point

Being Proactive Pays Off

Paul Clements

One of the most exciting aspects about software product lines is how they put technical and enterprise issues on the same page.

Software product lines represent a paradigm on the rise in software engineering that comes with true order-of-magnitude improvements in cost, schedule, and quality. As the field grows and matures, case studies are becoming more plentiful and beneficial. Books, papers, conferences, workshops, and special issues of magazines such as this one provide ideas that can inspire us.

For me, one of the most exciting aspects about software product lines is how they put technical and enterprise issues on the same page. This is best demonstrated when a software product line capability helps a savvy organization quickly enter and thrive in a whole new market area. CelsiusTech Systems, a Swedish seller of shipboard command-and-control systems, recognized that a new market lay nearby in ground-based air defense systems—guns mounted on moving platforms. On the first day CelsiusTech decided to enter that market, 40 percent of its entry system was complete because of its roots in a ship system product line.[1] Cummins, an American manufacturer of diesel engines, recognized that a vast untapped market in industrial diesel engines lay right next to its product line of (software-intensive) automotive and truck diesel engines. The industrial diesel domain encompasses an extraordinary range of applications, from ski lifts to rock crushers, but no single application is a high-volume proposition. Without the capability to field a product variant quickly and easily, the market is not attractive. But with that capability—that is, with a product line capability—an organization can score a coup, which is precisely what Cummins did.[2]

The key to this enterprise-level strategic positioning is understanding the *scope* of the product line. A product line's scope states what systems an organization would be willing to build as part of its product line and what systems it would not. In other words, it defines what's in and what's out. Defining a product line's scope is like drawing a doughnut in the space of all possible systems. The doughnut's center represents the set of systems that the organization

continued on page 4

counterpoint

Eliminating the Adoption Barrier

Charles Krueger

Stories of successful software product line deployments often read like epic adventures. In the end, there is triumph of inspirational proportions, but along the journey there is risk, hardship, sacrifice, heroes, antagonists, love lost, love found, and fortuitous events of both happy and tragic consequences. For example, for Cummins to achieve its impressive software product line successes, it stopped all product deployments for six months while it rearchitected its engine control software, support technology, organizational charts, and processes. Imagine the consequences if, after an extended production shutdown, unanticipated events had led to project failure.

Although these epics make for great, inspiring reading, many software organizations need to operate on a more predictable and less dramatic story line. They can't afford to slow or stop production for six months to reinvent themselves, even if the potential payoff is huge. For most organizations, the risks, timetables, efforts, and costs experienced by the pioneers represent an adoption barrier to software product line practice.

For software product lines to become part of mainstream software engineering culture, organizations need software product line strategies with low adoption barriers. They need low-risk strategies that afford small upfront effort, incremental transition from current practices, and rapid return on investment. Several organizations have recognized this need and are successfully creating technology and techniques that lower the adoption barrier to software product lines (see www.biglever.com, www.esi.es/Projects/Reuse/projects.html, and www.iese.fhg.de/Business_Areas/Product_Line_Development).

These new approaches offer two things not found in the epic proactive software product line approaches. The first is lightweight technologies and techniques that specifically support software product line engineering. The second is using a variety of adoption models for establishing and operating a software product line practice.

Lightweight software product line technologies and techniques minimize the paradigm shift between conventional software

continued on page 4

Although these epics make for great, inspiring reading, many software organizations need to operate on a more predictable and less dramatic story line.

point

continued from page 2

could build, and would be willing to build, under the auspices of its product line capability. Systems outside the doughnut represent those that are out of scope, that the product line is not equipped to handle well. Systems on the doughnut itself could be handled with some effort, but require case-by-case disposition as they arise. In a product line of office automation systems, a product with a conference room scheduler would be in, but one with a flight simulator would be out. One with a specialized intranet search engine might be in if it could be produced in a reasonable time and if there were strategic reasons for doing so (such as the likelihood that future customers would want a similar product).

Explicitly scoping the product line lets us examine regions in the neighborhood that are underrepresented by actual products in the marketplace, make small extensions to the product line, and move quickly to fill the gap. In short, a consciously preplanned, proactive product line scope helps organizations take charge of their own fate. The scope feeds other product line artifacts; the requirements, architecture, and components all take their cues for the variabilities they need to provide from the scope statement.

Putting an organization on the same strategic page requires vision, strong management, technical competence, process discipline, and no small amount of dedicated leadership. But the payoffs can be spectacular, as companies large and small in all domains are discovering. Help is available. The Software Engineering Institute's product line practice framework (www.sei.cmu.edu/plp) describes how to extend software engineering and managerial practices from one-system-at-a-time product building to make them apply to product line engineering. The growing body of literature and case studies also provide invaluable guidance for practitioners who want to adopt the approach. Together, we are taking product lines into the realm where organizations can be proactive about the systems they are prepared to build.

References

1. L. Brownsword and P. Clements, *A Case Study in Successful Product Line Development*, tech. report SEI/CMU 96-TR-016, Carnegie Mellon Univ., Software Eng. Inst., Pittsburgh, 1996.
2. P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*, Addison-Wesley Longman, Reading, Mass., 2001.

Paul Clements is a senior member of the technical staff at the Software Engineering Institute, Carnegie Mellon University. His technical interests include product line systems, software architecture, software design, and product line practice. He received his PhD in computer sciences from the University of Texas at Austin. Contact him at SEI, 4500 Fifth Ave., Pittsburgh, PA 15213; clements@sei.cmu.edu.

counterpoint

continued from page 3

engineering and software product line engineering. They let organizations reuse their existing software, tools, people, organization charts, and processes.

