10 Developing with Essence

The goal of this chapter is to demonstrate how Essence can help teams find the best approach to development endeavors via selection of appropriate practices, mini-practices, and tools. Specifically, we show

- situations within software endeavor that require the team to be ready to resolve minor challenges during software development, because not everything runs smoothly in real endeavors;
- the importance of communication and teamwork during software engineering; and
- that "essentializing software engineering" means representing the way your team is working using the Essence language and the Essence kernel common ground.

Furthermore, using the example of iterative development performed by Smith and his team, we will also show

- what typically happens during parts of the development cycle that include planning, doing, checking, and adapting;
- the role of Progress Poker, Chase the State, and Objective Go when used in development planning; and
- the mechanisms to update the cards (by hand) when needed during the process.

To achieve the Way of Working: Foundation Established state, Smith's team needed to agree on their approach to development, which included selecting their key practices or mini-practices and tools. They agreed to work in an agile way, splitting up the whole endeavor into smaller mini-endeavors, each mini-endeavor resulting in progressing the work on the software to be built. This approach is called iterative development and each mini-endeavor is called an *iteration*. An iteration is a fixed period of time in which the team develops a stable piece of a software system.

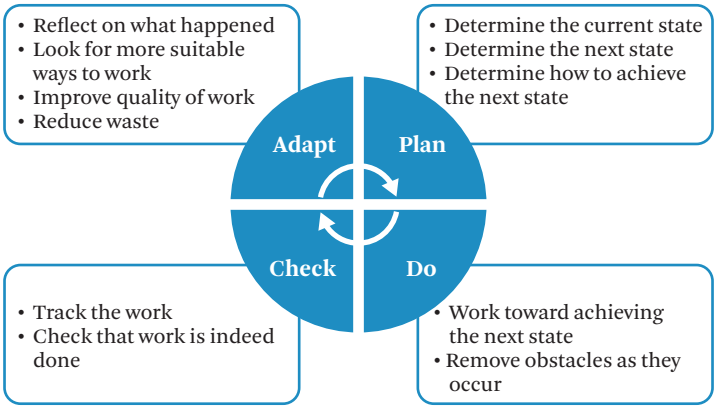


Figure 10.1 Plan-Do-Check-Adapt cycle.

The length of an iteration is typically two to four weeks and can involve all kinds of activities, including requirements gathering and the deployment of the resultant software system.

Developing your software iteratively is like taking a journey in your car. You need to know where you are, where you are heading, how much fuel you have, and how much further you have to go before reaching your destination. You adapt to the road conditions, traffic, and weather as you drive. You are continuously planning, doing, checking, and adapting[1] (see Figure 10.1). This is how Smith and his team ran their iterations. Each iteration was set to be one week in duration.

Plan. First, Smith's team would ascertain the current state of the whole endeavor by determining the current state of each of the alphas. Before their first weekly iteration, they decided to start by playing the Progress Poker game (Section 8.1). But after having done it for the first two alphas they continued to play the Chase the State (Section 8.2) game for the other five alphas. So Smith began by placing the first Alpha Overview card in his deck in the middle of the table. Each team member thought about which state they believed that the Requirements alpha was in and then placed the selected state card face down on the table. When everyone was ready, they turned the cards face up. After the team discussed the results and reached agreement on the state of the Requirements alpha, the team members

1. We modified Deming's PDCA cycle (https://en.wikipedia.org/wiki/PDCA), replacing Act with Adapt, as this is more descriptive of the intent.

picked up their cards. Together they had agreed that the Requirements had reached the Coherent state.

Smith then selected the Software System alpha and placed its Alpha Overview card in the middle of the table. He and his team repeated the same steps with this alpha and agreed it had reached the Architecture Selected state.

It turned out that the team wasn't needing to discuss too much to obtain consensus; they found that they were in agreement immediately. So Smith suggested instead to play the Chase the State game (Section 8.2) next. This open discussion would more quickly reveal the state for each of the remaining five alphas.

