DOI:10.1145/2524713.2524723

Article development led by acmqueue
queue.acm.org

Combining agile and SEMAT yields more advantages than either one alone.

BY IVAR JACOBSON, IAN SPENCE, AND PAN-WEI NG

Agile and SEMAT—Perfect Partners

TODAY, AS ALWAYS, many different initiatives are under way to improve the ways in which software is developed. The most popular and prevalent of these is the agile movement. One of the newer kids on the block is the Software Engineering Method and Theory (SEMAT) initiative. As with any new initiative people are struggling to see how it fits into the world and relates to all the other things going on. For example, does it improve or replace their current ways of working? Is it like lean, which supports and furthers the aims of the agile movement; or is it more like waterfall planning, which is in opposition to an agile approach?

Many have wondered whether SEMAT and agile are complementary or competitive initiatives, and if they are complementary, how do they fit together? In this article we demonstrate how these two initiatives support each other and we discuss the benefits of using them together.

Generally speaking, both initiatives promote non-prescriptive value-based philosophies that encourage software development teams to select and use whatever practices best fit their context and, more importantly, continuously inspect, adapt, and improve their ways of working. These two initiatives complement one another, providing the perfect foundation for teams that want to master the art of software development.

The agile movement has provided a new way of looking at the day-to-day activities of software development—how teams are built and work is organized. This has led to the empowerment of development teams and the prominence of agile practices (such as Scrum and test-driven development) as the modern developer's practices of choice.

SEMAT is a new way of looking at the domain of software engineering (here the term is used interchangeably with software development), providing an understanding of the progress and health of development efforts and how practices can be combined into an effective way of working. SEMAT adds to the power of agility by providing a common reference model all teams can when continuously inspecting, adapting, and improving their ways of working.

The two initiatives when used together, truly empower teams to innovate, experiment, and improve the results they deliver.

This article focuses on how SEMAT can help existing and future agile teams. It is designed for those already familiar with agility.

What SEMAT Adds to Agile

Agile provides a set of values that influence and shape the way software developers go about their daily work and interact with one another, their customers, and their stakeholders.

It has also given us many methods that share common principles but differ in practice. These are methods that developers must be able to inspect and adapt as circumstances change. The agile methods give teams a great starting point on their agile journey but they need to evolve to meet the team's changing needs and reflect the lessons they learn. This is reflected in the growing number of agile teams that assemble a bespoke method from the available set of practices rather than taking a made-to-measure method off the shelf.

The use of SEMAT can help agile teams do the following:

Detect systemic problems early and take appropriate action. Agile teams continuously inspect and adapt using fast feedback and close collaboration to avoid problems and provide direction to the team. To support and encourage this way of working, SEMAT provides a number of simple checklists to help teams understand their progress and health, and to help them in the early detection of problems with their way of working. The checklists that SEMAT provides are akin to those used in other professions. For example surgery teams in U.K. hospitals reduced death by surgical errors by 47% by using a simple 19-question checklist that had questions such as "Do you know the names of the other members of the surgical team?" In the same way the use of the SEMAT checklists reduces the risk of teams failing catastrophically by helping them avoid many of the common mistakes that lead to failure such as ever increasing technical debt, loss of stakeholder support, inefficient ways of working, unrealistic expectations and dysfunctional teams.

Measure the team's progress and health regardless of the method or practices selected. The key measure of progress for all agile teams is the amount of working software they produce and the speed with which they produce it. SEMAT complements these measures by providing another view of the progress and health of the team and its work—a view that can help teams maintain their speed as they and the systems they produce mature. By using SEMAT and the simple checklists it provides to assess their current state, teams can easily understand where they are, where they should go next, and how their efforts fit within any organizational governance practices they need to support.

