The Economic Impact of Product Line Adoption and Evolution

Klaus Schmid, *Fraunhofer Institute for Experimental Software Engineering*
Martin Verlage, *Market Maker Software AG*

Software is increasingly turning into a commodity; thus, people increasingly expect systems that are customized to their needs. This situation is forcing nearly every software development organization to develop multiple variants of their systems to serve the specific needs of different customers or market segments. Thus, many, if not most, software development organizations are finding that they need to build families of systems or *product lines*.

Experience shows that a company can drastically improve its competitive advantage if it optimizes how it develops these product lines.[1,2] Using product line engineering, some organizations have reduced the number of defects in their products and reduced costs and time to market by a factor of 10 or more.[1,3] However, many companies don't use a product line engineering approach when developing their product lines. More often than not, they either start from a single system, branching off new variants as the need arises and ending up with completely independent code bases, or they start with the different variants as independent projects from the beginning.

Product line engineering focuses on developing multiple variants of systems jointly, thus exploiting the commonality among systems in the form of reuse.[1] The key to successful product line engineering approaches, such as Pulse[4] (a component-based product line development approach developed at the Fraunhofer Institute for Experimental Software Engineering), is to identify early on a reference architecture that provides a blueprint for producing different variants. The structural similarity among the variants, resulting from the common architecture, enables developers to reuse components across a range of different products in the product line. However, when implementing product line engineering, a wealth of options exists, so companies must make wise decisions to optimize their economic benefit. To exemplify this, we discuss Market Maker Software AG's[3] Merger product line.

When transitioning to product line development, an organization must consider the adoption context and should use product line scoping techniques to optimize the economic benefits.

Product line adoption schemes

In theory, the optimal product line development adoption scheme is to set up a com-

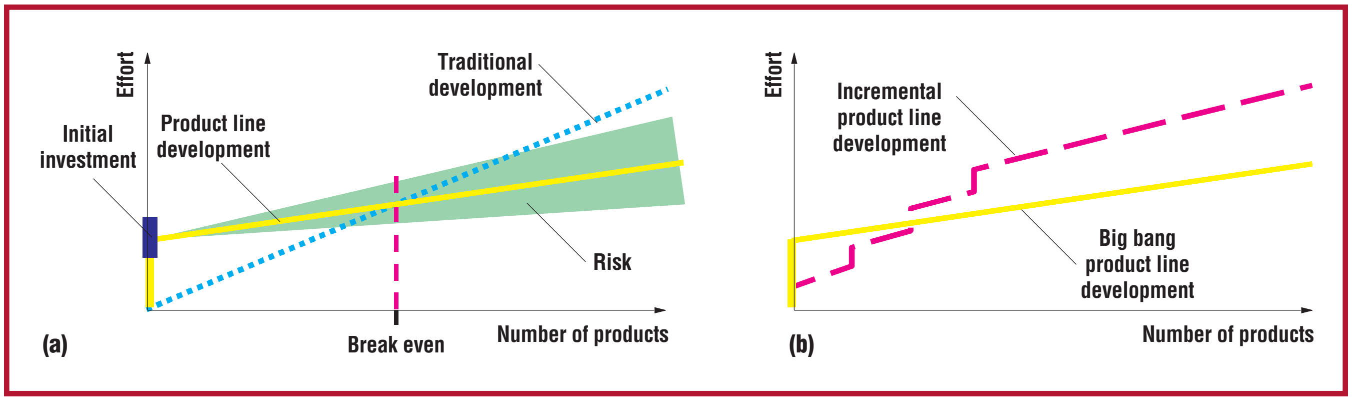

Figure 1. Product line investment curves: (a) the big bang approach, including risks; (b) the big bang versus incremental approach.

pletely new product line by developing a reuse infrastructure for the whole range of products right from the start. We often call this the big bang approach. You can use this infrastructure to develop new products, which could drastically cut costs compared to traditional stovepipe development. Of course, when first planning a product line, predicting the specific investments and benefits is hard, so economic results will contain some uncertainty.

Unfortunately, this ideal approach is hardly ever adequate in practice. In principle, strong upfront planning should let you develop assets that support the full range of functionality the product line requires, but organizations often use a more incremental approach. With an incremental approach, you develop assets to support the next few upcoming products, deliberately excluding highly uncertain potential products. Over time, you need to extend and adapt assets to address further products. Usually, constraints on available resources force this more incremental approach, but it's generally useful even with unlimited resources because of the intrinsic uncertainty of future products and their requirements. Figure 1a shows the ideal big bang pattern, and Figure 1b compares it with the corresponding patterns for the incremental approach.

