ISSN 0361-7688, Programming and Computer Software, 2012, Vol. 38, No. 1, pp. 1–12. © Pleiades Publishing, Ltd., 2012.

Semat—Three Year Vision[1]

Ivar Jacobson[a], Shihong Huang[b], Mira Kajko-Mattsson[c], Paul McMahon[d], and Ed Seymour[e]

[a] *Ivar Jacobson International*

[b] *Computer Science & Engineering, Florida Atlantic University, USA*

[c] *Royal Institute of Technology, Sweden*

[d] *PEM Systems, USA*

[e] *Apt Methods & Tools, Fujitsu, United Kindom*

e-mail: ivar@ivarjacobson.com, shihong@fau.edu, mekm2@kth.se, pemcmahon@acm.org, Ed. Seymour@uk.fujitsu.com

Advisors of the paper: Semat Advisory Board (Bertrand Meyer, Richard Soley), Arne Berre, Donald Firesmith, Capers Jones, and Harold "Bud" Lawson

Received July 9, 2011

Abstract—The purpose of writing this Three Year Vision paper is threefold. Firstly, it briefly recaps the progress Semat has made thus far; secondly, it lays out the future directions for people working actively within the Semat community; thirdly, it provides the background for seeking funding support from agencies, such as the European Community and the like. Funding support is necessary to sustain the ongoing activities of Semat and its growth into a broader community effort, as most people working within Semat are volunteers. As such, the paper may be both too much and too little for the wider supporter base. However, we intend to make our work fully transparent, hence, we publish it widely. We seek feedback and comments from supporters and signatories in order to improve the vision. In this context, other companion papers are being written to better address the specific needs for the practitioners, the industry and the academia.

Keywords: Software engineeirng, method, theory, practices, kernel, common ground

DOI: 10.1134/S0361768812010021

At the end of 2009, Ivar Jacobson, Bertrand Meyer and Richard Soley started a new initiative called Semat (Software Engineering Method and Theory) with the aim of refounding software engineering as a rigorous discipline. They recognized that the natural tendency in our field is to perturb systems minimally into approximate correctness, but this path cannot be sustained any longer if we are to support the computing industry and help it meet the demands of society.

They established a need to restart on a solid basis, taking advantage of all that has been learned in software engineering theory and practice over the past five decades.

The objective of this paper is to present how far we have come in realizing the vision for the first year, and to give a picture of where we want to be within three years, i.e., early 2014. The realization plan will be presented in separate documents.

1. GRAND VISION

The original call for action [3], as formulated in September of 2009, pinpointed the paramount concerns and issues that challenge the field of software engineering such as the reliance on fads and fashions, the lack of theoretical basis, the abundance of unique methods that are hard to compare, the dearth of experimental evaluation and validation, and the gap between academic research and its practical application in industry (see Appendix 1).

Against this backdrop, a solution was envisioned—the Grand Vision [4], which would be based on a solid theory, proven principles and best practices. At its heart it would be a kernel of widely agreed elements. The Kernel would provide the common ground reference to among other things help practitioners (e.g., architects, designers, developers, testers, developers, requirements engineers, process engineers, project managers, etc.) to compare methods and make better decisions of their practices.

The Grand Vision is defined as a multi-year effort, certainly more than three years. The work started in early 2010. The first step toward the Grand Vision [4] is described in Section 2, 'First Step—Creating the Basis for Semat'. In Section 3, 'Beyond the First Step—Moving Forward with Semat', the products of Semat are discussed, and how we will move forward

[1] The article is published in the original.

and realize substantial and measurable value to the software community are set out.

2. FIRST STEP—CREATING THE BASIS FOR SEMAT

To refound software engineering we knew we needed to do something that had never been done before: to discover "a kernel of widely-agreed elements"—the common ground of software engineering.

2.1. The Common Ground of Software Engineering

The software community has developed software for more than 50 years. Irrespective of the code being written, the software system being built, the solution being constructed, the methods employed, or the organizations involved, there is a common ground—a kernel of elements that are pervasive concepts and qualities—always prevalent in any software endeavors. Examples of essential elements include: work, team, requirement, software system, opportunity and stakeholder. The kernel provides the essence of software engineering.

Establishing this kernel provides software practitioners with the tools needed to better understand, compose and compare individual practices and methods, and to do their job more effectively. Companies can realize a consistent, identifiable framework for governance, whilst allowing their developers the freedom to use their preferred practices. It will provide a learning roadmap to help form new curricula and personal development goals, and it will support research by providing context and agreed subjects of value. Moreover, it will reduce the fashions and fads prevalent in the software industry today, and usher in a more pragmatic and objective era.

Finding the constituents of the kernel is crucial. We are uncovering universal, significant and relevant elements guided by the notion that, "You have achieved perfection not when there is nothing left to add, but when there is nothing left to take away.[2]" We are also ensuring that these constituents—"nothing left to take away"—are widely agreed upon.

Thus, a widely agreed upon kernel is the essence of Software Engineering. When we use the phrase "common ground" in this paper it refers to achieving consensus on the essentials of software engineering. When we use the term "kernel" it means the realization of common ground—common ground being the specification and kernel its realization.

