July 5, 2010

The Hidden Potential Value of Semat

Recently there has been significant discussion on the Semat tracks surrounding the term "alpha." This term is new to many people, but the idea behind an alpha is not really new. Ivar Jacobson has stated that Ericsson has used "alphas" for more than 50 years. Although it was not referred to as "alpha" I have observed a similar concept used within both successful product-focused organizations, and within successful industry-wide process improvement initiatives. I will provide more information on these examples later.

When I first heard the term alpha being discussed within Semat working groups, I was unsure if it was essential to Semat. Although the precise definition of an "alpha" is still evolving, I now believe it is inside this alpha concept where the greatest hidden potential value of Semat lies.

To help explain one first needs to understand the idea of a Semat Universal, how the Universals are being developed on the Universals Track, and how "alphas" can bring critical value beyond the simpler Universals.

I am currently leading the Assessment Track of Semat supporting Watts Humphrey and I am participating actively on the Requirements Track, the Universals Track, and the Kernel Language track. As of this writing in early July, 2010 we have developed about six Universals on the Universals Track. You can think of a Universal as something that is essential to all software engineering efforts. For a Universal to pass the test of being a Universal it must meet the following criteria:

- Universal/Essential;
- Relevant;
- Actionable;
- Assessible;
- Defined Precisely.

A fundamental goal of Semat is to find those elements that we all agree are indeed Universal so we can stop the continual reinvention that occurs when new fads or new software engineering methodologies arise. But just finding those "universal" elements alone will not be enough to make the Semat initiative a success.

Semat will not provide a methodology. It will provide a kernel that contains the agreed to Universals, a kernel language that will support users in "combining" Universals in a well-defined way, and precise agreed-to related definitions. Another fundamental goal of Semat is to help users both define new practices and methods and compare existing practices and methods using the kernel to help them make more effective decisions related to their software engineering work.

Nevertheless, if you look at the first few Universals that have been proposed by the Universals Track some interesting questions quickly arise. Early example candidate universals include Software System, Team, Work, and External Stakeholders.

One question that arises is: What is the value of defining these "universal" things that to many seem obvious?

To understand the answer to this question we need to discuss the concept of an "alpha", how an "alpha" relates to Universals, and the added potential value alphas can provide. Let's start by looking a little closer at a candidate Universal.

A Candidate Universal

Software System is one of the candidate Universals being proposed by the Universal Track. Its current proposed description is:

A software system is a system made up of software, hardware and digital information that provides its primary value by the execution of the software.

The Software System candidate has been verified by the Universals track as meeting the Universal criteria identified above.

What Is An Alpha?

A currently proposed (but evolving) criteria to determine if something is an Alpha is the following:

- The candidate has a Name
- It has a set of associated or referenced work products
- It has a well-defined set of states
- It has a completion criteria
- Optionally, it contains sub-alphas

Verifying if a Universal is an Alpha Through an Example

As an example assume you are a potential Semat user who is using a Component Development methodology on your project. It can easily be verified that the candidate universal "software system" is a potential "alpha" by using this criteria as follows:

- The candidate has a Name which is Software System
- It has a set of associated work products which could be your code, test cases, design artifacts, and requirements artifacts
- It has well-defined states which could be based on your specific project agreed to life cycle phases
- It has a well defined completion criteria which could be based on your specific project agreed to completion criteria for your specific work products
- It has sub-alphas. Each individual component in your project architecture can also readily be seen to meet this criteria

Reasoning Process To Determine If a Universal Is an Alpha

We have verified that the Software System Universal is a candidate Alpha. But are all Universals valid Alpha candidates? Let's consider another candidate Universal "Team". "Team" has also been proposed by the Universal track as a Universal. Its current proposed description is:

The set of people actively engaged in the development and delivery of a specific software system.

If you run through the criteria for an alpha with the "Team" Universal you could reach the conclusion that it is not a good candidate to be alpha. This is because you could conclude that a team doesn't have well-defined states. But under further scrutiny this may turn out to be false. Consider the following argument.

All Universals must be assessable. This is part of the criteria to be a Universal. How do we assess a team? Current work on the Assessment track is starting to develop a conceptual map for each Universal to help us understand each Universal in the context of its relationships to its attributes and to organizational objectives. With such a map we can then make intelligent decisions about what aspects of a team we should be assessing. This work is still evolving, but an early conceptual map of the Universal "Team" is leading to an increased understanding of certain attributes of a team that many believe are essential to all successful software engineering endeavors. Examples include what are currently being referred to as "coordinated-ness" and "collective mind-ness".