Regardless of which approach you use, it is best to first distinguish several basic situations from which product line adoption can start. You can then link each situation to corresponding strategies (or adoption schemes) and connect a different pattern of investment and resulting benefits to each one.

We can distinguish four main types of situations for adopting product line engineering:

- *Independent*. The company starts a new product line without any predecessor products.
- *Project-integrating*. Existing systems are already under development to address a new market. As part of product line development, the software engineers integrate the systems so that they can derive them from the same reuse infrastructure.
- *Reengineering-driven*. Legacy systems already exist, but the engineers can't use them for product line development—rather, they need to perform a nontrivial reengineering effort.
- *Leveraged*. The company sets up a new product line (to address a new market) based on a product line that is already in place.

In practice, these situations often overlap.

Consider Market Maker's Merger product line. Market Maker started in 1990 as a one-person company with a single product: a DOS-based system for tracking stock information. It has since grown to 60 employees, but to optimize its limited resources, Market Maker has always used a product line approach. Even in the DOS version, various modules were available to address specific data-processing needs, and customers could independently bring in other modules. (However, the implementation level only had a single variant and certain menu entries enabled, based on the given license key.)

As the company created new software platforms, it found it had to develop more sophisticated approaches to address increasingly complex variability needs. In 1995, it started developing a new software system aimed at supplanting the original DOS-based product and transferring its potential to the Windows-based age. Like the original product, the new system supported the module concept. However, over time, the company created additional implementation-level variants (for special applications) and

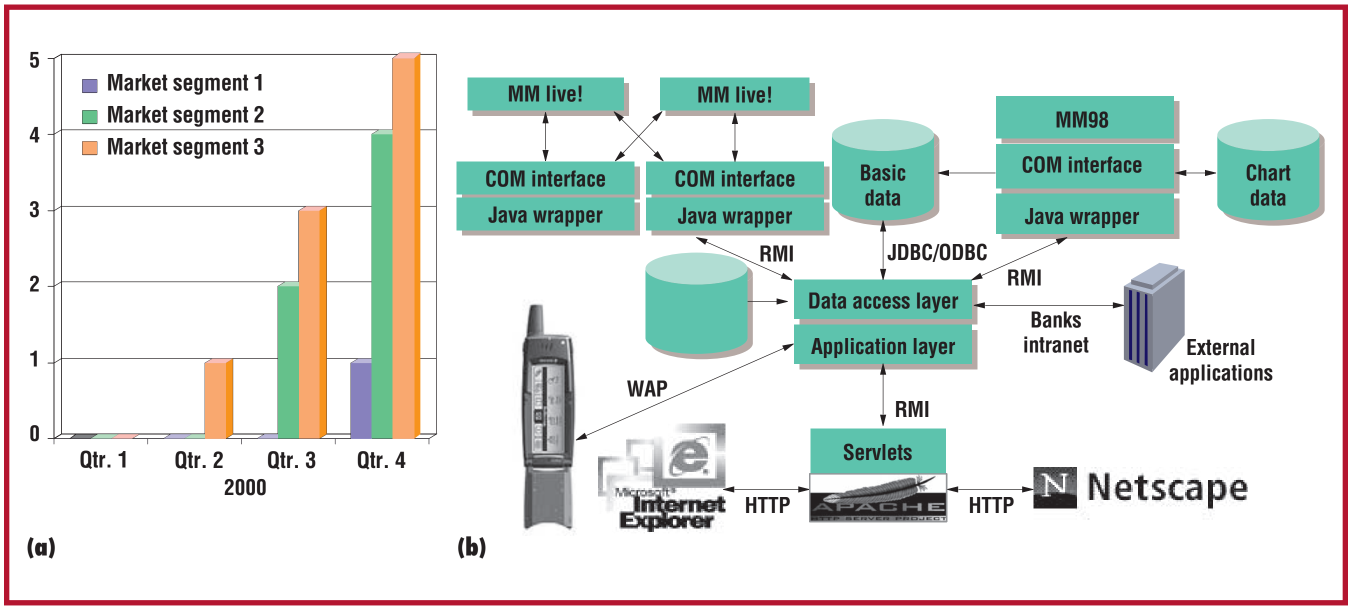

Figure 2. The Merger product line: (a) systems delivered over time and (b) the product line architecture.

entry-level mass-market variants. These variants constituted the Market Maker product line—the company produced them from a single, common code base.