Compare and contrast practices and select the best ones for the team. Agile teams are perpetually looking for new practices to help them improve their way of working and evolve their methods. SEMAT provides the mechanisms to understand the extent, purpose, and content of practices, helping teams understand their coverage and where they overlap, conflict, or complete. It also allows teams to plug-and-play practices, safely mixing and matching them within the context of their favorite agile framework—for example, Scrum or Kanban.

Evaluate the completeness of the set of practices selected, and understand the strengths and weaknesses of the way of working. In the rush to adopt new practices, teams sometimes leave holes in their way of working, the consequences of which often do not become apparent until the team's speed starts to drop and it consistently falls short in achieving its objectives. This does not mean the way of working needs to be predefined or complete; in fact, it is probably better if it is not. What is important is the team members are aware of what they have agreed on, where they are aligned, and where they might need help. The use of SEMAT helps teams reason about the way of working and make fact-based decisions about the breadth and depth of their selected set of practices. Having mechanisms to help understand the strengths, weaknesses, and completeness of their way of working is invaluable for those teams truly committed to continual improvement.

Keep an up-to-date record of the team's way of working, and share information and experiences with other teams. The agile community thrives on collaboration and interaction. The sharing of practices and experiences helps individuals, teams, organizations, and the industry as a whole improve and evolve. SEMAT provides mechanisms to help teams accurately record their way of working in a lightweight, agile fashion, which they can share in real time with their colleagues and collaborators. This provides transparency with respect to the team's way of working, and it helps everyone understand what the team is doing without getting confused by out-of-date descriptions of what the team is supposed to be doing or what the team members thought they would be doing before they actually gained experience doing it.

Be agile with methods, easily and safely evolving the team's set of practices as it inspects and adapts its way of working. Inspecting and adapting the way of working is essential for any agile team that truly wants to continuously improve. Its effectiveness can be hindered when teams: become too wedded to the current set of practices, effectively freezing their way of working; select different but less-effective practices that introduce more problems than they address; or do not understand where they are in the evolution of the software system and therefore which practices they should change. SEMAT provides the frameworks and thinking tools to help teams more effectively inspect and adapt their way of working, understand the consequences of their decisions, and continuously improve their way of working.

SEMAT for Agile Organizations

SEMAT provides additional support that helps entire organizations become agile without compromising the agility of the teams that form them. In particular, it helps:

Establish the ground rules for software development within the organization, and capture organizational values and principles in a practice-independent fashion. Software development does not happen in isolation. Development teams must always be cognizant of the culture, values, and principles important to the organizations they work with. They need to establish some common ground and shared understanding with the other teams and areas of the organization they interact with. SEMAT provides a simple definition of the common ground shared by all software-development teams. This forms a firm foundation for organizations wanting to integrate software development into their businesses and value flows. Organizations can extend the SEMAT definitions to capture any additional rules or advice that applies to the specific kind of software they develop or the specific environment within which they develop it. Establishing the common ground is a prerequisite to organizational agility, but it is not sufficient. It should be complemented with an organizational practice exchange where the teams can share the practices they use.

Define practice-independent governance procedures and quality gates. For business, legal, and safety-critical reasons, many organizations feel the need to apply governance to their software-development efforts. Most large organizations are legally required to perform financial and/or technical governance on their software teams and the software they produce. Unfortunately, many organizations define their governance as a series of sequential phases, each with a predefined set of required artifacts that must be completed and signed off before the next phase can be started.

It is impossible for agile teams to achieve their full potential in this kind of rigid, prescriptive environment. Governance is there to provide checks and balances and ensure the quality of the results produced. Governance procedures and quality gates should be aligned to the natural evolution of the software systems produced, focused on the key results required rather than artifacts to be produced, and manifested as simple practice-independent checklists. They would then provide a framework to support, rather than inhibit, agile and lean ways of working. This is the approach that SEMAT takes, allowing governance procedures and quality gates to be defined in a lightweight and practice-independent fashion. The agile teams can then mix and match whichever agile practices they desire, and they can continuously inspect and adapt without ever having to fall out of governance.