2.2. Semat Applies Separation of Concerns

A key principle of the Semat initiative is to be inclusive of all relevant work in software engineering and not excluding anything that is or will be beneficial to any of its interest groups. For instance, even though a key target group is the practitioner, through extension we also address the process engineer; even though the primary subject area is software engineering, we also support systems engineering through extension; even though we focus on the essentials of software engineering, we also allow people to add details; even though our definitions are generic, we also allow them to expand and be more specific. Our approach to achieve this dualism is through the principle of Separation of Concerns (http://en.wikipedia.org/wiki/Separation_of_concerns). This principle allows us to specify a core, and then through extensions without changing or complicating the core add what is needed for the more specific cases (see Appendix 2).

In particular making the practitioner the target group is fundamental. Projects come in different scales. There is a huge amount of software development that occurs in small to mid-sized companies that do not have process engineers. Moreover, without getting the practitioners to adopt the result of this initiative, it will frankly just be an intellectual exercise.

2.3. Requirements

2.3.1. Getting to the requirements

Until now different methods[3] have primarily been described as isolated islands. Every method is basically a unique phenomenon described in its own language and vocabulary, not standing on any widely accepted kernel. A kernel of essentials will allow methodologists to describe new approaches without reinventing what is already known and agreed to.

Related to method but different is the concept of practice. It has been used frequently in software engineering for the last 50 years. The intuitive understanding is that a team usually has one method but many practices. Thus a method is perceived as larger and more complex than an individual practice.

Basically every software development team, with some exceptions, has its own method. Thus we expect that today there to be probably over 100000 methods in existence, with many of them never being described. This is perfectly right. We should expect a huge number of methods, but the number of relevant separate practices in use should be much smaller. In the software engineering literature there have only been a couple of hundred practices identified.

Thus, being able to design a method from a set of relevant practices, all described using a kernel of essential elements are key requirements of Semat. In Appendix 3 the relationship among method, practice, kernel and language is described.

[2] Antoine de Saint-Exupery.

[3] Here we don't distinguish between the term method and terms like methodology or process.

2.3.2. The one-line goal

The goal is to create a kernel and a language that are scalable, extensible, and easy to use, and that allow people to describe the essentials of their existing and future methods and practices so that they can be composed, compared, evaluated, tailored, used, adapted, simulated and measured by practitioners as well as taught and researched by academics and researchers.

2.3.3. Key requirements

With some repetition of what already have described, this section specifies the key requirements associated with the first step of Semat. They have not changed in any significant directions since they were first laid out in the first-year vision statement in February 2010 [4].

One of the key requirements is to identify and specify a kernel including the essential elements of software engineering. This kernel would then serve as a vocabulary—a map of the software engineering territory. The map would be used as a base on top of which we can define any method or practice in existence or are foreseen in the near future. This kernel should also be extensible to care for new technologies, new social working patterns, and new research. Note that this requirement is also an application of the principle of separation of concerns: separating the kernel from the specifics of the different methods.

Though every practice will be described using the kernel, it is also a separate concern allowing a practice to be merged with other relevant practices to form a composed practice. A method can be thought of as a higher-level practice that can be instantiated and used. We have sufficient evidence from many experiments that several such compositions results in a method and that every existing method in fact could be described as a composition of complementary practices. The kernel supports a way of defining methods allowing for reuse of practices (see Appendix 3).

However, being able to define methods in a practical way with high level of reuse was not all we intended to get. The methods defined using the kernel also have a dynamic aspect to support the practitioner in using the practices to do what they want to do, and after being done (for instance at the end of an iteration or a sprint) they are expected to retrospectively exam the method and adapt the method for what they have learnt. This changes the traditional understanding of a method.

Traditionally, a method definition is thought of as being instantiated, and the activities—created from the definition—are executed by practitioners (analysts, developers, testers, project leads, etc.) in some order to get result, specified by the definition. This view—"the team is the computer, the process is the program"—is not suitable for creative work like software engineering, which requires support for work, which is agile, trial-and-error based and collaboration intensive.

In effect, the kernel is defined using a domain-specific language, which has a static base (syntax and well-formed-ness rules) to let us define methods effectively, and with an additional dynamic concern (operational semantics) to let us use, adapt and simulate them.

The kernel and the language are small and light at their base but extensible to cover more advanced uses, such as dealing with life-, safety-, business-, mission-, security-critical systems.

2.4. Realizing the Kernel

These requirements are ambitious, but we had enough evidence that the goal could be achieved. Many of the leading supporters of the Semat initiative told us when joining that they had the kernel we were looking for, or had seen examples of kernels that had been developed in other companies. Moreover, there is evidence that a kernel can be developed on top of which practices can be defined (see ftp://ftp.ivarjacobson.com/outgoing/kernel/Software_Development_Kernel.pdf). This work only serves as evidence that the requirements are reasonable and can be implemented.

While each of these existing kernels may provide part of the answer Semat seeks for, none of these existing kernels have been widely agreed upon, which is critical to the ultimate success of our initiative.

