Feb, 2011

How An Innovative Semat Kernel Can Bring Real Improvements And Cost Savings To Industry

Semat Status

Some of you might be wondering what is going on with Semat (www.semat.org). In October it was announced that part of the Semat work would move to the Object Management Group (OMG, http://www.omg.org/) to take advantage of their open, transparent governance model. The move to the OMG is taking place as planned, and many of us who have been involved in Semat during 2010 are supporting that effort. There has also been a team formed from the original Semat working group members that is moving forward in preparation to respond to the expected RFP from the OMG. The RFP is anticipated to ask for a software engineering kernel and a supporting kernel language.

Part of our goal in supporting this effort is to break some of the heaviness that earlier OMG efforts have brought to the table. As we work with the OMG we are insisting that the language and kernel must be lightweight and supporting an agile way of working.

The subject of the kernel and kernel language brings us to the purpose of this article. When I spoke at the Rochester Institute of Technology (RIT) (http://www.gccis.rit.edu/how-semat-can-change-future-software-engineering-personal-reflection) on the subject of Semat last October, at the conclusion of my talk one of the first questions I received from a software engineering professor related to why we needed a language to express software processes when the Software & Systems Process Engineering Meta-model Specification (SPEM) already existed.

I answered this question by stating that Semat wasn't trying to compete with SPEM, or any other previous software initiative. Semat's goals are different. For example, a primary goal of Semat includes a kernel of widely agreed essential elements supporting the needs of industry, academia and practitioners. Semat seeks to separate these essentials from the specific needs of each organization and project. SPEM does not have

such a kernel. Semat also seeks a solution with a greater focus on enactment than past approaches. This means the solution must do a much better job of supporting the practitioner's day to day activities. To fully understand the importance of this goal, you have to first understand the problem we face today in the Software Engineering community.

The Problem The Software Engineering Community Faces Today

The traditional way often used to describe software processes or practices includes a flow diagram demonstrating a sequence of steps (activities), resulting in the production of artifacts, such as a software requirements specification, design document, a test plan, or a test procedure. While these process/practice representations have historically been viewed as an acceptable way to achieve the intent as identified in many process improvement frameworks, such as the CMMI, too often today we still hear from the software practitioners in the trenches that these process descriptions don't really help them with the real job they face each day.

When you dig a little deeper asking more in depth questions about that real job they face each day, and you listen to those practitioners describe what their real job specifically entails, it isn't hard to understand why these process descriptions fail to help people where they need help most.

Digging a Little Deeper

While Semat seeks to separate essentials from specifics, the problem we face goes deeper. Many organizations today need help figuring out what information really should be captured in their "specifics". They need to know what their people really need help with to get their job done effectively. It has been my experience that this important subject too often gets glossed over. We don't pay enough attention to where practitioners make their biggest mistakes.

For example, some may need help in estimating because they tend to be overly optimistic. Some may need help assessing risks. Some may need help in determining what to do when they run into a certain type of problem. The reason this isn't easily

solved is because these examples I just provided may be issues in one organization, but in another organization the critical issues may be completely different.

The key issues that need attention the most differ in different organizations, and they differ in the same organization over time. When I asked Ivar Jacobson for his thoughts he said these are what we would call extension practices. He also said that the difficulty is that you need to know these "small fragments beforehand." And he followed that thought by saying if you did have a way to find them beforehand, "it would be great if we could hang these fragments on some kind of hook."

It is worth noting here that this problem is not unique to the Software Engineering Community. As an example, similar findings are being discovered in today's processes within the health industry. The common thread is "knowledge work", a term coined by Peter Drucker over 50 years ago in his book, "Landmarks of Tomorrow" (1957). Knowledge work refers to "work done in the workers head, rather than with their hands" and there exist other initiatives underway, besides Semat, to help address this problem. As an example, refer to work in "Adaptive Case Management" . [1]

At this point I would like to step back and ask a question. Could there be a better way to help software practitioners with the real issues they face each day on the job? More specifically, is there a way to give the software practitioner the *specific* help they need right when they need it most? Can we, as Ivar refers to, know these small fragments beforehand, and might it be possible to, in fact, "hang them on a hook" so practitioners could access them, right when they need them? And I want to highlight here *specific* and *small* because what we are hearing, particularly from industry, is that general and heavyweight guidance, is not cutting it.

So What Is Semat Trying To Do About This Problem?