Track and encourage the use of practices within the organization. Agile teams love to learn and share new practices; it is a fundamental part of the approach to continuous improvement. By basing all software-development effort around a common ground, teams can more readily and easily share their practices. By setting up a practice exchange to facilitate the sharing and distribution of practices, an organization can gain insight into which practices are being used where, and which sets of practices are producing the best results. This helps organizations to become true learning organizations continuously evolving their set of recommended practices, withdrawing those that are past their sell-by date, and promoting new practices when needed.

More readily and easily form teams and mobilize teams of teams. Although agile teams are intended to stay together, the reality is they are regularly changing team members, even when they do not work for organizations that insist on matrix management approaches and constant reorganization. Context switching in this way can often reduce velocity, increase friction, and waste time. SEMAT provides teams with a common language for software engineering that will help them understand one another, clearly express themselves, and share the practices they know—all of which will help them collaborate quickly and effectively—minimizing wasted time, pointless discussions, and unnecessary misunderstandings. It also provides mechanisms for modeling the competency required by teams and attained by individuals. This can help organizations find the right people to join the right teams and then observe their development as software professionals.

Scale agile approaches across teams of teams and systems of systems. Scaling agility is one of the biggest challenges currently facing organizations that want to become more agile. The SEMAT approach helps organizations scale agility in a number of ways:

- It establishes a common ground for all the teams involved. Scaled agility requires many teams to collaborate, working on the same systems and improving the same value flows. In this situation it is even more essential that all the teams have a shared understanding of what they are doing and a shared language to help them communicate.
- It allows teams to be flexible about their practices. Scaling agility requires even more flexibility in the set of practices that teams can use. Teams collaborating on the same system will need to share practices with one another. Teams working on certain systems will need to use some of the practices originally used to develop the system. SEMAT's ability to mix and match practices, swapping them in and out of play as needed, provides the flexibility in the way of working that teams need to succeed in a scaled agile environment.
- It helps teams understand their interaction points with other teams, the boundary of their responsibilities, and how their progress and health affects the teams they work with. If everybody is using a common ground to indicate their responsibilities and how they are progressing, then inter-team working is easily monitored and improved.

Select enterprise-level tooling. By providing a common ground for software development, SEMAT also provides a common ground for enterprise-level tooling. The separation of the shared common ground from the various practices used helps organizations understand which practice-independent tooling they need, which practice-specific tooling they need, and how these are related. SEMAT also helps teams understand how to integrate the tools they use by providing definitions of the common elements they will share.

Figure 1. Software development as a multidimensional endeavor.