Developing the kernel is not just a technical problem. It is primarily a matter of reaching an agreement on which the essential elements of the kernel should be and what they should contain, such as states and criteria for state transitions.

2.4.1. The concept of kernel

The kernel should neither be a new unified methodology, a new software process meta-model, a new body of knowledge, a new modeling language, nor is a trick to get people to build or buy more tools. The kernel should be as simple as a map of what we already have (e.g. teams and projects), what we already do (e.g. specify and implement), and what we already produce (e.g. software systems) when we develop software, irrespective of the way we work, whether we write documentation, or even if the result is good or bad. The kernel should be concrete, focused and light. For more detailed explanation of the concepts please see Appendix 3.

2.4.2. The governance of the kernel development

Recently, in order to provide the necessary governance of the work on developing the kernel, the responsibility for this work has been moved to the object Management Group (OMG, http://www.omg.org/). This move to OMG ensures the openness and

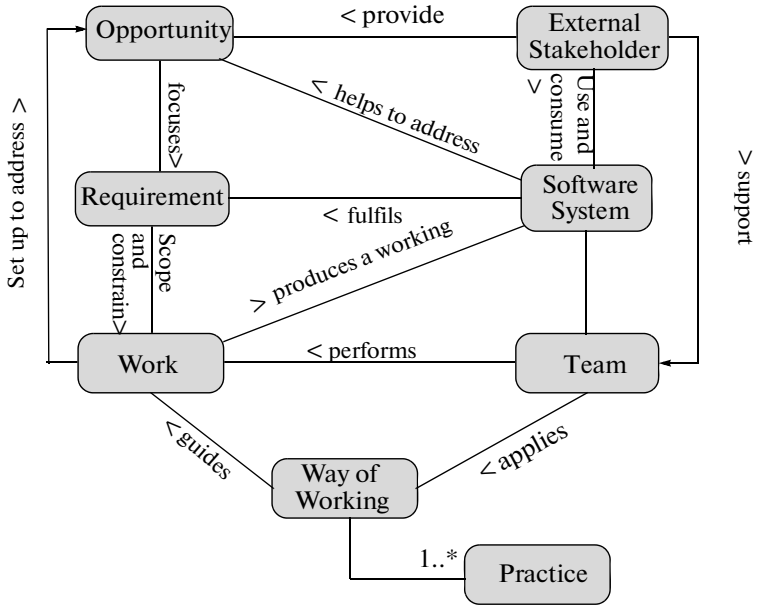

Fig. 1. The current set of suggested kernel elements. (Work is in progress and subject to change.)

fairness of the selection process and that the results benefit the entire community. A Request for Proposal (RFP) titled "A Domain-Specific Language and a Kernel of Essentials For Software Engineering (ESSENSE)" has been prepared by a group of OMG members and presented to OMG. The RFP solicits submissions for a language and a kernel allowing people to in a light way describe the essentials of their current and future practices and methods so that they can be composed, simulated, applied, compared, tailored, used, evaluated, measured, taught and researched.

This allows the methods, practices and the essential elements of the kernel to be described in the language.

The RFP is based on the Semat vision statement presented a year ago [4]. The next step is that OMG will review the RFP proposal and when approved (expected to happen in June 2011), any organization or team can respond with an initial submission for the RFP.

A group of about 20 people within Semat, who has been working on a candidate kernel since March 2010, will be one of the submitters. This group has members coming from around the world representing practitioners, executives, instructors and researchers.

2.4.3. Current status of kernel development

A working group within Semat has today a first version of a candidate kernel defined. It will be ready for a small number of potential users to test and to give them feedback indicating if the work is on the right track. As of today, this candidate kernel includes a map of the territory including an initial version of "things we always produce and progress", "things we always do", and "skills we always need to have". For example, the following elements are now rated as essential elements of this candidate kernel: opportunity, requirements, software system, team, work, way-of-working and practice (Fig. 1). These elements contain well-defined states.

As an example, requirements move through the states of conceived, shared, stable, correct, testable, and fulfilled. We are still working on "things we always do", and "skills we always need to have". This candidate kernel will eventually provide guidance during the usage of a method to assist practitioners in assessing the progress and health of their project in comparison to desired target states. Practitioners will employ this kernel to evaluate their current practices, and to extend their practices to fit specific circumstances.

However, it is expected that other parties will also provide their candidate kernels. In case there are several proposals, it is a standard procedure that OMG requests the different proponents to work together to come up with a joint proposal, which eventually will be adopted. The kernel finally adopted by OMG is expected to be stable but not static. It will continue to evolve as our understanding of software engineering improves and the field grows.

2.4.4. Continuing existing work

Since we today don't have a widely agreed upon kernel (which the OMG adopted kernel is expected to become), and since the existence of a kernel is the basis for Semat, how can we then today create a three-year vision for Semat?

Continued work in Semat will of course be dependent on the outcome of the OMG effort, but there is a significant amount of work that will have to occur just based on the fact that we are moving towards getting a widely agreed upon kernel.

