

Software Product Lines: Marathon Man

Paul Clements

Product line champions emerge where you find them, but one person who absolutely has to have a clear, coherent vision for the product line is the product line architect. And beyond having the vision, he or she must be unwavering in communicating the vision in the face of reactions that may well range from apathy to hostility. Why hostility? Because for pockets in the organization that are resistant to change, the architect will be seen as the instigator of that change. Management may issue dictums and platitudes about how the product line approach will be good for the enterprise – and who could oppose shorter time to market and a fatter bottom line? – but the product line architecture is often the first sign to the product developers that things have changed.

A sure sign of trouble in an organization striving to produce a product line is that the role of product line architect is unfilled. A second sign of trouble is when that role is filled but its occupant has no nominal authority over the architecture of the product line. And a third sign of trouble is when the role is filled, the occupant has titular authority, but no one listens to him or her.

This is about one organization that went through all three of these phases. The first phase was preceded by a period in which a loose committee was formed to create a product line architecture. The committee had a chairperson who was a gifted designer, but – due to circumstances beyond his control – lacked the organizational presence to make his vision stick outside the committee (where it was needed to produce products). He, and management, needed to understand that his job didn't end when the architecture was drawn up. What he produced was essentially a re-structuring of the organization's application libraries that product-builders largely weren't using anyway. Management was unenthusiastic about the efforts and disbanded the committee, which it had failed to sufficiently empower in the first place.

This organization still believed in the concept of software product lines and tried again. Its second attempt fared better. Another individual was appointed to the job of designing a minimal product line architecture, which amounted to a set of common platform extensions that all products would be required to use. The goal was simple hardware portability. But unlike his predecessor, this architect grasped the big picture by understanding immediately that the platform extensions alone, while better than nothing, would not achieve the product line vision. He lobbied hard for a single architecture for every product in the product line, whereas at the time of his appointment a product team's only obligation was to use the common extensions in whatever manner it wished. This architect grasped the essential difference between a true product line and products built with reuse. With a common architecture, products could become interoperable (a rich new capability they lacked before) and exploit application-level commonality among themselves.

But first, the extended platform had to be designed. He formed a design committee, consisting of a few of the best designers in the domain plus representatives from several of the key product teams. Buy in was always in short supply and foremost on the architect's mind. And over the course of several months,

they produced a detailed design of that platform *that also included a picture of how a standard product that used the platform would be structured*. The architecture was, in its simplest form, layered; the top-most layer (L1) was product specific; the second layer (L2) was composed of software generic to each of the half-dozen or so classes of products that would populate the product line. That is, a product would incorporate one of several available L2 layers, depending on what kind of product it was. Lower layers formed the common platform. Well-defined interfaces and interaction mechanisms connected all the layers. (This is a standard scheme for a product line's software architecture.) The architect began to describe the architecture to management and to the product teams, always explaining the benefits of the vision. But he was also careful never to let the vision interfere with current work – after all, he would not be able to achieve that vision if he were removed from his position because short-term deadlines were not met.

Buy in came gradually and sometimes not at all. A gain in authority here means a loss of authority over there, and those giving it up did not do so easily. Some members of the design committee, instead of assuming ownership of the architecture and becoming its proponents in their home groups, seemed to view their role as reporting to their home groups “what that wacky group was up to now.” Our architect realized that his committee was not the group of proponents he needed them to be. He knew that some members were even telling their home groups that they didn't understand the new product line architecture – even though they had helped to craft it. He was dealing with a severe case of organizational inertia and tried to remedy it. In a series of meetings and e-mail messages, he tried to impress on the committee and management the importance of the unified architectural vision. “If you feel we should not have one architectural model for the overall product line,” he wrote, “please contact me.” In other words, either get on board or step off the train now. “As a group,” he wrote to the committee, “we need to be comfortable and confident with the architecture. I am leading this effort but not dictating it. However I am adamant about having a single architecture (for the whole product line) that we all support and understand. We need to know the architecture like the back of our hands if it is to work. We created it and we will be the ones who understand it, teach it to others, and grow it.”

Eventually, the overall architecture was embraced by management in the form of a reorganization whose work units reflected the structure of the architecture: the platform group and the project groups were now joined by a group responsible for turning out the L2 layer. But management omitted an important component when it failed to appoint a single individual to have overall architectural authority over products. Conversely, the L2 group was not clearly chartered, and confusion was the result. The new head of the L2 group wrote our architect saying “Our group is starting to think about how to design this thing called L2. Our group seems to be the best place to start to think about such things. I'd like to make a presentation that shows how this work may fit into the overall architecture work being done by the design committee. I'd like to include a slide of the overall architecture in a presentation. Do you have one I could use?”