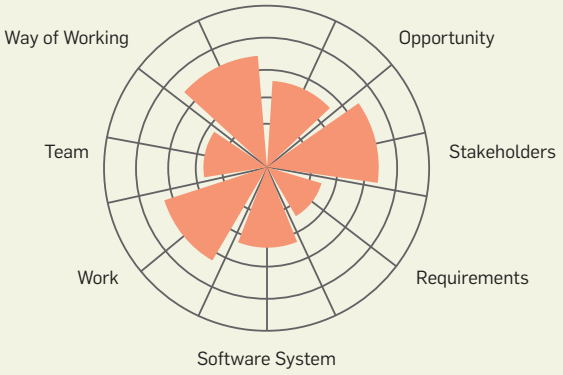
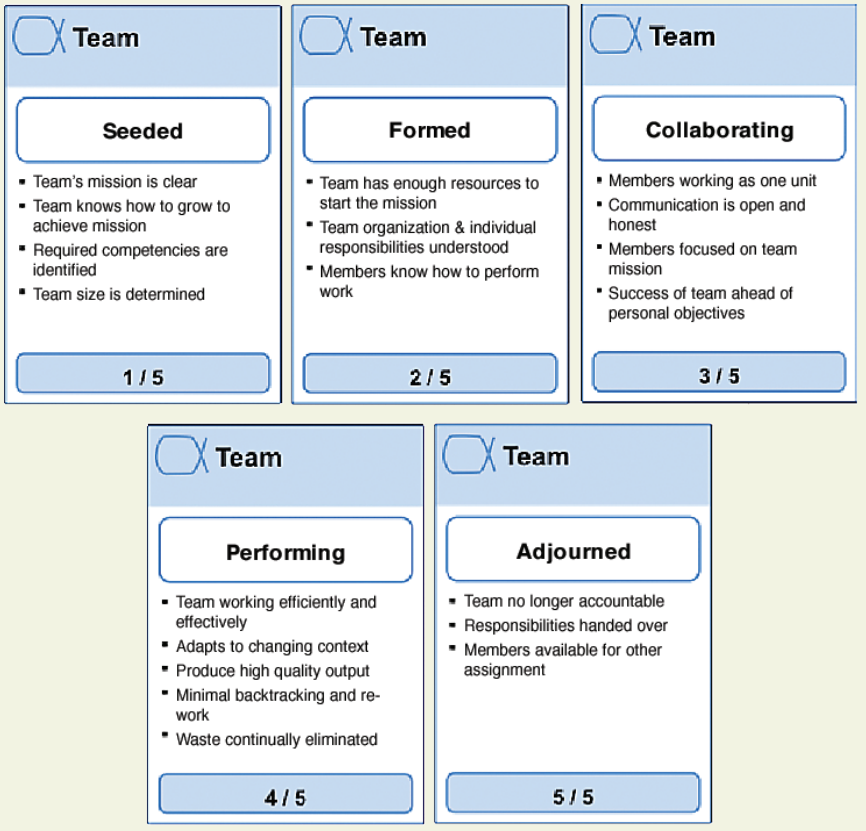


Figure 2. Alpha state cards with checklists.



What Agile Adds to SEMAT

SEMAT is nonprescriptive to such an extent it does not even insist upon adopting an agile approach. It does not care what approach a team adopts as long as it produces "good" software in an effective and healthy fashion.

Adopting agile values brings many benefits to teams and organizations—too many to go into in this brief article. For organizations adopting SEMAT, agility adds a number of important elements in the area of software-engineering methods, including:

- *Principles and values.* The addition of agile values to the SEMAT framework provides a necessary qualitative dimension to the evaluation of progress and health.
- *Many practices.* The agile community is a hotbed of new and innovative practices, all of which could be codified and made available as SEMAT practices for teams safely to compare, contrast, and mix and match.
- *A driving force for improvement.* Agility embeds the inspect-and-adapt cycle into every aspect of the team's work.

Before adopting SEMAT, a team should establish the principles and values it would like its new way of working to embody; otherwise, it will be very difficult to select the right practices or break out of what can at first appear to be an academic process-building exercise.

The brevity of this section, when contrasted to the earlier "What SEMAT Adds to Agile" section, is a reflection of the broader acceptance and knowledge of agility than SEMAT. It does not represent the relative value or impact of the two initiatives.

How Does SEMAT Do All This?

The goal of the SEMAT initiative is to provide software developers with a sound practical and theoretical foundation to improve their performance. (For more detailed information, see Jacobson et al.[3])

The first step in the SEMAT initiative is to establish a common ground for software professionals (developers, testers, among others) to stand upon when they talk about what they do. This common ground manifests itself in Essence, a kernel of universal elements in software development—elements prevalent in every development endeavor. Essence includes these elements: requirements, software system, work, team, way of working, opportunity, and stakeholders.

These elements have states, which can be used to measure progress and health. For example, a team can take the following states: seeded, formed, collaborating, performing, and adjourned. To achieve a particular state, a number of checkpoints must be fulfilled, representing real achievements. To achieve state collaborating, for example, the following checkpoints have been fulfilled: the team works as one cohesive unit; communication within the team is open and honest; the team is focused on achieving the team mission; and the team members know each other.