For example the following kind of work, of which some already have started, are not really dependent on a detailed kernel being available: 1) Formulating the requirements on Semat from different users needs (work already started). 2) Assessing that the work of OMG meets the needs of the user groups. 3) Dealing with feedback from the user groups. 4) Keeping definitions available. 5) Developing practice-related theories covering societal and technical needs that support the concerns of industry, academia and practitioners, and can be validated and verified both empirically and formally. 6) Certifying practices. (Semat will not approve and disapprove the ideas of a practice but ensure that the definition of the practices allow for reuse.)

It is also very important for the success of any OMG adopted kernel that work goes on outside the standard forum that OMG sets up; this is to ensure that what comes out of OMG is also what users request.

2.4.5. Work after OMG's adoption of a kernel

After the OMG makes their adoption of a kernel, there will still be more work to do since we anticipate the kernel to be stable, but not static as we continue to move forward. Thus, we see it as imperative to work in parallel with the realization of the kernel as well as applying any versions of the kernel on real cases.

There is other work that is dependent on the kernel, for instance specification of reusable practices. Practices are expected to be defined both in textual form, which of course can be done already today, and in the language being developed. Thus some work has to wait until the language is specified. As we know more about the details of the kernel, we will need to update parts of the results, but at the end of the three-year period we anticipate that we will have settled on the products of Semat, as further discussed in Section 3.1.

2.5. Using the Kernel with Varying Level of Detail

The kernel is lightweight so it is easy to understand and use as a vocabulary or a map for defining practices and entire methods. Practices can be defined at any level of detail with two extreme uses—very light and very rigorous. In many cases a middle ground will be used.

For small teams it is sometimes perfectly right to keep the practice definitions as tacit knowledge developed through conversations within the team. For teams working with business-, mission-, safety-and security-critical systems, more rigor is required and the practices have to be defined in a language with a formal foundation. Thus the kernel language should support several such styles to define practices.

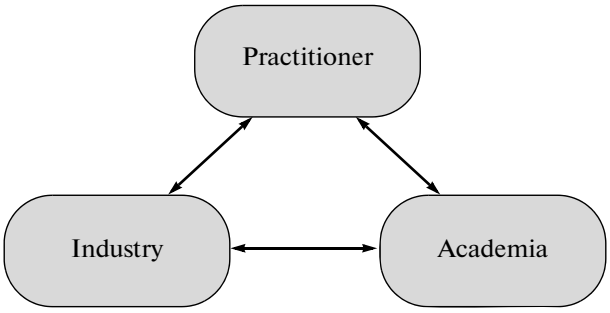

Fig. 2. The software engineering community and its most important user groups.

3. BEYOND THE FIRST STEP—MOVING FORWARD WITH SEMAT

A widely accepted kernel and the language are fundamental basic components and critical tools to refound software engineering and to establish a common ground. "Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning."[4] It is the first step that allows us to widen the work, even before their implementation will be generally available.

The paramount goal is to make successful the software community as a whole. This community is huge and to reach our objectives we will have to focus on the most important user groups, namely the practitioners, the industry, and academia (Fig. 2). We need to make them successful individually and as "partners". In simple and not exhaustive words: the practitioners and the industry drive what needs to be taught and researched. The academics teach and formalize what needs to be better understood. The practitioners are the doers in the industry and the industry leads the practitioners in creating business value.

These are broad user groups with a variety of diverse interests within the field of software engineering. Nevertheless, Semat has been, and shall always be, an open and inclusive initiative. The simplification of the user groups is intended to aid the discussion and elaboration of key objectives and measures.

Over the course of the next three years Semat will through its participants develop a set of products in order to be successful.

3.1. Products

The Semat community is through its participants expected to develop a whole spectrum of products to support its vision. These products are expected to be released separately and at different points in time over the next three years.

[4] Sir Winston Churchill, Speech in November 10, 1942.

The Kernel and the Language. Through the work of OMG a kernel and a language based on the agreed common ground for software engineering will be established. Its publication will promote and enable a new ecosystem for methods & practices based on an open standard. It will not be large. This kernel will encapsulate what we all agree is essential and used by practitioners on every project when delivering software. It will include terminology in a way that aids understanding and communication with respect to what we all agree about requirements, teams, software system, stakeholder, work, opportunity, and measurement.

Tools. A collection of tools (including open source)—either as stand alone or plug-ins for existing tools—will become available that enable people to author, browse, compose, compare, question, measure and use practices and methods. Tools for different areas of concern in software development become interoperable through the use of the kernel and the language. The focus of these tools will be the needs of the practitioner whose primary concern is developing quality software leading to satisfied customers.

The Practice Market Place. The open standard kernel and the language will enable the publication, cataloguing and exchange of practices. The marketplace will provide an environment where developers are given appropriate freedom to use their preferred ways of working within their specific context. It will be a place where proven practices as well as new innovative ideas are easily accessible.

Curricula. A new and more systematic foundation for teaching software engineering will emerge, which supports learning in academic and professional environments. Curricula based on the kernel, the language, practices and methods will be developed and used both in computer science and software engineering programs in our universities, and in education given by research scientists, electrical engineers, mechanical engineering and others whose primary field of expertise are in varied disciplines.

Text Books and Papers. New textbooks and reference material to support curricula and personal development based on the kernel and the language will be authored and made publically available. Many books on practices defined using the kernel that target at different level of users will be written to support practitioners in improving their way of working.

