into multiple levels of modules or components (i.e. modules consisting of sub modules, components consisting of sub components). As the

² We use the term Product Family Engineering throughout in this paper. We regard the term Product Line Engineering, used by others, fully equivalent.

complexity reaches a certain level, these components are likely to be developed by different people, different parts of the organization or even different organizations. After some time, many organizations will establish a repository of such assets, including significant experience related to their use and limitations.

Now, the company might see a potential to exploit this repository to support a product family – for example with the objective to target different markets with somewhat similar products, using distinct combinations of these assets. This reduces the amount of product specific code that goes into each product, thus promising to reduce cost of development and time to market. Similarly, valuable experience related to the use of the assets is maintained. We believe it is important to understand aspects related to how the organization can use this experience to support continuous learning. The figure below illustrates the problem context.

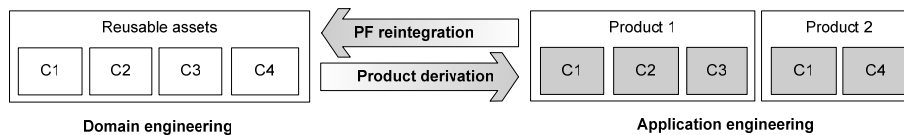


Fig. 1: Product derivation and asset reintegration

According to [3], a software product family is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way. Therefore, PFE is the key approach for managed, intra-organizational, large scale reuse, thus leveraging the opportunities that are offered by the application domain of a software organization. On a high level, the activities to achieve this reuse are subdivided into domain engineering and application engineering (see figure 1). Domain engineering encompasses the activities to create and maintain a complete and consistent set of artifacts to be reused by developments of software systems in a certain domain (i.e., a product family platform). Application engineering subsumes all activities reusing artifacts in such a platform while developing single systems (i.e. family members) [4]. For assets within the product family platform, the variability is explicitly captured. This explicit management of variability differentiates PFE from other reuse approaches like framework or library engineering, which mostly focus on the identification of commonalities.

However, facilitating large-scale reuse – and thus justifying the investments in the software product family - implies a two-way evolution. First, the PFE approach itself has to be adapted to the changing needs of the organization. Second, PFE implies a controlled evolution of these assets to keep the integrity of the product family platform. On the one hand, PFE has to coordinate more people across more organizational sub-structures like project, business units or departments compared to single systems engineering. On the other hand, the freedom of those sub-structures should be kept to the largest extent.

Achieving these ambitions requires elaborate planning, operative support and careful attention to organizational capabilities [5]. An often touted benefit of PFE is the potential for streamlining the product production process by organizing the workforce into two groups, one building reusable assets (i.e. domain engineering),

and one building products from these assets (i.e. application engineering). In this way, the organization can foster specialist knowledge among the people developing reusable assets while allowing people with solution building strengths to exploit the asset repository in assembling complex products. However, the effects of this segmentation have not yet been thoroughly investigated [1]. Furthermore, another problem from practice with this style of development is that specialists only receive indirect and sometimes insufficient feedback from field applications.

Capturing, representing and sharing domain knowledge in a PFE organization can be assisted by artifacts such as patterns, frameworks and reference architectures [6]. Also, proper knowledge representation eases communication of architectural principles through the development organization [7]. Additionally, relevant domain knowledge not only concerns the structure and contents of the domain, but also organizational and managerial aspects. For example, if a company has a divisionalized structure such as a set of business units, this may impact its software assets. Capturing and explicitly documenting such attributes enriches our knowledge of these assets. Furthermore, experience with PFE evolution shows that it is often hampered by tacit assumptions. These may be technical (such as standards concerning the underlying platform on which products rely), but equally often they are managerial or organizational in nature [8]. Thus, we believe it is worthwhile to further investigate knowledge management practices within PFE organizations.