As these discussions continue to unfold it is quite possible that we will see that teams do indeed move through "states". It is already well known and accepted that teams undergo phases commonly referred to as storming, forming and norming. And we may find that phases or states exist related to the degrees of "coordinated-ness" and "collective mind-ness".

While this example doesn't demonstrate for certain weather the Universal "team" is an alpha, it does demonstrate a criteria and a thought process by which we can assess a

Universal, and reach a judgment. Why this is important will become evident in the following paragraphs.

Not All Alphas Are Universals

So far we have been discussing Universals and how to reason to determine if they are alphas. But could an Alpha exist that is not a Universal? To answer this question we need to go back to our criteria for Universals and Alphas.

To be a Universal, the candidate must be Essential on all Software Engineering efforts. As we consider the specifics of any software engineering endeavor we need to *tailor* our practices to fit our environment. This *tailoring* of practices leads us out of the world of Universals because the practices we modify/tailor are now specific to our project conditions and therefore will not apply universally. But it doesn't lead us out of the world of Alphas. The criteria to be an alpha doesn't include being essential on all software engineering efforts. Alphas are not necessarily universal. This is a critical point and one we will see is crucial to helping real software practitioners on real software engineering efforts.

Let me give an example of an Alpha that is not a Universal. Consider the *Software System* Alpha. Suppose our Semat User has tailored their practices to address their unique needs and we have made some project-specific decisions on the form the software requirements artifact will take. Because *Software System* is an Alpha it has a set of well-defined states. But because it is also a Universal these states must also be "universal."

However, because we tailor/extend practices to address specific project needs we may also tailor/extend the states based on these specific decisions on the form chosen to reflect those software requirements. Note that in doing so the extended/tailored version still meets the criteria for being an Alpha, but it is no longer a Universal.

Why Should You Care?

At this point you may be asking why should I care about Alphas in the first place? In other words, what is the value of the Alpha to the end Semat User?

Part of the answer to this question lies in why I say the idea behind an alpha is not really new. What I believe we are actually doing on Semat is defining precisely and institutionalizing a concept that has proven to be a critical success factor in multiple past software engineering endeavors. Let me give you three examples to demonstrate this point.

Example One: CMMI [1] Organizational Assets and Tailoring

Fundamental to the best practices within the CMMI model is the notion of a common organizational process asset library from which specific projects *tailor* their practices to the unique needs of each project. This concept of establishing common process assets across an organization along with tailoring guidelines has proven to work for many organizations that are succeeding with their software engineering efforts today around the world. The analogy I am drawing is that the Semat kernel can be likened to the Organizational common repository, and the tailoring rules can be likened to the kernel language which provides the rules for "combining" universals (or tailoring) to form practices for each project's needs.

I have long advised organizations that want to be *disciplined* and *agile* to use a "*tailoring up"* approach when developing processes using the CMMI model as a guide. This means keeping what I refer to as the "*minimum must dos*" at the common organizational level [2]. This is very much like the idea of the kernel which contains only the "essential" elements for all software engineering endeavors. You tailor up from the minimum core set we all agree everyone *"must do"* regardless of project scale or constraints.

Example Two: IBM/Rationale ClearQuest Tool

IBM/Rationale ClearQuest is a workflow tool that has "out of the box" defined states for the fundamental element referred to as an *Activity* within the tool. A number of my clients use ClearQuest, but they don't all use it the same way. Most have *tailored* the tool adding *their own states* based on their specific project environment needs. This is not uncommon for tool vendors to provide tool capabilities that are very basic, but extendible for specific user conditions.

Example Three: Successful Software Reuse Organizations

While it is popular to point to software failures, today there exists many success stories of software organizations that have reached high levels of maturity producing quality software rapidly despite facing continual changing requirements and technology. Many of these successful organizations have found ways to increase their productivity and competitiveness by automating parts of their software development through domain-specific solutions and tools which support more effective decisions by simplifying their most commonly occurring decisions.[3] This automation was made possible through their identification of repeating patterns with *well defined states* within their specific domain. For more details on two such examples see the referenced articles.[4]

Why Should You Care?

The answer to the question, "why should you care?" is because the fundamental idea underlying *"alpha"* is a proven critical factor that has been observed in multiple successful organizations in the past. These ideas work. They are not new. They have been proven over and over again in successful software engineering organizations. The alpha concept is a way for us to institutionalize these proven concepts within the broader software engineering domain—and this is really what is new and what Semat can do to provide real value to the software engineering community.

What Does Alpha Provide Beyond Fundamental Universals?