Research. The objective comparable nature and ability to tailor, use, adapt and simulate practices will result in a renaissance of software engineer research. Researchers have a common infrastructure serving as a test-bed and fast deployment of new ideas (extensible practices).

3.2. An Illustrative Usage Scenario: A Team in a Small Company Developing a Single Application

The following scenario illustrates the usages of Semat products:

Imagine a project lead is just about to set up a new project to develop a major release on an existing product.

Her team has a way of working but it is not documented. All have learnt about the Semat kernel. They want to change their way of working to become more agile, in particular they want to improve the way they work with requirements and test.

The project lead and her team start to work from the kernel. They sketch (e.g., do something similar to a use case model) some of their existing practices, which they want to keep. Since they know their way of working, this is done quickly. Basically they just work through their old terms and synch them up with the names of kernel elements.

Then they go to their companies practice library and select the practices, which best meet their needs. They download a tool assisting them to understand the new practices (maybe their existing practices will be briefly sketched as well). If needed, they will tailor the practices to fit their specific needs (in this scenario very limited tailoring). The tool also helps them to use the practices in the project, iteration (or sprint) after iteration. They also need (other) tools to support their new practices, but the old tools will still work for their existing practices.

A positive side effect of the practices is that the training material is very effective. The practices in the practice library are actually coming from the Semat marketplace for practices, so its training material has been developed, has been used around the world, and it has been improved over and over again. Some training may also have e-learning facilities.

The team is supportive of the changes because they can grasp more easily the health of the project; at any moment they know where they are and where they are going; they know exactly when something is done. The team was also happy because they had very little overtime.

The project lead is satisfied because she sees that the outcome is better, the time to market is shorter and predictable, the costs are lower than estimated, and her customers are happy. And, now other teams want to reuse her experience, so she is also happy.

For illustration purpose, some selected scenarios for the primary user groups—practitioners, industry and academia—are presented in Appendix 4.

3.3. Success Factors for Users

Semat's success will be measured in its ability to positively affect the course of software engineering

with the practitioners, the industry and academia. Better, Faster and Happier (BFH) are metrics addressing both common objectives and specific concerns of these individual groups.

In the broad sense, 'better' implies lower defect potentials, less rework, and higher levels of defect removal efficiency than today's norms; 'faster' implies quicker development cycles than today's norms; whereas 'happier' implies improved customer and employee satisfaction throughout a multi-year period that encompasses total cost of ownership.

In a specific situation, the BFH has distinctive meanings for the three user groups we focus on:

(1) For the practitioner better means your competitive value is higher, you develop better software, and you have experience of more practices. Faster means you learn faster, and you get your job done faster. Happier means you are more self-confident and you can easily move from one organization to the other.

(2) For industry better means you are innovative, you deliver software with high quality and you have objective measures of project health.

Faster means reduced delivery time. Happier means satisfied customers and satisfied employees.

(3) For academia better education means competent students equipped with solid concepts and theory meeting industry needs and instructor's employing a consistent teaching roadmap independent of fads but open to innovations. Faster means faster learning and faster transition of students to industry. Happier means more motivated students and a more enjoyable education experience.

For academia better research means researchers focusing on areas more relevant to real-world problems. Faster means faster technology transfer from laboratory to industry and faster feedback from industry to research. Happier means research has impacts on the quality of software products.

3.4. Measuring Success

Semat is set up to refound software engineering; a goal, which of course is hard to measure, nevertheless we must do it. Eventually we want to measure the impact our products have on the community building better software. The impact can be measured from two aspects that complement each other and yet perhaps overlap in some cases: qualitative measurements and quantitative measurements.

3.4.1. Qualitative measurements

We need to identify how Semat products can be related clearly to the objectives of practitioners, industry and academia. These objectives can be qualitatively measured by using BFH—more precisely by using the BFH attributes.

The BFH are the objectives we wish to accomplish within the software community. They have attributes that are more concrete and conducive to measurement (see Fig. 3). We will as Semat goes forward select and agree on these potential attributes, and we will identify, describe and specify how to measure these attributes. We will—apply the principle of separation of concerns—separate those attributes that are common to all our target user groups from those representing specific needs requested by the diverse users.

Some of the means of conducting qualitative measurements include gathering feedback using for instance interviews and questionnaires of comments, opinions, evaluations, and views on our products. For instance, we will collect opinions on how our products influence the attributes for each user group (practitioner, industry and academia). As an example we may measure how much more competitive a practitioner becomes using our products.

To illustrate our BFH approach, we have outlined a set of attributes and placed them within the BFH objectives in Fig. 3. The example, however, should not be perceived as a fully-fledged and ready to use framework. We suggest that three of these attributes are essential: Objective Measures of Health, Competiveness, and Satisfaction. They will therefore be further elaborated in Appendix 5.

3.4.2. Quantitative measurements

The impact of the Semat product can also be measured quantitatively by measuring the direct impact of our products on the software being built. For example, measuring quicker development cycle, better ROI, fewer defect and rework, better reliability, and improved customer satisfaction. These quantitative measurements provide objective evaluation and assessment of success. Assessing the long-term value requires to conduct empirical studies within the organizations after the adoptions of the results, which will be an ongoing task for the initiative.