This leads to the following three challenges concerning PFE: introduction, asset usage and asset evolution: During *introduction* of PFE, the members of an organization need to learn how to work together in a product family context. The most promising approach is to incrementally add assets to a product family platform, which implies a scaleable organization of this learning effort. The *asset usage* needs to be supported by platform developers that help appliers in integrating assets into the systems derived from the product family (i.e. during application engineering). Concerning *asset evolution*, feedback gained from asset usage gives the asset developers directions on how to evolve asset to met the needs of the appliers (i.e. during domain engineering). The summarized, underlying challenge is to optimize communication among product family stakeholders, thus reducing the overhead to coordinate their activities.

3. Communities of Practice and brokering

To address this coordination we investigate an online community of practice as a flexible organization form addressing these challenges. In the course of this paper we adapt the definition of [2, 9, 10]: Communities of practice are understood as a group of people sharing the same (professional) interest and who interact regularly to learn how to do their jobs better. Such communities differ from normal business units in that their purpose is knowledge-creation, and they are usually loosely connected, self-managed and informal. Communities of Practice enable the capturing and transferring of tacit knowledge by letting people from different departments in an organization discuss common topics of interest.

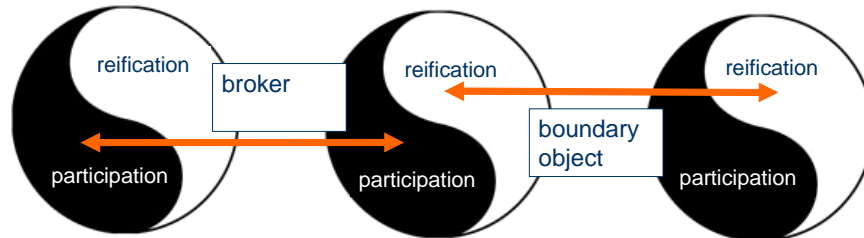


Fig. 2. Communities of Practice

Communities of Practice are said to consist of two main parts: *participation* in the community – interacting and “active involvement in social enterprises” [2] and *reification*, the process of creating artifacts from the community, like documents (see figure above).

Knowledge sharing between Communities of Practice is referred to as *boundary relations*, where connecting communities through participation is called *brokering*, and through reification *boundary objects*. Many practices can classify as brokering – like job rotation, physical design of work-environment to stimulate cross-sectional discussions etc. Typical boundary objects are documents that are shared between communities. In the case of PFE communities, brokers are prominent stakeholders in product family. The boundary objects are the assets within the product platform. Organizations report that such communities help improve communication and save time by “working smarter” [10]. Communities depend on communication, but not on a specific way of communication. The community can be organized around regular meetings, face to face or on-line, but also on interacting through mailing list or news group discussions between meetings. Communities often also include a role as a contact point to get relevant knowledge in a topic area for people not participating in the community. Because most people who are involved in a Community of Practice have their main responsibility in one of the business units, it is important to organize it in a flexible manner. Work should not depend on particular individuals being active at all times. Usually, one person in the core group is appointed as coordinator with the responsibility of arranging meetings.

We therefore define a product family community within an organization as the group of people using and contributing to the product family platform. The purpose (or mission) of this community is: To support the exchange of experience about product family related issues among members, to facilitate the adaptation and evolution of product family practices and to introduce new members to PFE. In this paper, we will present our considerations on how such communities can be established in a PFE context. In particular, we will describe a more detailed role model that focuses on the brokering aspect of communities, while the evolution of assets is emphasizing how boundary object are used within PFE communities.

4. Distributing Responsibilities for Product Family Assets

One major influence on optimizing coordination is a clear definition of responsibilities. Furthermore, this distribution of responsibilities needs to be clear and simple to avoid creating overhead while assigning and determining responsibilities. Our suggestion for this distribution is a simple general assignment based on the impact of assets, combined with a role model to clarify the responsibilities of the individual member of the community.