Traditionally, checkpoints have been used to measure the completion of an activity or a document, but

the SEMAT checkpoints measure outcome. Thus, the universal elements represent achievements rather than documents or artifacts. This makes them agnostic to any particular method—agile or not. These elements are called *alphas*.

Software development is multi dimensional, and alphas identify the typical dimensions every software-development endeavor must consider to progress in a healthy manner. A radar chart, as depicted in Figure 1, gives a view of the current progress along each dimension.[1] Each line originating from the center represents an alpha, and the radials on that line represent the current state for that alpha.

Essence also provides a lightweight approach to describe practices on top of the kernel and thus extend the kernel. From a library of practices, teams can select appropriate ones and compose them to get the way of working they are satisfied with. In this way, they can evolve their way of working over time by replacing existing practices with newer and better ones. Practices can be of different kinds—for example, business, social, or technical. Each practice can add guidance for moving an alpha from one state to another, or it can add alphas not included in the kernel. In this way the endeavor will add more dimensions. It can also add work products to each alpha it touches. For example, the use-case-driven development practice might add a use case as an alpha and use-case specifications and realizations as work products.

Cards and checklists. The Essence specification provides a detailed description of the kernel alphas, including the definition of their checkpoints. In its daily work, however, a team would not carry the Essence specification with it. A more concise and practical representation in the form of a deck of cards suffices. Figure 2 shows the state cards for the team alpha.

Each card has the name of the alpha at the top, followed by the state name and a concise list of checkpoints. These act as useful reminders for developers.

Boards and visuals. In addition to state cards, there are alternative ways of working with alphas—for example, an alpha abacus, as shown in Figure 3. An abacus is a Chinese calculation device with beads (counters) on a wire (representing digits). In the alpha abacus, each wire represents an alpha, and each bead an alpha state.

This visual board can be used for a variety of purposes. One possible use is for a team to evaluate its current state (where it is) and discuss its next objective (where it wants to go next). This is easily visualized by drawing imaginary lines and positioning the beads as shown in Figure 3.

Games. Once cards and visuals are available, it becomes natural to have games. For example, Progress Poker, a game that evaluates progress and health, is similar to the Planning Poker game used in agile methods. In Progress Poker, each member of the team selects a state card for each alpha to represent the current state of development. If they all choose the same state card, it means they have a common understanding of the progress. If they choose different cards, they probably have different understandings of where their development stands, and different expectations of what needs to be done. This misunderstanding usually signifies the presence of risks. Once it is discovered, team members can have further discussions to reach a consensus. Other games—Objective Go, Chasing the State, and so on—can be found at http://www.ivarjacobson.com/alphastatecards.

Case Studies

The case studies described in this section are good examples of how software-development teams can make good use of SEMAT and Essence.

Equipping Coaches in a Large Telecommunications Company

We worked with a large Chinese telecommunications-product company that had a number of internal coaches. The capabilities of these coaches were critical to each team's ability to improve. Equipping the coaches to detect development problems early was important. In our first contact with one of the coaches, we asked how his team was doing. He felt that progress was good. We then asked him to evaluate progress using a deck of alpha state cards. He laid the cards on the table and started shifting them and quickly identified that progress of the stakeholders alpha was slow. He recognized this was a risk and made it a point to work out a plan to address the risk, which was basically to achieve the first four states of the stakeholder alpha. The initial discussion with this coach took only 15 minutes. A further discussion found the coach came from a development background rather than a business-analysis background, which was probably the reason he neglected the stakeholders dimension.

In this particular case, the coach in question was weak in one area. In other cases, coaches had neglected other dimensions represented by the software-system alpha such as design and quality. In yet other cases, there were disagreements among team members about the way-of-working alpha. Whatever the case, the Essence alphas were simple, intuitive, and effective tools for evaluating progress and health.