Thing? Your group seems to be the best place? Start to think? This work may fit? Our architect nearly *threw* a fit, and although his reply was composed, the message was clear: “The work you are describing is overall architecture work that is in the purview of the design committee.” And he appended a copy of

the design committee's scope to gently refresh the L2 leader's memory, and pointedly invited him to the next design committee meeting.

The reorganization should have been the final action in the adoption of the overall product line architecture. In fact, the entire effort nearly failed at about the same time. Delivery pressure from some of the product groups caused management to temporarily embrace an expeditious but purely short-term approach: use as much code as possible from the current product set and get the products out the door. Later products could follow the product line approach. And who better to lead this effort than a man of proven accomplishments? Our architect was told to delay (or wrap up) his work on the architecture and begin porting code.

This was not the first time this organization had faced a crucible like this; schedule pressures had previously been used as an excuse to delay implementation of product line practices. Our architect knew that there would always be schedule pressures, and he knew that a true product line capability could never emerge in fits and starts. Rather than openly rebel against his management, to no good end, he saluted smartly and prepared for his new assignment. In a message to the design committee, product groups, and management, he accepted his new assignment with good grace but made sure the architectural vision did not die.

“As some of you have gathered,” he wrote, “my project scope has narrowed considerably with respect to ownership of the overall product line software architecture definition. To this point in time I have been the lead architect. A meeting was held recently in which I was requested to transfer the responsibility of the overall product line architecture to my replacement. I wish him the best of luck and will implement his vision as it becomes defined. Along with leading the design committee, the following responsibilities accompany the lead architect role and will from this point in time be transferred to him: (a) lead design meetings; (b) develop and maintain the product line architecture and model; (c) communicate the architecture; (d) develop a training class for the architecture; (e) define the technology roadmap of the architecture; and (f) schedule and release dates of architecture. From this point on all questions pertaining to these issues should be addressed to my replacement. To the best of my ability I will help him transition into his new role.”

Later, he explained his gambit to me. “As you can probably tell by the email flying around,” he wrote, “I have been doing my best to keep my company on one architectural path. I have been in extensive meetings with [upper management] and the [middle-level] supervisors. I could write a book alone on what has transpired in the last couple of weeks here! It all comes down to how we are to migrate to a complete product line and the fact that it does take several years to reach Nirvana. But as we migrate we need to spit out applications at various stages along the way. The first of these junctures was just encountered in this new generation. Several non-product-line proposals were flying around such that the architecture would become a splintered, ugly monster that would remain forever. It got to the point where I was completely out of architecture management for about the next nine months. That is why I relinquished control of the design committee. Obviously the people who wanted to split off did not want to take the baggage of architecture management with them. I wanted to make a point that someone is always responsible for

the overall architectural vision and the path to that vision. So I handed off the vision! It was at that point that I believe the lights in the managers' heads started turning on as to what they were about to do. I believe we are back on line and I will explain more in detail as we unveil the plan. *The plan gives me technical responsibility to get the architecture done through all the layers for the pilots.*"

Our architect had finally brought his organization through the three signs of trouble I described back at the beginning. The victory was not achieved overnight. But through it all, he persevered in keeping his unerring vision for a single architecture – and a single source of architectural authority – for the entire software product line.

I have learned that this architect is a marathon runner in his non-professional life. This did not surprise me. Although personally I find the thought of running 26 miles horrifying, I understand marathon runners to be driven, disciplined, goal-oriented people who can see the future clearly, don't mind some temporary pain, and who actually do their best when they hit what they call "the wall," the point at which no sane person could possibly take another step. Clearly this architect had exactly the right qualities he needed to succeed in his task at this organization.

About the Author

Dr. Paul Clements is a senior member of the technical staff at Carnegie Mellon University's Software Engineering Institute, where he has worked for 8 years leading or co-leading projects in software product line engineering and software architecture documentation and analysis.

Clements is the co-author of three practitioner-oriented books about software architecture: *Software Architecture in Practice* (1998, second edition due in late 2002), *Evaluating Software Architectures: Methods and Case Studies* (2001), and *Documenting Software Architectures: View and Beyond* (2002). He also co-wrote *Software Product Lines: Practices and Patterns* (2001), and was co-author and editor of *Constructing Superior Software* (1999). In addition, Clements has also authored dozens of papers in software engineering reflecting his long-standing interest in the design and specification of challenging software systems.

He received a B.S. in mathematical sciences in 1977 and an M.S. in computer science in 1980, both from the University of North Carolina at Chapel Hill. He received a Ph.D. in computer sciences from the University of Texas at Austin in 1994.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate

further discussion about this topic.

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.