The classification in [11] provides a distinction of the assets within the platform: On the top level, it distinguishes domain assets and application assets. Application assets are work-products of application engineering, while domain assets capture the domain experience in a reusable form. A special sub-category of domain assets are meta-assets that contain information to reuse other assets. Since these meta-assets can be documentation of methodologies of PFE, they can also be used to introduce such methodologies.

To maintain the integrity of the platform, responsibilities have to be assigned to knowledgeable persons, preferably people with a good overview of the platform (i.e. platform developers). This is of particular importance for meta-assets, where each asset describes a certain view of the whole platform. This structure also helps to evaluate appropriate mechanisms for evolution management: Assets with a high impact are more likely to receive more feedback, since more users are involved compared to assets with lower impact. In addition, faults in these assets will affect more users. Therefore, the evolution mechanisms should be simple and impose low overhead. The quality control of the evolved asset should be more rigid. For less impacting assets like application assets, the feedback can be more complex, whereas inconsistencies can be (temporarily) tolerated.

Based on these considerations, we adopt the generic role model for communities presented by [12]: Visitor, newbie, member, leader, senior. Furthermore, we will also investigate the motivation of being in the product family community: The *visitor* could be any software developer inside the organization that is not already a member of the product family community. To allow the visitor to decide whether to become a member, meta-assets and description of generic assets should be readable with no registration. However, to get an overview of the product where a certain asset is used, the access to solution assets might be restricted. A visitor becomes a *newbie* when s/he actively wants to use the results of the product family, i.e., when the visitor registers to download a component. Therefore, the membership of the newbie can be recognized, and further educational activities can be triggered. The newbie becomes a regular *member* when s/he provides feedback about asset application, which implies that the applied solution asset is used inside a development project. The main motivation for *newbies* as well as members is to use the assets within the product family platform. Since the product family platform provides more support to a member as long as changes do not affect the integrity, there is also motivation to invest extra effort in maintaining this integrity. The next role level is the *leader*, when the regular member coordinates community activities. For each asset, at least one leader is assigned that is responsible for coordinating further implementing members during the evolution of the asset and for handling feedback and questions. Since this role needs additional effort that might not be covered, the organization should provide

additional effort for performing this role. Some of the active leaders might become *seniors*, which take care of evolving the product family as a whole. These seniors also act as leaders for meta-assets supporting product family application.

The application of communities in PFE allows measuring the impact on the business of an organization in a concrete way: First, the impact of an asset is explicitly described by its variability. Second, the quality and benefit of assets are validated by application in systems derived from the product family. This allows an organization to justify investments into the community like the assignment of effort for experts mentioned above.

5. Issue Management for Evolving Assets

Controlling the evolution, i.e. managing the changes to an asset in the product family platform, is key to avoid architecture degradation. The explicit variability of product family assets allows identifying affected users not only on an asset level, but based on the variability actually used by certain members. These subgroups created by variability usage allow handling the following communication procedures even if a large number of members are using a certain asset. We propose three communication procedures for managing asset related issues: Relate and comment, consensus based, and transfer and re-integrate.

Relate and comment is used when the users of an asset might detect problems during reusing an asset, but do not have the competence or experience to decide whether this change affects the integrity of the product family or not. In this case, users are allowed to comment the asset, but are not allowed to change its content. An example would be a change in a meta-asset like a process description about product family methodologies.

Consensus-based is used when an asset is used by several users which also have the knowledge to change it in a way such that the product family integrity is not affected. Then, consensus about how the changes are implemented is reached through discussion. An example is a code component, where the implementation of missing variability is distributed among the users of this component.

Transfer and re-integrate is used when project pressure avoid changing an asset in a controlled way or when the effort to gain a consensus exceeds the effort to re-integrate the changed asset in the product family later. In this case, the asset is transferred into the baseline of the project an integrated later on.