Running Development in an Internet Media Product Line

The next case study involves several development teams in Beijing collaborating to deliver an Internet media server. This was a new product line, and the team members and leaders were relatively junior. They had much to learn, not just about how to work, but also about their problem domain. In addition, they were transitioning from a traditional stovepipe organization where testers and developers worked separately to one in which developers and testers collaborated as a cross-functional team.

Our approach involved using the kernel and the use-case-driven development practice[2] to design the team visualization board shown in Figure 4. This team visualization board provided visualization from three different perspectives:

- *Process.* This made the alphas visible to team members so they would know their current iteration objectives (that is, which kernel alpha states they needed to get to). This also included a section showing the current states for the use-case slices they are working on. A use-case slice is a piece of use case that represents a unit of work. The states of a use-case slice were described using state cards similar to those in Figure 1. This made the cri-

Figure 3. Alpha abacus.

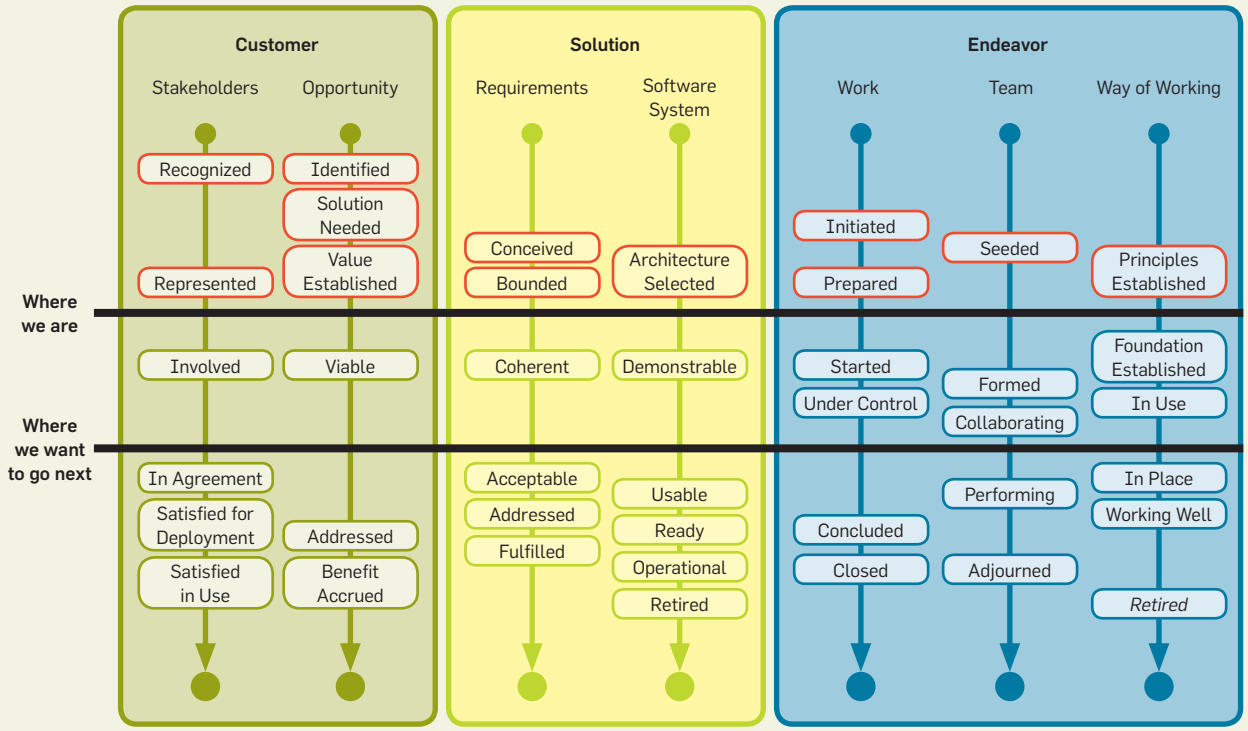
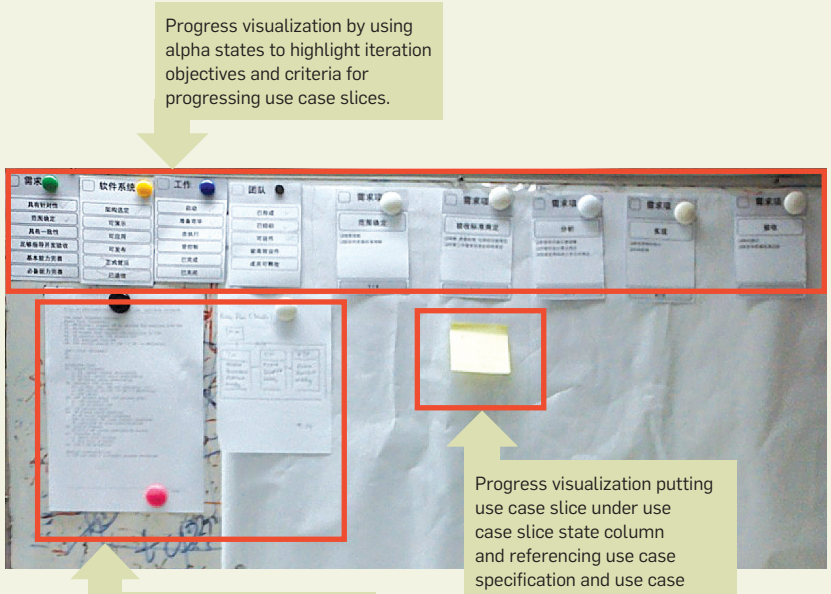


