SEMAT Position Paper

Ian Spence, Chief Scientist/European CTO
Ivar Jacobson International

ispence@ivarjacobson.com

The most interesting thing about the SEMAT initiative is not the discussions about what software engineering is or isn't, or even the investigation of the underlying theories behind software development, but the search for a kernel.

I have thought for a while now that there must be some basic truths underlying what is involved in developing a piece of software. Some concrete frame-of-reference that we can use to understand and drive our software development efforts (see for example [1] and [2]).

Over the last 10 years or so I have spent a lot of my time evolving, comparing and documenting methods and practices. In this time I must have worked with and analyzed over 50 different methods including:

- commercial processes such as the Unified Process and the MicroSoft Solutions Framework
- public domain processes such as SCRUM, XP and DSDM
- organizational specific processes for domains as diverse retail banking, telecommunications, internet shopping, and military control systems
- assessment frameworks such as CMMI and SPICE

One thing I've noticed is that regardless of the method, or its underlying philosophy, there are some things that always have to be dealt with. Things that have to be handled regardless of what the particular method or practice calls them. In fact if these things aren't aligned it actually becomes almost impossible to select the right practices to support the needs of the software development teams.

A similar observation occurs whenever one does any kind of project or organizational assessment, another type of work I have been involved in many time over the last 15 years, without a shared frame of reference that identifies these universal elements comparing and contrasting the different ways-of-working becomes almost impossible.

The problem is that these universals are generally:

1. Not first class items in the practices and methods that depend on them. For example things like iteration and project are just concepts in the Rational Unified Process and even in a more focussed method like Scrum the Sprints themselves are not given the same emphasis as the Product and Sprint Backlogs.
2. Re-named and re-interpreted on a method-by-method basis. Everybody talks about releases but as a profession we don't seem to have standard definition we can all share.
3. Described in terms of sets of work products and deliverables rather than in terms of measurable objectives, achievements and goals. For example we always talk about the

project plan and the organizational structure rather than state of the project and the state of the project team.

Most recently I have been involved in the creation of a new kind of method [3], one that is assembled from a collection of independently defined practices. One of the most interesting things about this work is that it soon became obvious that if you want to mix and match practices from different sources they need to plug-into a specific frame of reference; a frame-of-reference that not only anchors the practices but allows you to understand their impact and coverage.

One of the more interesting things about this frame-of-reference was that we found that it actually turned out to be much more than just a conceptual model that could be shared by the practices. It actually ended up defining a practice independent process – a map of the territory that could be used to direct teams even without the explicit definition or presentation of the practices being followed.

As it was more than a conceptual model we called it the "process kernel" and used it as the entry point into any composed processes constructed from the available set of practices.

The actual format and content of this kernel is irrelevant to the goals of the SEMAT initiative what is more important is its impact and usage.

To date more than 25 practices have been defined that leverage and extend this kernel. These practices include a number of competing lifecycles, both waterfall and iterative management approaches, technical and social practices, pure development and coding practices, and quality and assessment practices amongst many others.

The kernel has also been used as the basis for measurement and reporting tools that can be applied even when the pre-defined processes and practices are not being used. It has even enabled a large software development organization to become more lean and agile by combining practices from the agile community such as Scrum and User Stories with other practices from the Rational Unified Process all within an environment of CMMI and process assurance. In this case the kernel provided the shared point truth for all projects regardless of their size, levels of ceremony, or selected practices.

The problem with any locally defined kernel of this sort is that its vocabulary and heritage limit its applicability. With SEMAT we have the opportunity to build on the experience of all the signatories and supporters to create a truly universal kernel, one that uses terminology acceptable to the whole software engineering community, one that is supported by many, many practices, one that is used in industry and academia, and one that truly reflects the discipline of software engineering.

As out-lined in the SEMAT call for action a kernel is only one of the many things needed to re-found software engineering, but in my experience it is the piece that will bring the initiative to life and provide a concrete frame-of-reference that software engineers will use on a day-to-day basis for many years to come. Without a concrete kernel that is used as the basis for describing the myriad competing practices and tracking the progress of software development teams the SEMAT initiative is in danger of becoming an academic talking shop that attempts to sit in judgement on people's practices rather than enabling them.

References:

[1] Enough Process. Let's Do Practices Parts 1 – 3, Ivar Jacobson, Ian Spence and Pan-Wei Ng, Dr Dobb's Journal Spring 2007 and Journal of Object Technology vol 6, no 6 July-Aug 2007

[2] Why We Need a Theory for Software Engineering, Ivar Jacobson and Ian Spence. Dr Dobb's Journal, October 2009

[3] The Essential Unified Process, www.ivarjacobson.com