6. Suggested study

As depicted in Fig. 3 (below), PFE represents knowledge about the product families mainly from two perspectives: the representation of the common and variable features in the product family (feature models - depicted in the left part of the figure) and the representation of technical assets like code components (structural models - depicted in the right part of the figure). Feature and structural models can be used by the Communities of Practice as the core mechanisms for brokering: feature models are

domain-level representations that can be easily understood by people that want to decide on the functionality of products; structural models are representations (of generic/reusable and concrete/product-specific solutions) that can be used by people producing or using reusable assets.

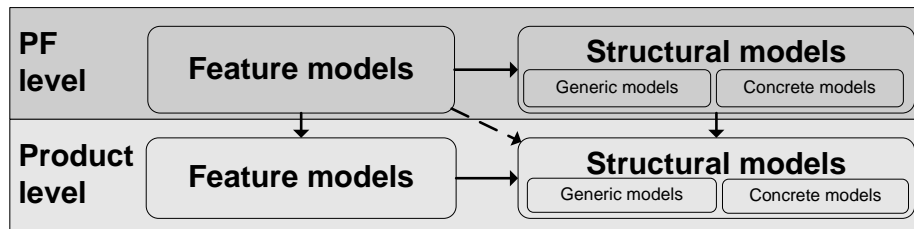


Fig. 3. Product Family modeling framework

Further, PFE knowledge can be represented at the family level (in the top half of the figure above) and at the product level (in the lower half of the figure). The Product Family level describes the aspects shared by all products in the family, whereas the Product level focuses on aspects that are specific to individual products.

The arrows represent traceability links that model the derivation dependencies between models. They can be navigated to share knowledge about artifacts. For example, the horizontal arrows (between feature model and structural model) can be used to learn how a feature has been implemented by a selected component. The vertical arrows (crossing the two levels) can be used to learn if/how elements at the product family level are supported by corresponding elements in different products (e.g. which features shared by the product family are supported in which products, or how a product family component is refined within a selected product).

For a product family to be successful, it is important to understand the factors related to adoption and refinement of reusable assets in the products. It is also important to understand how these aspects relate to the scoping of the PF level feature models, i.e. the ability of the asset repository to reflect the real needs in the product development groups. This is paramount in order to maintain a vital repository that is able to support the required variability in the product family. It is an enabler for organizational learning.

We intend to investigate how knowledge sharing through brokering between people working with reusable assets and products affects the dynamics of the product family, in particular the level of alignment of the repository of reusable assets at the system family level compared to the assets actually used in the delivered products. We will also study the effects of knowledge brokering with regards to reintegration of capabilities developed in the product organizations back into the system family. We will classify these effects for the different roles defined above (visitor, newbie, member, leader and senior) and across the three proposed communication schemes (relate and comment, consensus based, and transfer and re-integrate). The development repositories and issue management systems might act as a source of data for this.

Due to lack of validated metrics [13, 14], we have chosen a qualitative study. Through semi-structured interviews, we suggest studying two different organizations

working with product families, and choose a) a company that has few or no brokering between reusable asset and product departments, and b) a company that has many brokering practices in place. We will study brokering on functional and non-functional aspects of assets, as well as the extent to which organizational and managerial aspects that impact the assets, are taken into account, or had better be taken into account.

7. Related Work and Outlook

In this paper, we presented our initial consideration about product family communities as a scalable, member-centric approach to control the evolution of assets in product families. Other approaches address this issue from an organizational point of view:

Bosch [15] gives an overview of four different explicit organizational structures suitable for product families (Development Department, Business Units, Domain Engineering Units, Hierarchical Domain Engineering Units) and also gives criteria when to apply which organizational structure. In [16] he organizes these structures according to the three dimensions concerning scope of product family platform, responsibilities level and lifetime of product family activities. Schmid [17] explains why an organizational structure based on features rather than on development phases offers a more promising approach. He also lists a range of communication patterns and organizational success factors of product family approaches. In between the organizational and individual point of view is the experience report of the Owen software cooperative at HP by [18].