Figure 4. Team visualization board.



teria for achieving each state visible to team members during their daily work.

▸ *Product.* Each team was assigned a use case. The use-case specification and realization, represented by a UML diagram, were pasted on the team board and always represented the current agreement. Changes were scribbled onto the use-case specification and realization. If they became eligible (after some significant changes), someone on the team had to create a clean version.

▸ *Progress.* Post-it notes representing use-case slices were pasted on the board. During daily meetings, members working on the use-case slices would talk about their work in progress by referencing their slices against the requirements and design, as well as the "definition of done" from the process visuals.

Having process, product, and progress visuals readily available not only helped the junior members to understand quickly what they needed to do, but it also helped the team detect any misconceptions quickly.

Improving Collaboration Among Teams

This case study occurred in a Japanese consumer electronics product line of e-book readers. The company had three models, each with different capabilities such as Wi-Fi or 3G access, touchscreen, and so on. It had three product teams (one for each model) and three development teams, as well as an acceptance test team, user experience team, and hardware team. Each team had about four members. Because each team worked separately, coordination was poor, leading to bottlenecks.

We helped the teams make their development work visible through the use of alphas. Specifically, they identified two alphas: a use-case slice and a user-experience element. They defined states for these alphas and the checkpoints for achieving those states. They made the current states visible on a product-line visualization board similar to that in the previous case study, with two exceptions: it had a section for user-experience elements; and it encompassed the entire product line rather than a single team. This was possible because the number of members in each team was relatively small.

Team leaders used the product-line visualization board to plan and discuss progress. With the visualization board, they were able to look ahead and make necessary preparations. In this way each team could make the effort to complete their parts for each integration event, thus eliminating bottlenecks.

Quick Start for Offshore Collaboration

The final case study involved a Japanese company that started a new product line with the help of a Chinese offshore vendor providing development and testing. The product line evolved from an initial eight-person team, with whom we worked primarily, into 50 (local Japanese) plus 200 (offshore Chinese) members. These numbers excluded hardware development and local contractors (working on device drivers) who were an integral part of the overall development. This all occurred in the span of about two years.