First, Universals by themselves are not very useful because they are agnostic. Alone, they are like nuts and bolts on the floor. Stating that a *team* or a *software system* is a Universal is interesting, but alone it provides no unique added-value to help Semat users do their job more effectively. This is partly because Universals are intentionally agnostic. Their value is found in the fact that they get us to that all important *common ground* from which we can begin to communicate more effectively to one another. This point should never be underestimated.

However, being agnostic can be both good and bad. It is good because it provides commonality, but it is not so good because it doesn't provide enough value alone to help people do their job on real projects. To do this we must have a way to connect what Semat is producing to the real world that our users deal with every day.

Consider again the following key elements within the criteria for an alpha:

- Associated work products
- Well defined set of states
- Completion criteria

The associated work products are specific to your project. They are currently being referred to as *"Betas"*. The well defined "set of states" still need to be worked on Semat and this is where one of our greatest challenges lie. The completion criteria still needs to be worked on Semat. What I hope the reader understands is that Semat is still in its infancy. It has great potential value, but we are only just starting.

Four Specific Values of Alpha

Let me explain four specific values that Alpha can potentially bring to the real users of Semat.

First, the Semat User doesn't have to start from a blank sheet of paper on every new project defining their methodology. Alphas can help us by saving us time and effort by allowing us to reuse from our past. We no longer have to go back to ground zero on each new project.

Second, because the alphas (some of them, not all) are also Universals, which are agnostic, they are not forcing us into any specific methodology. A fundamental requirement of the Semat kernel is its support for new practices and methods, and extending existing ones.

Third, this is not unproven theory. The concept has been proven with CMMI tailoring approaches, successful commercial tools, and in organizations that have achieve high levels of software reuse through automation. The principles on which Semat has been founded have already been proven. The added value the Semat initiative is bringing is to "abstract" these principles so they can be applied across the wider full domain of software engineering, and alpha is critical to achieving that added-value.

Fourth, the alpha concept can help the Semat end user with one of the hardest problems of many past software engineering endeavors. That is, measuring our progress more accurately.

How? The "base states" of the alphas that are part of the Universals will have "base measures" and we can reuse this history to make more accurate future estimates. We have learned that progress cannot be measured solely by measuring work products. History has proven this. Using historical data from efforts with similar practices will help us estimate more accurately in the future, but please don't mis-read this as a "silver-bullet."

Conclusion

Tailoring practices to the unique needs of each project will always be needed for effective and efficient software engineering. But we can minimize the effort this takes by minimizing the reinvention. This is fundamental to the goal of Semat. Universals can provide the "common ground" necessary to get us to a starting point. But this won't get us where we ultimately need to go for Semat to succeed.

For Semat to succeed we need to provide more value to the user that goes beyond the agnostic Universals to include agreed-to universal states that can help the User understand where they are. This is not something that will be achieved easily. We are only now beginning to find the Universals, and just beginning the discussions on which might be alphas, and what their "universal" states might be.

Following is a currently proposed definition of an "alpha." It is worth pointing out that this definition is still evolving, as it is now being discussed on the Semat Kernel Language Track. Therefore it is likely to change before the initial release of the Semat product.

"An alpha is an entity of a software organization ecosystem that reflects the team's work. Alphas have state that allows the team to track its progress, co-ordinate and plan its work, judge the effectiveness of its practices, and (most importantly) judge the effectiveness of its work. An alpha is described by a set of work products. An alpha may consist of other alphas. An alpha is dynamic and its' state evolves over time."

I am not yet certain whether we will find useful Universals that are not "alphas." I believe you can measure anything. And anything of value should be measured. Furthermore useful measures vary over time. This would seem to imply that "useful universals" would have multiple states and therefore would be candidate alphas. But this discussion will lead us to another critical issue Semat must face. If the resultant product is too "heavyweight" with "explicit measures" it will fall under its own weight. It is balance we seek for Semat to succeed.

Where the challenge ahead for semat lies is not so much on agreeing with the concept of alphas, but finding the alphas that are universal, and then finding their base states that are also essential to all software engineering endeavors. This will be no easy task, but as Ivar Jacobson recently said, "nothing I ever did in my life that had any real value was easy." .

Notes and References:

[1] CMMI refers to the Capability Maturity Model Integration which is a process improvement maturity model for the development of products and services developed by the Software Engineering Institute (SEI).

[2] Refer to WWW.PEMSystems.com and related book "Integrating CMMI and Agile Development: Case Studies and Proven Techniques For Faster Performance Improvement."

[3] Refer to http://www.pemsystems.com/SEMAT_position_McMahon.pdf

[4] Refer to http://www.pemsystems.com/iitsec-4-2002.pdf , http://www.stsc.hill.af.mil/crosstalk/1995/03/Pattern.asp