Another quantitative measurement of this initiative's impact is measuring the adoption of its products by the software community. Whether or not this is an acceptable approximation will remain for discussion as the initiative moves forward, it certainly is an interesting measure and an important one as wide acceptance of its products is critical to success. The actual number of adoption shouldn't come from thin air, but from the successes that practitioners, industry and academia experience with its products.

We project (but are open to other suggestions) that within three years' period of time:

(1) More than 30,000 practitioners use Semat inspired products in their day-to-day work;

(2) More than 50% of all universities with computer science departments use the kernel and the lan-

Fig. 3. The BFH objectives illustrated with potential attributes.

guage as part of software engineering curricula, students' senior projects and theses;

(3) 80% of the companies in the Fortune 500 have at least one active project deploying the kernel and the language;

(4) The Semat tools' open source project has more than 100 committers;

(5) More than one major software vendor supports Semat's kernel and language;

(6) More than two major outsourcing service companies have adopted the Semat kernel and language to host its practices.

We have suggested the numbers based on an anticipation of what is needed to declare a success.

Although they are not based on any empirical study, they are not coming from random speculation either. These numbers are achievable, since two of the leading vendors have significantly more users of products in the process space. The major differences are that these products have no common ground (or kernel) and use vendor-specific languages—both being critical objectives of Semat. Clearly, getting support from major vendors is important, but such support will only come through support from major customers of these vendors. We have gained some support from several major corporations in the world, and we still need to do much more here going forward.

4. FINAL WORDS

Semat is now on its path to implement the Grand vision that initiated the community. The road ahead will be filled with the challenges of a frontier expanding community. Challenges are also opportunities. When succeeded, Semat will significantly change the software engineering community and give it a new platform from which to build software better, faster and happier. Watts Humphrey may have shown real forethought ahead of the first Semat meeting in Zurich, March 2010, when he said: "This meeting in Zurich is likely to be an historic occasion much like the 1968 NATO session in Garmish." Semat results are a collective effort from the community for the community. Working together, we can refound software engineering.

ACKNOWLEDGMENTS

The authours would like to thank the following people who have contributed to the paper and given valuable feedback: Dave Cuningham, Brian Elvester, Michael Goedicke, Winifred Menezes, and Ian Spence.

APPENDICES

The five appendices below detail some of the topics in the main body of this paper.

APPENDIX 1: BRIEF SEMAT HISTORY

At the end of 2009, Ivar Jacobson, Bertrand Meyer and Richard Soley (known informally as the "troika") started a new initiative with the aim of re-founding software engineering as a rigorous discipline. They recognized that the natural tendency in our field is to perturb systems minimally into approximate correctness, but this path cannot be sustained any longer if we are to support the computing industry and help it meet the demands of society. They established a need to

restart on a solid basis, taking advantage of all that has been learned in software engineering theory and practice over the past five decades. In a 'call for action' statement they described the challenges and a path going forward—The Grand Vision.

CALL FOR ACTION

Software engineering is gravely hampered today by immature practices. Specific problems include:

—The prevalence of fads more typical of fashion industry than of an engineering discipline.

—The lack of a sound, widely accepted theoretical basis.

—The huge number of methods and method variants, with differences little understood and artificially magnified.

—The lack of credible experimental evaluation and validation.

—The split between industry practice and academic research.

We support a process to refound software engineering based on a solid theory, proven principles and best practices that:

—Include a kernel of widely-agreed elements, extensible for specific uses

—Addresses both technology and people issues

—Are supported by industry, academia, researchers and users

—Support extension in the face of changing requirements and technology

The troika were pleased, honored and gratified to find that in a short period of time, a dozen corporate and academic organizations, and some three dozen well-known individuals from the field of software engineering and computer science, became signatories to support the vision. In addition, more than 1400 other supporters agreed to the call.

The Semat Vision Statement (http://www.Semat.org/pub/Main/WebHome/SEMAT-vision.pdf) captured the troika's understanding of the problem, the potential, and a vision for its first step. Since its publication in February 2010, more than twenty people from a cross section of industry & academia have volunteered significant amounts of time and effort to help bring that vision to life. That is to support a shared idea of software methods based on rigorously well-understood and comparable practices that are defined through a kernel set of elements and a domain-specific language.

APPENDIX 2: SEPARATION OF CONCERNS

Semat relies on the principle of Separation of Concerns (for general discussion see http://en.wikipedia.org/wiki/Separation_of_concerns).

1. It separates its support for software systems from its support for systems (including hardware) and solutions (including hardware and peopleware). Thus the kernel and the language must be extensible additionally to support systems and solutions without complicating their usage for people who are software practitioners.

2. It separates at least two different views of the process: the process engineers' view and the practitioners' view. The primary users of methods and practices are project practitioners (developers, testers, project leads, etc.). Semat result should be accessible to both practitioners and process engineers, but should target the practitioners first and foremost. Through extensions the result should also support process engineers efficiently without complicating its usage for the practitioners.