Our approach to establish product families with communities does not contradict these approaches. Product family communities can be organized according to these approaches, in particular when product families have been established within an organization.

Beyond the survey presented in this paper we see two major directions for future research in software product families: 1) Analyzing and adapting organizational principles in Open Source development. Product family communities are based on voluntary participation like open source communities. Successful organizational patterns (e.g. meritocracy) from open source development might support the instantiation of product family communities. However, these patterns might need adaptation due to the commercial and intra-organizational setting. 2) Quantifying the business benefits generated by communities. This is a problem for communities in general [19]. However, product family communities create a magnitude of analyzable artifacts (code, documentation), which builds a promising basis for such analysis.

8. References

1. Linden, F.v.d., *Software product families in Europe: The ESAPS and CAFÉ projects*. IEEE Software, 2002. **19**(4): p. 41-49.

2. Wenger, E.C., *Communities of Practice: Learning, meaning and identity*. 1998: Cambridge University Press.
3. Clements, P. and L. Northrop, *Software product lines: Practices and Patterns*. The SEI series in software engineering. 2002: Addison Wesley. 563.
4. Muthig, D., *A Light-weight Approach Facilitating an Evolutionary Transition Towards Software Product Lines*. 2002, Stuttgart: Fraunhofer IRB Verlag.
5. Bühle, S., et al. *Exploring the context of product line adoption*. in *5th International workshop on Product Family Engineering*. F.V.d. Linden, Editor. 2003. Siena, Italy: Springer Verlag.
6. Hallsteinsen, S., T.E. Fægri, and M. Syrstad. *Patterns in Product Family Architecture Design*. in *5th International workshop on Product Family Engineering*. F.V.d. Linden, Editor. 2003. Siena, Italy: Springer Verlag.
7. Mustapic, G. *Real World Influences on Software Architecture - Interviews with Industrial System Experts*. in *WICSA-4*. 2004. Oslo, Norway.
8. Lago, P. and H.v. Vliet. *Observations from the Recovery of a Software Product Family*. in *Third software product line conference*. 2004: Springer.
9. Wenger, E.C., *Communities of practice: A brief introduction*.
10. Wenger, E.C., R. McDermott, and W.M. Snyder, *Cultivating Communities of Practice*. 2002, Boston: Harvard Business School Press.
11. Becker, M., *Anpassungsunterstützung in Software-Produktfamilien*, in *Technical Universität of Kaiserslautern, Department of Computer Science*. 2004.
12. Kim, A.J., *Community Building. Strategien für den Aufbau erfolgreicher Web-Communities*. 2001, Bonn: Galileo Press.
13. Fenton, N. and M. Neil, *Software metrics: A roadmap*, in *The future of software engineering*, A. Finkenstein, Editor. 2000, ACM Press: New York.
14. Seaman, C.B., *Qualitative Methods in Empirical Studies of Software Engineering*. IEEE Transactions on Software Engineering, 1999. **25**(4): p. 557-572.
15. Bosch, J. *Organizing for Software Product Lines*. in *In Third International Workshop on Software Architectures for Product Families*. 2000.
16. Bosch, J. *Software Product Lines: Organizational Alternatives*. in *Proceedings of the 23rd International Conference on Software Engineering*. 2001: IEEE Computer Society Press.
17. Schmid, K. *People Management in Institutionalizing Product Lines*. in *In Proceedings of Net.ObjectDays*. 2003.
18. Toft, P., D. Coleman, and J. Ohta. *A Cooperative Model for Cross-Divisional Product Development for a Software Product Line*. in *Proceedings of the First Software Product Line Conference*. P. Donohoe, Editor. 2000: Kluwer Academic Publishers.
19. Millen, D.R., M.A. Fontaine, and M.M. J., *Understanding the benefits and costs of communities of practice*. Communications of the ACM, 2002. **45**(4): p. 69-73.