For the Stakeholder alpha, although the team agreed that they knew Dave and Angela were their stakeholders, they still had not yet agreed on how they would get them involved. Thus, the Stakeholder alpha continued to be in Represented state. For the Opportunity alpha, the team agreed that they had now achieved the Solution Needed state, but no one thought they had achieved the Value Established state. At this point, Joel and Tom started to discuss how they might go about convincing senior management of the value of the endeavor, but Smith quickly interrupted, saying, "Wait. We are just conducting the game now to agree where we are. Let's hold off discussing which states we need to focus on next and how we will achieve them. We will do that after we have fully assessed where we are now and can then decide what is most important to do next." Everyone agreed and they continued with Chasing the State to reach agreement on the other alphas (Work: Prepared, Team: Formed, and Way of Working: Foundation Established); see Figure 10.2.

After finishing the games, knowing where they were, they discussed and agreed to what alphas and states to progress in the coming iteration. This was done by the team playing the Objective Go game (see Section 8.3), Smith decided to start the Objective Go game saying, "We now know what state we are in for each of the seven kernel alphas. Now we need to agree on which states to focus on achieving in our next iteration." Joel decided to cut to the chase and asked, "So does everyone agree that the most important thing in the next iteration is to convince management of the value of our endeavor?" All of the team members nodded in agreement. Smith then said, "This means we have concurred that we need to achieve the Opportunity: Value Established state in the next iteration."

The team continued to discuss, and reached agreement on which states in the other alphas they needed to achieve in the next iteration. For one thing, they needed to get their key stakeholders, Dave and Angela, involved (meaning reaching the Stakeholder: Involved state). They also agreed that the requirements for the upcoming iteration were coherent, which means they were consistent, and that

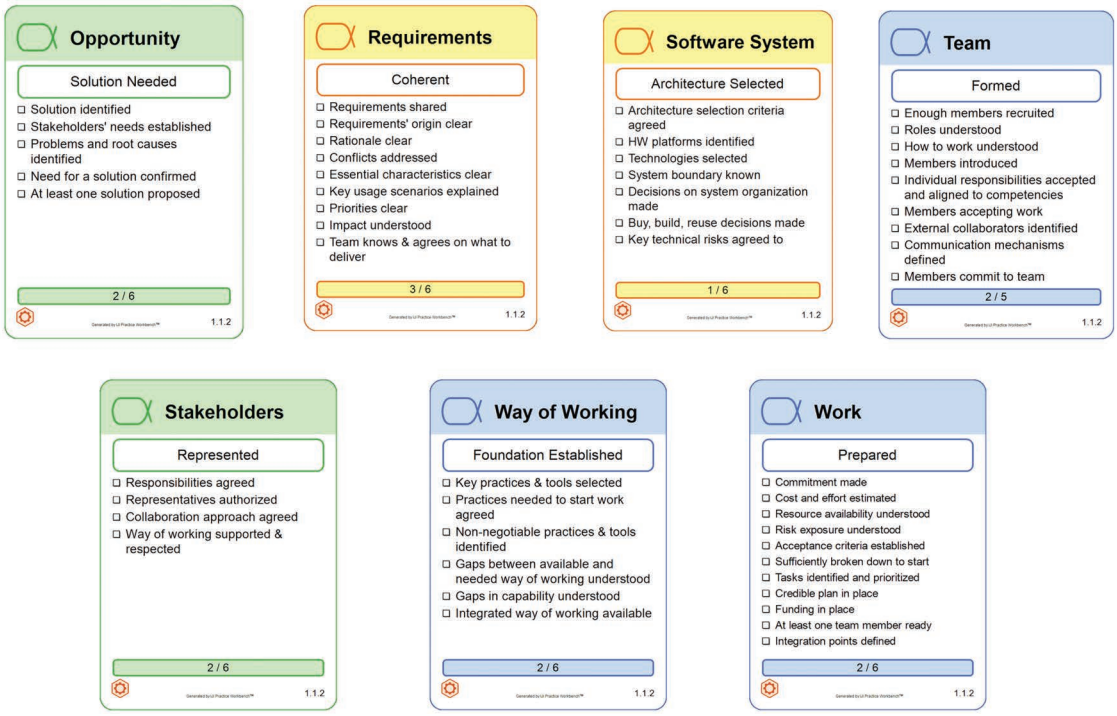


Figure 10.2 The alpha states agreed on after playing the Progress Poker and Chasing the State games.

they needed to address them by demonstrating the implementation of those requirements to Dave and Angela. (That is, they needed to achieve the Requirements: Addressed state.) They also agreed the team needed to be collaborating, the work must be started, and the way of working needed to be in use; see Figure 10.3.