Much has been written about what Semat is and isn't trying to do. But one of the more exciting ideas is that *practices*– in the Semat context– are all about what people do, not just descriptions of what we would like them to do. This—in the words of Ivar Jacobson—turns the idea of a "practice" upside down. This idea opens up great possibilities. So lets explore what it could actually mean with respect to how our practices will exist in the

future—beyond just simple descriptions– and how they might help people with the real tasks they face each day.

A Different Way To Think About Practices: Upside Down

Traditional process representations have been static and impersonal. They focus on activities and artifacts, and while some do make an effort to identify roles, and skills, and responsibilities, the people side has been largely missed. This is not anything new. At the Semat Kickoff workshop in Zurich last March Larry Constantine told us he and Ed Yourdon knew they had missed the people side from their Structured Analysis work in the 1970's. And it has continued to be missed ever since, including much of the current process representations that are used today.

But what if we actually did turn the way we viewed *practices* upside down? In other words, what if we switched the focus of practice from "things" to people? In particular, the people who must use software practices—the software practitioners. Now, think about the possibilities.

This is not a completely new idea, and it is one that I have already found has proven useful in my own consulting engagements over the past few years. When I hear those familiar words from a client – "the company processes don't help me do my job,"—I typically have done the following:

First, I dig deeper to find more *specifics* related to what that individual means. I ask, "could you tell me a little more about your job?" And as they explain their job, I keep digging by asking, "what are the tough issues you typically face each day where you need more help?"

Usually, I find that people talk easily about their job and can give me all kinds of situations that come up that they must face where they need help, but their documented company process descriptions give them no help at all.

After talking to a number of people in a given organization I often observe *common patterns* emerging from these discussions. I then take the data from these interviews

and produce a set of simple *scenarios* that represent these common repeating patterns in that organization. The scenarios I develop often demonstrate multiple possible *decisions* and possible consequences of each choice. You could think of these scenarios as extension practices specific to a given organization. You could also think of these scenarios as including more of the "how to" implement the practice, whereas the practice itself often just focuses on telling you "what" you must do. Viewing it this way you can begin to see how such scenarios can really help the practitioner because it is the "how to" that they need when they are in the trenches doing their day to day work. We package these scenarios up and use them as part of the training material when we deploy the new processes in the organization.

What I have found is that because the scenarios are real to that organization, people relate to them and find them more helpful than static process flows. This became so popular in one of my client organizations that I was asked to build more scenarios and then make them available on–line through their intranet. This allowed the software practitioners to access the scenarios "just-in-time" during the "heat of battle" of doing their job, to refresh themselves on their options, and potential consequences, when faced with a given situation.[2, 3]

This is a perfect example of what I believe Semat is trying to achieve when we say practices are what we do, not just descriptions of what we would like people to do.

It is also worth noting that Watts Humphrey has observed in his work a similar need for more "how to" guidance for the practitioner. He has stated that when using the CMM/CMMI ® it helps with the "what" you must do, but doesn't help enough with the "how" to do it. [4]

How The Semat Kernel Language Can Help Solve The Problem

At RIT when I spoke about Semat last October to help people understand what the Semat kernel language was about I made the following analogy: I stated that the Unified Modeling Language (UML) is a modeling language for software. The Kernel Language will give us a way to express the way we work when we develop software. But if this language is to help people with the real issues they face each day on the job, it must be

able to represent those issues that can help most, and be usable by software practitioners in the "heat of battle" of their job. Earlier efforts have largely failed here.

So if Semat is to be successful, what would such a language look like?

First, because people need information quickly when they are in the heat of battle of their job, the language must be flexible and capable of providing just the information a software practitioner needs when they need it. This ties into an idea that has been discussed on Semat and is part of the vision going forward—*separation of concerns*.

When our concerns are not appropriately separated supporting the needs of the individual we create frustration, inefficiency, and we reduce our competitiveness. The software practitioner needs what they need, when they need it, and no more. And when they need it, they must get it in a form that is usable. This is not to say they don't have responsibilities to interact with key stakeholders, but only to state they only need to know what they need to know.

How can this be achieved?

I suggest that a similar idea that has already proven successful can work for practice representations as well—multiple views where each view focuses on a specific software practitioner role.

Whenever we model– or represent what we do in some tangible way– multiple views can be useful. So why not represent our practices from the viewpoint of those who must use those practices making sure to only provide the information that person needs at that point in time? And from my own experience with my clients, the more *specific* we can make these views and the more "how to" information we can provide, the better it becomes as an aid to help with the real issues faced each day.