In 1999, the company decided to enter the market of Internet-based stock-market information systems. It thus created the Merger product line, which was based on a completely new infrastructure developed in Java (Martin Verlage managed the Merger product line's setup and evolution). Currently, this product line includes about 15 variants addressing three different market segments (see Figure 2a). When developing Merger, the company decided not to replicate functionality that already existed in the Market Maker product line—such as the MM98 and MM live variants that address the historical data feed and the real-time data feed, respectively. So, it created specific variants of the Market Maker product line and used them as data servers in the Merger installations (see "MM98" in Figure 2b).

When adopting and evolving its product line approach for these two product lines, Market Maker applied the schemes we discuss here with huge success.

Product line entry and exploitation potential

Depending on the specific situation in which you start product line development, you usually find different patterns of how the company incrementally develops the product line and, similarly, different patterns of investment and return on investment. In particular, you need to determine whether the increment originates by successively extending and adapting the reuse infrastructure for additional products, or whether the company extends it by successively adding assets covering additional functionality.

Independent adoption

The independent product line adoption scenario is the prototypical product line situation (see Figure 1a). Because no products exist yet, the company can plan in detail and optimize its product portfolio. Compared to the product line's overall setup time, the planning time would be rather low, so it wouldn't significantly increase the time to market.

However, starting a completely new product line means venturing into the completely unknown. Thus, technical feasibility studies and detailed market analyses are necessary to control the overall uncertainty. Furthermore, even if you use these measures, you usually still have significant uncertainty, because, for example, some products could become more or less important as product development progresses.

In the context of the Merger product line, the company made a detailed analysis of potential portfolios, identifying major market segments, key requirements, and so forth. Additionally, it performed technical feasibility studies and competitor surveys. Although these analyses were rather thorough, plans still required adjustment. For example, the company addressed some market segments later than anticipated or not at all, because more products than expected could be delivered to the customer in the initial market segments. Despite these deviations, it is clear in retrospect that the initial efforts were not wasted. Rather, they played a key role in focusing the reuse infrastructure's development, so now Market Maker can efficiently develop new variants.

Project-integrating adoption

From our experience in industrial practice, we've found that the independent situation is rather uncommon. Usually, several products exist in a company that have some commonalities but were more or less independently developed. There is continuous pressure to bring to market new products, so it is impossible to put product development on hold to focus on developing an integrated product line infrastructure. Rather, an incremental approach is required, where key components are successively generalized into common, reusable components.

A special case of the project-integrating situation exists when two product line infrastructures must merge. Such a situation is currently occurring at Market Maker—the company is integrating the original Market Maker and Merger product lines into a single product line architecture. Although companies can usually avoid project-integration by replicating functionality among the two product lines, at times profound differences in the nonfunctional requirements make it necessary. In this situation, the company integrates the reuse infrastructures, focusing on integrating the replicated components. Because the company must continually derive new products and releases from the product lines, it can only perform an incremental, component-wise integration of the reuse infrastructures—similar to the incremental pattern Figure 1b shows. However, if compared to the initial situation, the entrance barrier (the effort required before you can derive the first products from the resulting infrastructure) is usually lower, whereas the number of steps (effort to extend the product line infrastructure) will be higher. Also, the company generally must expect more problems with the degradation of the reference architecture.

This is the general adoption pattern for project-integrating product line development—create a common infrastructure by integrating technical areas in a component-wise manner. This leads to an incremental approach, similar to the one in Figure 1b. However, compared to the initial situation, the entrance barrier (effort required until the first products can be derived from the resulting infrastructure) will usually be lower, while the steps (effort to extend the product line infrastructure) will usually be higher. Also, more problems generally occur with the degradation of the reference architecture over time.

Reengineering-driven adoption

A company usually undertakes reengineering-driven adoption if it finds that its software development is bound to hit a wall. This can manifest itself in many ways—for example, if the cost of product development grows too high or if it becomes impossible to derive new, envisioned products based on the available systems. Companies typically will tolerate a certain level of pain before making the investments associated with performing a major reengineering effort. However, when they do undergo such reengineering, it's then fairly easy to introduce the additional effort required to plan a product portfolio. Reengineering can either focus on packaging the existing legacy system as a whole or it can aim at a component-wise approach.