3. It separates the essentials from the non-essentials, such as key guidance from detailed guidance, or explicit knowledge from tacit knowledge. This allows process engineers to create lightweight methods with scalability. In other words, this work is about the essentials only—the smallest common denominator that is present in all successful software efforts.

4. It separates the generalized definitions of terms from specialized definition details, allowing for the inclusion, rather than the exclusion of earlier work on methods. In other words, we are looking for the common ground upon which all existing methods can build.

APPENDIX 3: KEY CONCEPTS

The kernel and its elements will be precisely defined using the domain-specific language (the domain being practices for software development). Additionally, the language will also be used to define practices and entire methods.

The result will satisfy these requirements:

1. A method is a composition of practices (as opposed to an interconnection of process/method components, disciplines, or similar).

2. A practice is an approach to doing something with a specific purpose in mind. There are several kinds of practices, but the basic and most important kind of practices are the concrete practices. A concrete practice is a complete end-to-end activity with a clear beginning and end supporting software practitioners in getting their job done. These practices give value one-by-one, they are what users want to make lean, and they are what you want to measure and provide metrics for, all of which are critical differentiators.

3. All methods have something in common—'the common ground' or 'the essence of software engineering'—the kernel. Examples of essential elements are:

work, team, requirements, software system, opportunity, stakeholder community, etc.

4. Methods need theory—our work must stand on a solid theoretical basis. Methods being composed of practices, practices being described in terms of the essential elements and in terms of other elements such as activities and work products. All formalized into a language is the beginning of such a theory. Moreover, many practices can be formalized or supported by formal techniques. They can for instance be measured using statistical methods.

5. Methods are dynamic and used. Methods are not just descriptions for developers to read, they are dynamic, supporting their day-to-day activities. This changes the conventional definition of a method. A method is not just a description of what is expected to be done, but a description of what is actually done.

APPENDIX 4: SOME USAGE SCENARIOS OF THE KERNEL AND THE LANGUAGE

A4.1 A Practitioner Scenario

When a student leaves college today and enters industry, they can directly apply certain skills they have learned (such as JAVA, or C++), but there is a great deal they still must learn. While some of this is unavoidable, such as terminology unique to a given industry, today the use of terms as fundamental as requirements, and team can vary widely from one company to the next.

Today we live in a very mobile society where people change employers often throughout their career. When a software engineer moves from one job to another in a different company—or even in a different part of the same company—she can take her experience with her, but there is a great deal that must be relearned within that new environment. This can discourage the software practitioner even to the point of making a decision to seek a different career path.

Establishing a kernel is not about creating a standard that excludes certain users. A kernel based on common ground encourages new approaches appropriate to the job at hand, and established on essentials we all agree to.

So what does this mean for the software practitioner? Software practitioners of the future will have more opportunities for employment, as they will experience the freedom of knowing they have greater mobility without jeopardizing job satisfaction.

They will know that what they learn in the university can be counted on when they move into industry and as they move from one company to another within industry—or from one project to another within a company. Knowing they have learned the essentials will also bring greater self-confidence and self-fulfillment, as the software practitioner will be able to focus more of their limited time on the unique aspects of the job that brings greater value to themselves, their employer and their customer.

A4.2 An Industry Scenario

Key to all organizations is an accurate understanding of the health of their projects, and knowing what actions to take in the face of trouble. Effective objective measures are relied upon today to help managers know when action is needed. Mounting objective data exists indicating root causes of most troubled projects are traceable to a failure of organizations to enact appropriate ways of finding problems early.

Companies frequently adopt or develop methods with a one-size-fits-all view hoping corporate standards hold the answer. Unfortunately this approach has little chance of success as it fails to consider critical and varying factors between teams, projects, products, organizations, etc. Is it possible a better approach exists?

To the program manager working in the software industry the kernel provides a means to establish a consistent non-controversial framework of essential concerns across all her projects, irrespective of their size or shape. Distributed development is fairly common in many large companies. The kernel will among other things make communication easier across different geographies.

To the team leader establishing a way of working suited to her new project, the kernel is her means to select appropriate ways of working to support her team and meet her corporate goals. Another benefit is that it will be easier to assign people to projects (even if the domain is different) because they will have a common basis to build their work on.

To the process professional the kernel provides a means to communicate how an organization works in a format easily digestible by established employees and new recruits alike. It improves adoption, facilitates reuse, and provides teams with a means to integrate what we all know works into appropriately tailored approaches.

A4.3 An Academia Scenario

Looking at today's software engineering education, different universities and professors have different requirements and interpretations related to how software engineering should be taught, and what should be taught. Some accreditation guidelines exist. For examples, the Accreditation Board for Engineering and Technology (ABET) in the U.S, and SWEBOK Curriculum Guidelines for Undergraduate Degree Program in Software Engineering. However, these tend to provide guidelines at a very high principle level leaving the implementation details to individual schools and professors. Lacking a core fundamental kernel of software

Objective Measures of Health, Competiveness, and Satisfaction

Objective Measures of Health

Practitioner

—You, as a practitioner, will have the means to form objective opinions on your work, the way you work, and your own personal development. By having a common ground, you will be capable of making informed decisions regarding the design and reuse of the practices. You will be able to take a critical view of the innovative ideas, and, thereby, be able to look objectively at industry innovations—to evaluate whether they are beneficial or detrimental to your work. Your experience, from a basis of informed, sound practice, will feed you with new, innovative ideas for improving your ways of working.