Now let me be clear here. When I say "from the viewpoint of those who must use those practices" I mean software practitioners. This includes developers, managers, and testers. Each has their own viewpoint, with their own set of *specific* "how to" issues they commonly face each day.

It must be understood that the scenarios that I have found helped my clients in the past were very *specific* to roles within a given organization, and the "how to" solutions were very specific to the culture and environment in that organization. Therefore such scenarios would never be part of the kernel or the kernel language itself. That is, unless we as a software community can come to a common agreement that some of these are patterns that do indeed repeat across all software engineering efforts—at least at some level. This is still part of the challenge Semat seeks to solve as we move forward.

The *specific scenarios* unique to specific software practitioner roles in each organization would be developed by extending the kernel using the kernel language. Through such real scenarios that are developed from the real experiences of the people in your organization– using the kernel and the kernel language– we could have communities of not just shared best practices, but shared "best practice scenarios". This could make static practices a thing of the past. Our software practices would come alive! And the goal of a practice being what we do, not just what we would like people to do would be realized.

How Semat Can Bring Real Improvements and Cost Savings

Everyone is looking for innovate approaches to save cost and improve performance, especially during economically challenging times. Training is a critical element organizations need to provide to keep their workforce competent and remain competitive. However, historically it has been expensive with training costs continuing to rise.

Static process descriptions can never replace training, because they can never answer all the questions a software practitioner will have, nor can those descriptions provide the needed coaching and course corrections required when a software practitioner is faced with a new situation.

This brings us back to the issue Ivar Jacobson raised. The difficulty is finding those "small fragments beforehand."

It has been my experience that the best practitioners inside organizations usually know what the key specific issues are for their organization at any given time and they know "how-to" handle those issues when they arise. But what they too often don't have is an effective mechanism to communicate this critical knowledge to their fellow practitioners so they too know just what they need to know when they need to know it.

In the past efforts to capture this information have failed for multiple reasons, one being because we have failed to separate today's critical issues from yesterday's and we have failed to separate common essentials from "*specifics*". When we try to list in our practices all the possible issues that could possibly arise it overwhelms us and processes become unusable. This is the lesson from past heavyweight process approaches that have failed and continue to fail in many organizations today.

The key to the future is to learn to identify that "*small"* set of "*specifics*" that are critical in your particular organization right now. Then find a way to express them in an easy to understand way using a common language that includes a common agreed to set of underlying essentials. This is not to say ignore all else, as this would lead us back to chaos. What must be understood is that this is a matter of appropriate emphasis.

Organizations that survive in the future will be those who find innovative solutions to meet critical needs, such as learning how to systematically handle their current critical issues and communicating in a timely way expectations to those who need to know throughout their organization. By creating a language that can be used in real time by software practitioners in an environment where they can access "just-in-time" the information they need most when they need it, we can bring real improvements and cost savings to industry.

We envision here an innovative Semat kernel and kernel language that is easy to use by practitioners with easily accessible "hooks" to just the *specific* and *small* fragments of information they need when they need it. The organizations that survive in the future will be those who embrace such innovations helping them maintain high performance and a critical competitive edge.

Where We Go From Here

The Semat team that is continuing outside of the OMG effort is currently looking at options to gain broader involvement of the software engineering community in 2011. We want to get feedback from potential users on our kernel and planned kernel language. One possible activity that could be occurring at selected industry sites is looking at an organization's software practitioner common scenarios and "how to" solutions to see how they could potentially be brought to life in real time through the kernel language. We are still considering our options, and we are interested in hearing your thoughts and ideas. We are also interested in hearing from those in the community who are interested in becoming more involved in the exciting work of Semat. Please provide your thoughts and feedback through this blog site, or through the Semat blog (http://sematblog.wordpress.com/).

Looking forward to hearing from you!

[1] "Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done"

[2] For more information on services helping organizations develop scenarios for "just in time" training refer to (www.pemsystems.com). Follow the links to the "Training Products and Services" page.

[3] For more information on Paul McMahon's client cases from which he draws common patterns leading to the *simple scenarios*, refer to his book, "Integrating CMMI and Agile Development: Case Studies and Proven Techniques for Faster Performance Improvement," Addison-Wesley, 2010

[4] "Why Can't We Manage Large Projects?", Watts Humphrey, http://www.crosstalkonline.org/storage/issue-archives/2010/201007/201007-0-Issue.pdf