

The Man with the Plan

Paul Clements

One of the crucial aspects of getting a software product line production capability up and running is to synchronize the production of the core assets with the production schedules of the products that need to use them. At the heart of the problem is knowing which core assets need to be produced first, and channeling all of the product-builders' needs and demands through a disciplined process that results in a prioritization that is optimal for the entire product line, as opposed to any one product. For this process to work, planning is essential.

In October of 1999 I went to visit the software core asset manager for a product line effort we were helping to get off the ground. It had been a few months since my last visit, and in the interim the product line effort had undertaken a massive and detailed planning effort. The manager wanted to show me the results. "I have to give my manager credit," he said, referring to the overall product line manager. He began unrolling a cylinder of paper over a conference table, spreading out a sheet roughly the size of a queen-size bed, covered with an enormous PERT chart. "He asked me if it would be worthwhile to hire a management consultant to help us build this project plan for the core assets and keep it up to date. I said 'Sure, why not?' Since then, the guy has worked full time on-site for a week every other week for several months."

A full-time management consultant hired to help with planning sounded expensive, if not extravagant. Had it paid off?

"I can't believe what a difference it's made," he said. "Just this week, I've had four or five product managers come to me and say they needed to have a particular task moved over here" – with a sweeping motion of his hand he indicated the left side of the huge chart, home to the earliest tasks—"and they needed it now." He smiled—rather serenely, I noticed. "I say, 'Well, there are about 2,000 tasks in the plan. Yours is one. We can move it, but it's going to affect everything else. We'll have to see if that's what management wants.'"

This resonated immediately. Almost three years earlier we had performed a risk evaluation at this organization, and one of the most critical risks to their ambitions of launching a product line was lack of planning. Project engineers intent on getting a particular product out the door as soon as possible would make heated demands on the newly-formed, embryonic, and hopelessly over-worked core asset team. When they balked, the project engineer could point to a deadline, whether real or invented, whereas the core asset team could point to nothing. One of the risk evaluation participants lamented, "We never had enough information to be able to justify a 'no' answer."

Not that “no” was anyone’s goal. But some well-placed “no”s will keep an organization focused on the long-term product line goals and not let them get trodden over by short-term single-project concerns. Learning how and when to say “no” is a big part of product line culture.

And that wasn’t all. The manager went on to explain that his consultant would visit the product development, systems engineering, and hardware engineering groups to understand the inter-group dependencies and account for them in the core asset software plan. Groups who had poor plans of their own could not justify the arbitrary deadlines they once were able to impose on the core asset group. The message was “When you understand what you really need and why you need it, come see us and we’ll see what we can do to help you. Until then, poor planning on your part does not constitute an emergency on our part.” It was a message that was never delivered explicitly; it never had to be.

As we went through the plan, I noticed something I hadn’t heard before. He told me he thought that the software core assets were going to be ready sometime in 2001.

“Really?” I said. “The last I heard, you were on the hook to deliver those this spring.”

He nodded. “April 2000,” he agreed. That was only six months hence.

“And now not until 2001?”

“Yep.”

“What happened?”

With another sweep of his hand, he indicated the leftmost edge of his furniture-sized PERT chart, where there dwelled a column of blue boxes indicating dependencies on external projects such as the systems engineering group or the hardware selection group. “These won’t be ready in time,” he said, trying very hard to suppress a grin.

I thought a minute. “There’s no way that you would have been allowed to slip your schedule almost a year without this plan, is there?”

“Nope,” he said, grin no longer suppressed. “Now we’re going to get to do it right.”

The point, as we both knew, is not that a year delay in the product line was a thing to be smug about. But since the dependencies really did exist, the delays would have occurred anyway. But they would have occurred maybe a month at a time, each time precipitating unpleasant surprise, finger-pointing, hand-wringing, disappointment, and (more than likely) management pressure that intensified each time, to no good end. The plan had bought the core asset group the time it

would have had anyway, but without planning that time would have arrived in jerky four-week chunks and not have been used to best advantage.

If the truth will set you free, then I was looking at a very liberated software core asset manager.

Besides uncovering the truth, planning had another ironic side effect. Considered a way to let the core asset group to say “no,” planning now let them avoid having to. The message is one of cooperation, not confrontation:

- “We’ll happily move your task up—after all, we’re here to serve—but here are the ramifications. Let’s see if the product line manager agrees with those effects wrought on other development groups and the product line itself.”
- “We’ll happily move your task up. Show me in your plan where it feeds in to a critical path so we can agree on the timing.”
- “We’re all ready to produce the core assets in the amount of time we promised, but we can’t make progress until we get the following information from the following groups.”

“That consultant makes a lot of money,” the manager volunteered, rolling up the chart. I knew the manager was talking in terms of a per-hour rate, and consultants being consultants, I had no doubt that the rate was breathtaking. But if we were measuring the serenity that this information imparted to the core asset group, a serenity that I had not observed heretofore in this manager, then I got the distinct impression that the manager thought the consultant was not overpaid at all. I was inclined to agree.

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.

© Capability Maturity Model, Capability Maturity Modeling, Carnegie Mellon, CERT, CERT Coordination Center, CMM, CMMI, and OCTAVE are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

SM Architecture Tradeoff Analysis Method; ATAM; CMM Integration; COTS Usage Risk Evaluation; CURE; EPIC; Evolutionary Process for Integrating COTS-Based Systems; Framework for Software Product Line Practice; IDEAL; Interim Profile; OAR; Operationally Critical Threat, Asset, and Vulnerability Evaluation; Options Analysis for Reengineering; Personal Software Process; PLTP; Product Line Technical Probe; PSP; SCAMPI; SCAMPI Lead Assessor; SCAMPI Lead Appraiser; SCE; SEI; SEI-Europe; SEPG; Team Software Process; and TSP are service marks of Carnegie Mellon University..

TM Simplex is a trademark of Carnegie Mellon University.