This situation is similar to independent adoption and its economic patterns. If the company packages the legacy as a whole, it incurs rather large investments in the beginning but significantly reduced costs for developing future systems (see Figure 1a). If the company performs component-wise packaging, an incremental pattern (see Figure 2b) will result, but reengineering generally requires a large initial investment compared to typical project-integrating situations.

A good example of the component-wise approach to reengineering-driven adoption is the Ramsis kernel redesign project that Fraunhofer IESE performed with a small company. This project focused on a reengineering-driven product line design for a large legacy system of ergonomy simulations.[5] It started with a significant reengineering effort for identifying components in the existing system and packaging them to turn the system into an appropriate basis for further product line development.

Developing the Merger product line also mirrors this situation, because, as discussed earlier, the company reused Market Maker

There is continuous pressure to bring to market new products, so it is impossible to put product development on hold to focus on developing an integrated product line infrastructure.

From an economic viewpoint, a leveraged product line adoption entails a revolution, because the company can address a completely new market segment with low costs and few risks.

functionality as servers, and prior to that, it had to add specific interfaces to these servers. However, for Merger, packaging only required augmenting the existing systems with appropriate interfaces using COM. Also, the Merger product line was built on top of the existing one, which is why it's more appropriately characterized as leveraged product line adoption.

Leveraged adoption

Leveraged product line development is perhaps the most sophisticated approach to product line adoption. As opposed to the other patterns, it requires an existing product line and is characterized by a shift to a new market (system type). Examples of such a shift are Cummins Engines, which expanded its original product line for car and truck diesel engines to arbitrary industrial diesel engines, and CelsiusTech, which leveraged its product line of battle ship control systems by entering the market of civil air-control systems.[6] In these cases, the existing product line infrastructures provided leverage for entering the new market, giving the company a competitive advantage right from the start.

The Merger product line is clearly a case of a leveraged product line. The existing Market Maker product line offers leverage by providing data gathering, data management, and aggregation services, while the Merger infrastructure mainly focuses on data transformation and online presentation tasks. Market Maker packaged its original product line's functionality into Merger in the form of servers. This partitioning of the reuse infrastructure also benefits Merger products through ongoing development on the Market Maker product line.

From an economic viewpoint, a leveraged product line adoption entails a revolution, because the company can address a completely new market segment with low costs and few risks by building on an existing product line infrastructure. However, as in other situations—in particular, the independent situation—the company must perform a detailed product portfolio analysis, technology studies, and risk analysis. Similarly, leveraged adoption usually requires an initial investment and then shows a steady growth in the number of systems (see Figure 1a). For Merger, the leveraged approach proved to be highly successful. However, Figure 2a shows a slightly different pattern of nearly exponential growth. The reason for this is that the Merger reuse infrastructure itself grew over time. Thus, later systems could be built with more reuse.

Product line evolution

The main factor determining how a product line evolves is how much deviation the organization allows before reunifying the infrastructure. We can distinguish three basic situations for product line evolution (not taking into account replacing the infrastructure's parts over time).

In the first situation, *infrastructure-based evolution*, new product requirements that might be reusable immediately lead to a generalization of the product line infrastructure. Thus, the organization can avoid the problem of multiple implementations of the same requirement. However, this usually results in many changes to the product line infrastructure. A specific product (the first one in need of a new requirement) triggers each change, which is implemented in a way adapted to the next few products (see Table 1). Market Maker took this approach with its Merger product line. If it hadn't, the simultaneous demands for many new variants would have created a strong dispersion into variants of the product line infrastructure. The advantage is that for the second product requiring the functionality, it is already reusable. This can lead to the superlinear increase that Figure 2a shows.

The second situation, *branch-and-unite*, is common in industrial practice. Here, the organization creates a new version branch for a new variant and then reunifies this branch with the original infrastructure after releasing the product. In this case, the organization typically considers only a single product, although more experienced organizations also consider requirements for future products when determining an adequate implementation. Market Maker successfully pursued this approach with its first product line. The main reason for applying this approach is that new variants are rather infrequent and thus usually nonoverlapping. This wasn't the case with the Merger product line.