Together they then planned how to achieve the target states by identifying which tasks, if completed, would achieve these states. For example, one specific part of the work they needed to do was to set up a meeting with Dave and Angela to discuss how they would get them involved. They also knew they needed to set up a test environment.

This all enabled them to connect their detailed day-to-day work with the progress of the endeavor as a whole. If the effort to complete the tasks exceeded that available in the iteration, then it would take more than a single iteration to achieve the objectives and the target states. This means the team would need to break their tasks down further and agree on the pieces to complete in the current iteration. For instance, they knew they could not get all four requirement items done in the first

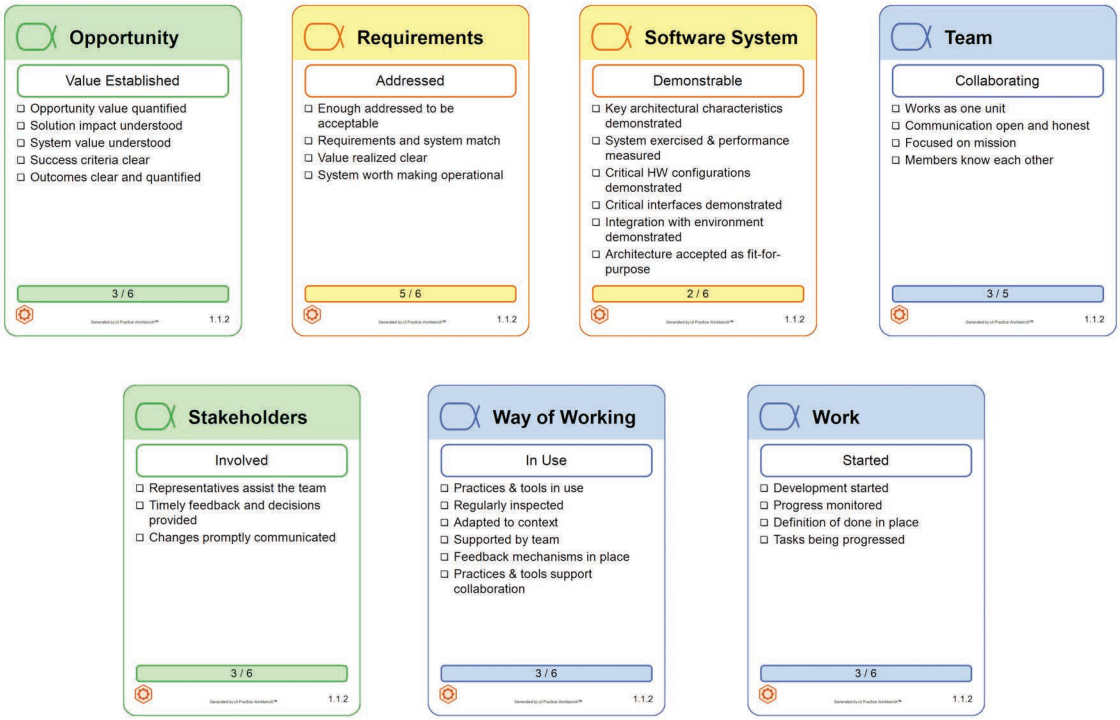


Figure 10.3 Result after applying the Objective Go game.

iteration, so they concurred to work on just the first three (for the four requirements items mentioned here see Figure 9.6 above).

Do. Each iteration, Smith's team worked on the identified tasks to progress the endeavor as a whole toward the target states. This involved setting up environments, discussing requirements, capturing agreements, writing code, testing, and so on; see Figure 10.4.

Check. Smith's team tracked the objectives and tasks to make sure that they were completing what they had planned as they used their agreed-on way of working. The team discussed the healthiness of all alphas; unhealthy meant the alpha had a checklist that had not yet been met but should have been met, or that the checklist had previously been met, but was no longer met due to some change in the condition of their endeavor. The team placed green stickers next to the alpha state cards they had agreed represented healthy alphas. They also agreed that anyone could

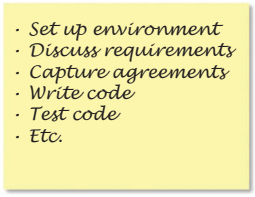


Figure 10.4 Task list.

place a red sticker next to a card if they felt during the iteration that an alpha had become unhealthy.