This Japanese company had no described way of working, and the Chinese vendor's norm was to follow its client's approach, so there was no starting point. Using Essence, we were able to help the Japanese company describe a way of working that included these practices: iterative development, use-case-driven development, continuous integration, and test-driven development.

The next challenge was determining how to allocate parts of the development to the Chinese vendors. The Japanese company wanted this to be gradual so that as the Chinese members grew in their understanding, they could take on larger responsibilities. The allocation of responsibilities was based on both architecture and process. In terms of architecture, the Chinese vendors could work on the user interface and mid-tier areas, whereas the device drivers and processing closer to hardware specifics remained within the Japanese developers' responsibilities because this required highly specialized skill and the hardware was changing.

In terms of the development process, the alpha states provided a convenient way of discussing responsibilities and involvement. The development process involved several streams of work represented by the alphas. The progress through the requirements alpha states represents the main development. Two other alphas were added to represent work on architecture and acceptance.

In the beginning the Japanese client had primary responsibility over most of the alpha states. As the Chinese vendor grew in knowledge, it assumed greater responsibilities. The alpha states provided a simple means of agreeing on the collaboration. It is important to note that when the Chinese vendor assumed responsibility over one alpha state, it did not mean the Japanese shook off all involvement. The Japanese developers were still involved, but as assistants to the Chinese members.

Using Essence, the Japanese product-line organization could describe their processes, responsibilities, and involvement. It helped the teams get started. It also helped team leaders (both Japanese and Chinese) understand their areas of responsibilities quickly as development grew from eight people to 250.

A Firm Foundation for Sustainable Improvement

SEMAT and agile are two complementary—and perfectly aligned—initiatives. They are both nonprescriptive frameworks that help you think about and improve your software-development capability.

If you are serious about making sustainable improvements in your software-engineering capability, either within your team or across your whole organization, then the combination of Agile and SEMAT offers many benefits above and beyond those gained from either initiative alone. C

Related articles on queue.acm.org

The Essence of Software Engineering: The SEMAT Kernel
Ivar Jacobson, Pan-Wei Ng, Paul McMahon, Ian Spence, and Svante Lidman
http://queue.acm.org/detail.cfm?id=2389616

UX Design and Agile: A Natural Fit?
Terry Coatta and Julian Gosper
http://queue.acm.org/detail.cfm?id=1891739

Breaking the Major Release Habit
Damon Poole
http://queue.acm.org/detail.cfm?id=1165768

References

1. Graziotin, D. and Abrahamsson, P. A Web-based modeling tool for the SEMAT Essence theory of software engineering. *J. Open Research Software I*, 1 (2013), e4; http://dx.doi.org/10.5334/jors.ad.
2. Jacobson, I. and Spence, I. Use case 2.0: Scaling up, scaling out, scaling in for agile projects. Ivar Jacobson International, (2011); http://www.ivarjacobson.com/resource.aspx?id=1225.
3. Jacobson, I., Ng, P-W., McMahon, P., Spence, I. and Lidman, S. The Essence of software engineering: The SEMAT kernel. *ACM Queue* (Oct. 24, 2012); http://queue.acm.org/detail.cfm?id=2389616.

Ivar Jacobson, chairman of Ivar Jacobson International, is a father of components and component architecture, use cases, the Unified Modeling Language, and the Rational Unified Process. A contributor to modern business modeling and aspect-oriented software development, Jacobson is one of the leaders of SEMAT, working to renew software engineering as a rigorous discipline.

Ian Spence is Chief Scientist at Ivar Jacobson International, and has been involved in many large-scale agile adoptions over the years reaching thousands of people. He also led the work of the Essence kernal and has co-authored three software development books.

Pan-Wei Ng is the Asia Pacific CTO at Ivar Jacobson International. He is an advisor and coach to software development organizations of all sizes and has helped teams apply SEMAT ideas since its inception. He also invented the alpha state cards while coaching in Japan.