Some organizations end up in a *bulk* situation, which allows larger branching of the reuse infrastructure. Then, at certain intervals, the organization reintegrates the prod-

Table 1

Product line adoption and evolution patterns

	Situation type	Product line planning look-ahead	Approach
Adoption	Independent	Broad portfolio of future systems	Big bang
	Project-integrating	Medium-size portfolio of future products	Incremental, by functional area or component
	Reengineering-driven	Broad portfolio of future products and legacy products	Incremental, by functional area or component, or big bang, by packaging existing legacy as a whole
	Leveraged	Broad portfolio of future products	Big bang
Evolution	Infrastructure-based	A small number of products	Incremental, by product
	Branch-and-unite	Single product	Incremental, by product
	Bulk	A small number of products (perhaps a market segment)	Incremental, by product group

Table 2

Scoping techniques and their relation to product line adoption

Mode of product line extension	Portfolio definition	Domain-potential analysis	Reuse infrastructure scoping
Partial big bang and evolution by product group	Very important	Recommended, but mainly for risk analysis	Recommended to support architecture definition
By (single) product	Not necessary	Only needed if the extension requires restructuring	Only needed if the extension requires restructuring
By component or functional area	Should be performed	Key for identifying the next component for product line extension	Should be applied to support architecture definition

uct line infrastructure. Larger organizations usually apply this approach, but it's best to avoid it as much as possible. It not only leads to major reintegration efforts (mapping to big jumps in the economic curve in Figure 1b), but it also usually entails significant synchronization efforts and quality problems.

The different patterns of product line evolution we've identified have different requirements in terms of look-ahead planning and the number of products simultaneously integrated into the product line infrastructure (see Table 1).

Product line planning techniques

How an organization performs product line adoption and evolution strongly influences its product line's overall economic results. However, even if it selects a specific adoption approach, it still must decide which products to consider when developing or extending the product line infrastructure, which technical areas to integrate next into its product line infrastructure, and which requirements reusable assets will directly support. Just as the basic adoption and evolution steps determine the product line development's basic economic pattern, answers to these questions help fine-tune product line development and its economic characteristics. Restrictions for answering the questions depend on the specific adoption or evolution situation (see Table 2).

Based on the types of decisions that must be made, we can distinguish the following three levels of decision making—or scoping—in the context of product line engineering:7

- *Product portfolio scoping*: Which products shall be part of the product line?
- *Domain-based scoping*: Which technical areas (domains) provide good opportunities for product line reuse?
- *Reuse infrastructure scoping*: Which functionalities should the reuse infrastructure support?

Affluency in these three scoping techniques will help you make the right decision when adopting and evolving a product line.

Product portfolio scoping helps establish a detailed vision of the products and their

About the Authors

Klaus Schmid is competence manager for value-based product line development at Fraunhofer IESE, where he has been involved in several projects that have transferred product line engineering concepts to industrial environments. He was also a member of the Pulse development team. His main research interests are the economic aspects of product line development and approaches for introducing and institutionalizing product line development in industry. He received an MS in computer science from the University of Kaiserslautern. Contact him at Fraunhofer Inst. for Experimental Software Eng., Sauerwiesen 6, D-67661 Kaiserslautern, Germany; klaus.schmid@iese.fhg.de.

Martin Verlage is director of the Online Products business area at Market Maker Software AG. His main software development interests are in the area of component-based software engineering, especially architecting and testing. He received an MS and PhD in computer science from the University of Kaiserslautern. He is a member of the Gesellschaft für Informatik e.V. Contact him at Market Maker Software GmbH, Karl-Marxstr. 13, D-67655 Kaiserslautern, Germany; m.verlage@market-maker.de.

requirements. First, you identify the general market potential based on market analyses, taking into account the market structure, potential customers, end-user needs, and the positioning of competitors. Then, you identify the market segments that fit the company background. This usually happens in a workshop representing the most relevant stakeholder groups. While coming up with an integrated definition of the product portfolio, it is important to address questions such as, "Will the products compete with each other on the market?" and "How much will it cost to develop these products?" It helps even if you just informally ask these questions.

While setting up the Merger product line, Market Maker performed a detailed product portfolio scoping. It analyzed markets and competitors, developing a first vision of potential market segments and products. This provided the necessary input for technical feasibility studies. At the same time, it refined the initial vision of the portfolio in several iterations. This actually led to rather severe changes, such as introducing additional market segments in the product line vision. You only need to perform this full-size approach if you develop a new product portfolio. Otherwise, just identify changes to the product portfolio—technical feasibility and market studies are usually not so important.