The variety of adoption models let organizations select one or more strategies that best meet their business objectives, engineering realities, and management styles. The three prominent adoption models are *proactive*, *reactive*, and *extractive*.

The proactive approach to software product lines is like the waterfall approach to conventional software. You analyze, architect, design, and implement all product variations on the foreseeable horizon up front. This approach might suit organizations that can predict their product line requirements well into the future and that have the time and resources for a long waterfall development cycle.

The reactive approach is like the spiral or extreme programming approach to conventional software. You analyze, architect, design, and implement one or several product variations on each development spiral. This approach works in situations where you cannot predict the requirements for product variations well in advance or where organizations must maintain aggressive production schedules with few additional resources during the transition to a product line approach.

The extractive approach reuses one or more existing software products for the product line's initial baseline. To be an effective choice, the extractive approach requires lightweight software product line technology and techniques that can reuse existing software without much reengineering. This approach is very effective for an organization that wants to quickly transition from conventional to software product line engineering.

The combination of lightweight technologies and techniques along with the variety of adoption models offers a dramatic reduction in the adoption barrier. For example, Salion, an enterprise software producer, needed to transition from conventional one-of-a-kind software engineering to software product line engineering.[1] Based on time and cost constraints, an epic proactive transition was out of the question. So, it adopted lightweight software product line technology from BigLever Software,[2] an extractive approach to reuse existing conventional product as the baseline for the product line and a reactive approach to implement unanticipated requirements from new customers. While maintaining its aggressive production schedule, Salion transitioned to a live software product line in about four person-months of total effort, which was less than 5 percent of the time required to build the conventional product used as the product line's baseline.

It has been said that "the right point of view is worth 20 points of IQ." That is certainly the approach we need to take in moving software product line practice from the realm of epic adventures to mainstream software practice. New advances in technology and methodology show that, by taking the right viewpoint, the adoption barrier disappears.

References

1. P. Clements and L. Northrop, *Salion, Inc.: A Case Study in Successful Product Line Practice*, tech. report to appear in 2002, Carnegie Mellon Univ., Software Eng. Inst.
2. C. Krueger, "Easing the Transition to Software Mass Customization," *Proc. 4th Int'l Workshop Software Product Family Eng.*, Springer Verlag, New York, 2001, pp. 282–293.

Charles Krueger is the founder and CEO of BigLever Software. His technical interests are in technologies and techniques that bring software product lines into mainstream software engineering practice. He received his PhD in computer science from Carnegie Mellon University. Contact him at BigLever Software, 10500 Laurel Hill Cove, Austin, TX 78730; ckrueger@biglever.com.

Paul Responds

Krueger's identification of adoption models—proactive, reactive, extractive—is a first-rate contribution to this field, as is his missionary work for low-cost adoption methods. But with the zeal of some missionaries, I think he's a little quick to spot the devil lurking about—in this case, hiding in the proactive approach.

First, the proactive approach does not require a halt in production. At Cummins, the new product line manager called a halt because the projects that were underway were running in different directions and would clearly not be able to deliver the large number of products to which the company had already committed. Turning the whole organization to the product line approach was its salvation, because it was on the road to major failure anyway. Here, the risk was in not taking decisive action.

Second, proactive adoption does not mean unanimous simultaneous adoption. In our book *Software Product Lines: Practices and Patterns* (Addison-Wesley, 2001), Linda Northrop and I write extensively about how an organization can launch pilot projects to introduce the concepts, demonstrate measurable benefit, iron out process and organizational details, and let other projects climb aboard when ready.

Finally, we read that in the proactive world "all product variations on the foreseeable horizon are analyzed, architected, designed, and implemented up front." Well, when you would not take the foreseeable horizon into account? Granted, in some environments, your foreseeable horizon might not be very broad. But shouldn't you still plan for the variations you know are coming? If you don't, your architecture (among other things) might simply not be up to the task. If your next customer wants a version of your basic product that runs 10 times as fast and supports 100 times the users, good luck achieving that by just iterating on your current inventory.

Being proactive simply means actively gathering what you know about your customers and your application area and using that knowledge as much as you can to plan for the future and, in some cases, to manage that future to your advantage.

Charles Responds

One of the insightful segments in the Clements and Northrop book on software product lines is a sidebar entitled *E Pluribus Unum* (Latin for "out of many, one"). There, Clements skillfully articulates how organizations who have mastered software product line engineering think of themselves as building a singular software "system"—the product line—rather than as building multiple software products. What this suggests to me is that all the issues that the software industry has explored for engineering singular one-of-a-kind software systems will be explored again for engineering singular software product lines.

The argument for or against proactive approaches to software product line engineering resembles the arguments for or against waterfall approaches to one-of-a-kind software engineering. Rather than be dogmatic about any particular approach, I prefer to keep a collection of approaches available in my toolbox. After exploring business conditions, knowledge of the domain, clarity and stability of the customer requirements, architectural complexity, likelihood of building the "wrong" system, available time and resources, and so forth, I can go to the toolbox to select the tool that best solves the problem.

The issue here, I believe, is whether proactive should be the only tool in my toolbox rather than whether it's a good approach. For Cummins, proactive was likely the most effective approach. For Salion, a combination of extractive and reactive approaches fit their business conditions perfectly.

For the mainstream software engineering community to embrace software product lines, the adoption barrier must be much lower than that experienced by the early pioneers. Providing lightweight technology and techniques plus a variety of adoption models will go a long way toward enabling the entire software industry to capitalize on the order-of-magnitude improvements offered by software product lines.