Industry

—You, as an organization, will be able better to govern your company. The common ground will enable you to support your teams in focusing on the opportunity through choices in the way they work, whilst you retain consistent corporate oversight. You will be able to take a critical view of the innovative ideas, and, thereby, be able to look objectively at industry innovations, evaluating the benefits or detriments. Your experience, from a basis of informed, sound practice, will feed you with new, innovative ideas for improving your ways of working.

Academia

—You, as a researcher, educator, teacher or student, will be able better to take control of your research, teaching, pedagogical approach or your approach to study respectively. By having a common ground, you will be capable of making informed decisions regarding the direction of your work. You will be able to take a critical view of the innovative ideas, and thereby, be able to look objectively at industry innovations. This implies that you will be able to evaluate whether ideas are beneficial or detrimental to your work. Your experience, from a basis of informed, sound practice, will feed you with new, innovative ideas for improving your work results.

Competitiveness

Practitioner

—You, as a practitioner, are more competitive because you are faster when doing your work and you produce high-quality results. Therefore, you will find it easier to frame your skills and achievements along with your colleagues so you can better work together and grow as a team.

Industry

—You, as a company, are more competitive because you can get up to speed faster, shorten lead-time and thereby be first on the market.

Academia

—Because your research is based on a common ground, you, as a researcher, create rock-solid and robust research results, are more competitive within research, and more attractive to your future/current sponsor.

—Because your teaching is based on a common ground, you, as an educator, provide quality education, are more competitive, and more attractive to both students and teachers.

—Because your knowledge is based on a common ground, you, as a student/teacher, possess a stable foundation with which you can evolve, explore, and convey new ideas in an objective and persuasive way. Therefore, you are more competitive, and more attractive to your future/current employer.

Satisfaction

Practitioner

—You, as a practitioner, are confident that you have the requisite knowledge to perform well, and be judged consistently within the market place. Your well-being is substantially improved because your competency is understood and recognized within your organization and others, providing better job mobility,

Industry

—You, as an organization, are confident that you have developed your products and peoples' competencies with better results and a recognized consistency. Your organization's interests are aligned with those of your customers, partners, and employees. Therefore you enjoy improved customer and employee satisfaction.

Academia

—You, as a student, are confident that what you have learnt is derived from a stable, consistent, and recognized foundation, and that you are employable at any software company in the world

engineering, results in a wide range of education approaches without a clear theoretical basis.

This situation leads to students from different education backgrounds having different skill sets, which fail to meet industry's need. It also leads to research chasing fads rather than following a clear balanced roadmap.

To academia a kernel based on common ground means a foundation to (a) teach software engineering, (b) design software engineering curricula, and (c)

demonstrate to students the pros and cons of different ways of working. A kernel based on common ground ensures the essentials of software engineering are taught in a uniform way across different universities and education programs.

From a research perspective, a kernel provides a reference for the conduct of experiments on different software engineering approaches relevant to real world problems, and a solid foundation to aid the separation of hypes from reality.

APPENDIX 5: SOME EXAMPLE BFH ATTRIBUTES

As shown in Fig. 3, we have initially come up with ten Better attributes, nine Faster attributes and six Happier attributes. Moving forward we will most likely find that several attributes may impact more than one of our BFH objectives, but for now our example serves as an illustrative approximation.

In table we discuss our three selected attributes and what they could mean for our user groups: practitioner, industry and academia.

REFERENCES

Below are some seminal documents of Semat history:

1. Jacobson, I. and Meyer, B., Methods Need Theory, *Dr. Dobb's Journal*, August 06, 2009. Online at http://www.drdobbs.com/architecture-and-design/219100242
2. Jacobson, I. and Spence, I., Why We Need a Theory for Software Engineering, *Dr. Dobb's Journal*, October 02, 2009. Online at http://www.drdobbs.com/architecture-and-design/220300840
3. Jacobson, I., Meyer, B., and Soley, R., Call for Action: The Semat Initiative, *Dr. Dobb's Journal*, December 10, 2009. Online at http://www.drdobbs.com/architecture-and-design/222001342
4. Jacobson, I., Meyer, B., and Soley, R., The Semat Vision Statement online at http://www.semat.org/pub/Main/WebHome/SEMAT-vision.pdf
5. Shihong Huang, *The 1st Semat Workshop Report*, online at http://www.semat.org/pub/Main/SematZurichMarch2010/Zurich _meeting_report.pdf
6. Shihong Huang, *The 2nd Semat Workshop Report*, online at http://www.semat.org/pub/Main/WebHome/2nd_Semat_Workshop_Report.pdf
7. Shihong Huang and Paul McMahon, *The 3rd Semat Workshop Report*, online at http://www.semat. org/pub/Main/WebHome/3rd_Semat_Workshop_Report.pdf