Based on a product portfolio definition, you can perform domain-based scoping. With this approach, you identify the main technical domains relevant to the product line and analyze their reuse potential. Different technical domains, even within the same product line, can vary considerably in terms of their potential benefit and inherent risks for product line engineering. Market Maker applied this approach to domain-potential analysis, which is part of the Pulse-Eco method,[8] for the Merger product line. In this case, Market Maker observed variations from "extremely well suited for product line reuse" to "not suited at all." In particular, domain-based scoping identifies areas where a reuse investment is particularly meaningful, which is especially important if the product line infrastructure is built in an incremental manner (for example, in a project-integrating adoption situation). Furthermore, it can help you decide what functionality to integrate next into the product line. In a situation such as the one with the Merger product line, where the organization basically built a full product line infrastructure with the first product, this approach is typically used only to inform the development of potential reuse risks (see Table 2).

Once you identify the key areas for product line reuse, the important question is which functionalities should be made reusable in the context of the specific product line, which involves reuse infrastructure scoping. The Pulse-Eco approach supports this activity in a quantitative manner. With reuse infrastructure scoping, you develop quantitative models to capture the desired product line benefits. You then use these models to identify functionalities that will provide the highest economic benefit if made reusable. This provides economic input for the architecture definition. Because of this focus on guiding the product line's implementation, this form of scoping is particularly useful when large parts of the product line infrastructure are built for the first time. This is the case if whole product groups or new functional areas must be integrated into the product line infrastructure. Market Maker applied this approach when extending certain functional areas for its Market Maker product line. In this case, the approach provided valuable input for making the most appropriate functionality reusable.[7]

Product line development is about to change how we perceive and perform software development. This transition is similar to the one made from craftsmanship to industrial production. Although this transition is strongly based on the increased understanding of software architectures, it is also changing how organizations go about their software business. During the industrial revolution, becoming a modern company involved more than just adding an assembly line to the factory floor. Likewise, it will not be sufficient for organizations to switch to product line development in an arbitrary way. Rather, a company must adequately adopt such an approach, and the approach must evolve from the perspective of potential economic benefits. To successfully reap these benefits, companies will have to develop an in-depth understanding of product line development's economic implications and how these implications relate to the possible product line adoption and evolution mechanisms.

Acknowledgments

The Eureka 2023 Programme, ITEA (Information Technology for European Advancement) projects ip00004 and 99005, Café (from *c*oncepts to *a*pplication in system-*f*amily *e*ngineering), and ESAPS (Engineering Software Architectures, Processes, and Platforms for Systems-Families) partially supported the work presented in this article.

References

1. P. Toft, D. Coleman, and J. Ohta, "A Cooperative Model for Cross-Divisional Product Development for a Software Product Line," *Proc. 1st Software Product Line Conf.* (SPLC1), Kluwer, Dordrecht, Netherlands, 2000, pp. 111–132.
2. J.C. Dager, "Cummin's Experience in Developing a Software Product Line Architecture for Real-Time Embedded Diesel Engine Controls," *Proc. 1st Software Product Line Conf.* (SPLC1), Kluwer, Dordrecht, Netherlands, 2000, pp. 23–46.
3. L. Northrop and P. Clements, *Software Product Lines*, Addison-Wesley, Reading, Mass., 2001.
4. J. Bayer et al., "PuLSE: A Methodology to Develop Software Product Lines," *Proc. 5th Symp. Software Reusability* (SSR'99), ACM Press, New York, 1999, pp.122–131.
5. J. Bayer et al., "Transitioning Legacy Assets to a Product Line Architecture," *Proc. 7th European Software Eng. Conf.* (ESEC'99), Springer Verlag, New York, 1999, pp. 446–463.
6. P. Clements, "On the Importance of Product Line Scoping," *Proc. 4th Workshop Product Family Eng.* (PFE'4), Springer Verlag, New York, 2001, pp. 70–78.
7. K. Schmid, "Scoping Software Product Lines: An Analysis of an Emerging Technology," *Proc. 1st Software Product Line Conf.* (SPLC1), Kluwer, Dordrecht, Netherlands, 2000, pp. 513–532.
8. K. Schmid, "A Comprehensive Product Line Scoping Approach and Its Validation," *Proc. 24th Int'l Conf. Software Eng.* (ICSE'02), ACM Press, New York, 2002, pp. 593–603.

For more information on this or any other computing topic, please visit our Digital Library at http://computer.org/publications/dlib.