SEMAT Position Statement
Jean-Marie Favre, OneTree Technologies, Luxembourg

Software? That is the question.
The Holistic View. What about this answer?[1]

Introduction

According to the SEMAT vision statement, one of the starting points of the project is to define what is "Software Engineering" and in particular what is "Engineering". However a fundamental question MUST NOT be eluded: "What is Software?". This notion is evolving over time and is polymorphic. We recommend considering an *Holistic View on Software*, and as a result an *Holistic View on Software Engineering*. We briefly describe the impact of this choice in terms of the set of theories to be "imported" and practices to be consolidated. Some suggestions in terms of SEMAT architecture are then briefly sketched. Finally, as SEMAT is a social construction we suggest considering this project from a Research 2.0.

The Holistic View on Software

As mankind is entering into a new era, the *Information Age*, Software becomes a *crucial* resource. Software is not only *ubiquitous* but also *polymorph*. The traditional view on Software is no longer enough to explain the broad range of phenomenon observed in the many software industries.

- *Traditional View on Software* (TVS) = Software RUNS on COMPUTERS
- *Holistic View on Software* (HVS) = TVS
  + Software lives WITHIN a CONTEXT, and is BY and FOR PEOPLE

The HVS recognize that they are many "*natures*" of software, the term nature referring here either to the *kind* of software, its *context*(s), or its *facets*.

- *Different kinds of software*. All software products are not equals. Consider for instance Information Systems, Embedded Software, Scientific Software, Autonomic Software, Social Software, etc. Defining a taxonomy of Software cannot be avoided in SEMAT.
- *Different software contexts*. Software could no longer be considered in isolation. Software is part of micro-systems to mega-systems (e.g. ultra-large scale systems, systems of systems).
- *Different software facets*. Development and usage are just particular facets to be taken into account when considering what software is. Other facets include software economics, ethics, law compliance, environmental issues, educational issues, etc.

The question "what is software?" is neither naïve, nor useless. The TVS consists in giving implicitly a single answer to this question; the fact that Software runs on Computers. While this view is simple and good enough in many circumstances, it implicitly assumes that everybody should agree on the exact

[1] A full version of this paper including references and detailed discussions is available at: http://megaplanet.org/jean-marie-favre/papers/semat-140310-full.pdf

(and unique) nature of software**. The Holistic View on Software seems to be the only way to address the true complexity that characterized the history of software industries, to understand what is happening now and even more importantly to be prepared for the future**.

The Holistic View on Software Engineering

The *Traditional View on Software Engineering* (TVSE) is based on the TVS. According to this vision, *Software Engineering* (SE) is to be based on *Computer Science* (CS), in turn based on *Mathematics*. One can only agree with the fact that SE should be based on Mathematics whenever possible. But this is simply not enough. Software is NOT a mathematical object and Mathematics is NOT a silver bullet. **Software is an intermediate between Computers, People and some Contexts. We see no scientific reason to concentrate only on the computer side.**

Beyond the TVSE, in the *Holistic View on Software Engineering* all scientific disciplines related to the notion of software are to be considered, ranging from social sciences (Software is BY and FOR people), to system sciences (Software is a system in context). Recognizing that Software is more than "programs running on computers" means that SEMAT should not only consider importing theories from Mathematics, but also from Ethnography, Anthropology, Cognitive Sciences, Linguistics, Social Sciences, Economics, Quantum physics, and … Computer Science. Software Engineering is *not* Computer Science.

Following Parnas, we believe that SE should be considered as the same level of other engineering disciplines and that this does NOT mean considering only the scientific side. That is, existing engineering disciplines provide a very large body of knowledge and SEMAT should probably rely more heavily on that. As Software Systems become parts of Systems of Systems and Ultra-Large Scale Systems, the integration with other engineering disciplines will sooner or later be *unavoidable* anyway. Moreover, software Engineering should be studied in the context of System Engineering, not in total isolation as this is too often the case.

On the short and medium term, reusing (abstracted) practices from other mature engineering disciplines may be more productive than spending all the energy on identifying theories that can be reused. It is more fun and intellectually rewarding from an academic point of view to (re)use theories from existing scientific disciplines but this is only *one* of the possible approaches to improve the maturity of SE.

All engineering disciplines are based on some holistic view on the artifact they produce. Should this be different in the case of Software?

The Holistic View applied to SEMAT

We believe that the core question is SEMAT should really be "What is software?". Accepting a variety of answers is a strategic decision. SEMAT should provide mechanism to describe explicitly different natures of software. Differences in nature are related to the very essence of software products, not (only) on the way they are produced. These different natures may (or may not) in turn lead to different Software Engineering Methods (SEM), but SEMs are consequences of software natures, not their essence.

Considering the evolution from UML 1 to UML 2 with all the corresponding pitaffs and successes is certainly of interest in the context of SEMAT, in particular if an holistic view on software is to be taken. It is clear that the architecture of SEMAT has been defined to build a family of method. But we believe that ideas like "*variation points*" should be reused at the level of the kernel of SEMAT. The idea is to deal explicitly at that low level with the different natures of software. We envision "software natures" as explicitly defined and named configurations of existing kernel elements. It should be possible to define configuration both via composition of existing elements or configurations, but also through parameterization of existing variation points.

To limit risks in the SEMAT project, it could make sense to define a "*traditional SEMAT kernel*" as the basis of the diamond architecture. However, whenever something does not fit in this traditional kernel, instead of answering that "this is out of the scope" of SEMAT, it could be worth to record explicitly the nature of the problem and the kind/context/facet of software that are involved in this problem.

It is clearly tempting to model only mature understanding of traditional phenomena (this could be the objective of the SEMAT traditional-mature kernel). But **it would be *much more powerful, yet not too risky* to consider an *onion architecture* with different levels of maturity explicitly stated.** As maturity and understanding will be gained, concepts could be promoted to more central onion rings and then be taken into account in the SEMAT diamond. This kind of architecture could prove very useful in the long term in particular because it enables at the same time to work on a core SEMAT architecture and while leaving room for more advanced and emerging aspects.

Towards SEMAT 2.0?

Software Engineering is a large field and many (sub)communities may have a different answer to the question "What is Software?". The HVS requires not only recording the variations (and evolution over time) to this question, but also to map them to the corresponding communities. In this context community engineering is of central importance and could empower the whole holistic approach.

While it is always possible to ask a few experts what they think about a particular topic and make them converge on a traditional view of software, the full power of community often comes with the *wisdom of crowds*. While a single expert has typically a single view on a particular topic, complexity but power comes with the confrontation of many different views by many different people. The holistic view on software is all about that and we believe that it can be only achieved through explicit community engineering. Just like for the kernel architecture, we envision a onion architecture for communities.

We see the SEMAT initiative as a potential catalysis for the SE communities to pack some SE knowledge and this in a community based approached**.** Community engineering is at the center of the holistic vision. **We believe that SEMAT could benefit from the recent advances in the Research 2.0 field and corresponding infrastructures. We suggest using an onion social architecture for SEMAT with (at least) the outside rings being included in a Social Network for Software Engineering.** While Research 2.0 (aka e-science, e-research, etc.) have been used for years in some scientific domains, there is no reason to not apply them in the context of Software Engineering (see for instance http://planet-research20.org/r2ose2010).