Adapt. Smith's team reviewed their way of working, identified obstacles, and found better or more suitable ways of doing things. This often resulted in changes to their plans and their way of working.

10.1 Planning with Essence

When you plan an iteration, the alphas can help you understand where you are and where to go next. By aligning the objectives for each iteration, Smith's team made sure that they progressed in a balanced and consistent way. The alphas helped by reminding them what was essential for success as the team decided what was most important to focus upon next.

As mentioned, Smith and his team agreed that they would work in cycles where each iteration was one week. It was the first day (Monday) of the first iteration week. Using Essence, they reviewed their current state and the states they had previously agreed that they wanted to achieve by the end of the first iteration. Based on that, they identified a set of tasks. In what follows, you will find task descriptions that were agreed on by the team after discussing each alpha, the state they had achieved, and the next target state(s). The first state on the left in each diagram is the current state, and the state(s) listed after the arrow is (are) the target state(s).

Sometimes we have more than one target state for a given alpha. This is because teams often work to achieve checklist items in more than one state at the same time. When a team works iteratively, they often are working to get some of the requirement items both coherent and addressed in the same iteration. For example, when our TravelEssence team was working on Req-Item #1, they needed to get the code to actually produce recommendations on hotels, and they needed to get clarification on how far from the traveler's current location they should search for possible hotels.

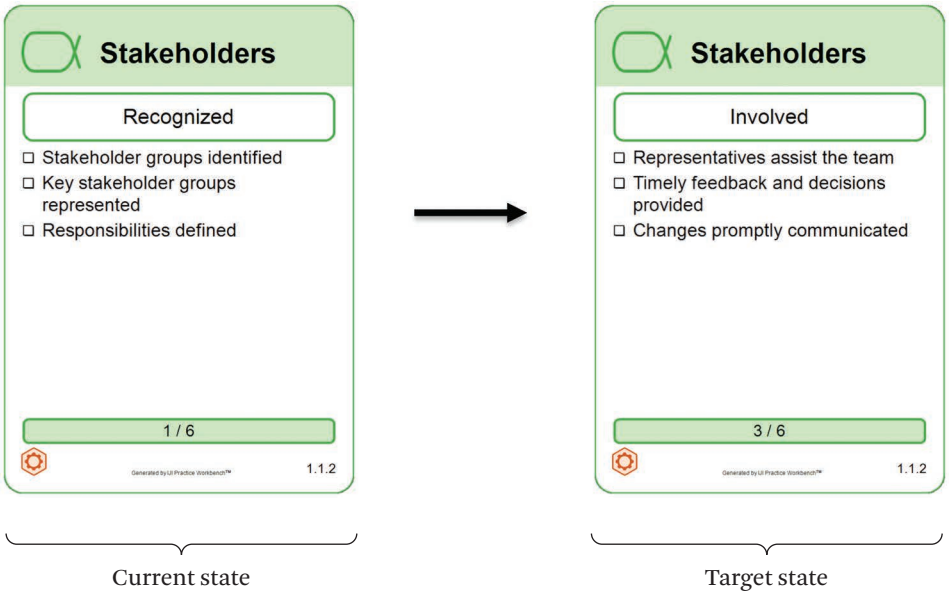


Figure 10.5 Stakeholders current and target states.

As we established in Chapter 6, the alphas fall into three areas of concern: Customer (indicated by green cards and notes), Solution (indicated by yellow cards and notes), and Endeavor (indicated by blue cards and notes). We will look at the planning in these terms; first we will consider it from the perspective of the Customer area of concern, comprising the Stakeholders and Opportunity alphas.

Stakeholders: Recognized → Involved (see Figure 10.5).
The team had identified two of their key stakeholders as Dave and Angela, and that there was not yet agreement on their involvement and commitment. As mentioned, they had agreed to set up a meeting with Dave and Angela to clarify their involvement.

(Task: Stakeholder involvement meeting)

Opportunity: Solution Needed → Value Established (see Figure 10.6).
A solution was needed to exploit the travelers' data, but what the solution required was still up in the air. The value of that solution still needed to be established to convince senior management at TravelEssence to move forward and fund the effort. The team's immediate priority was to set up a test environment where they could quickly experiment with different ideas for using the travelers' existing data

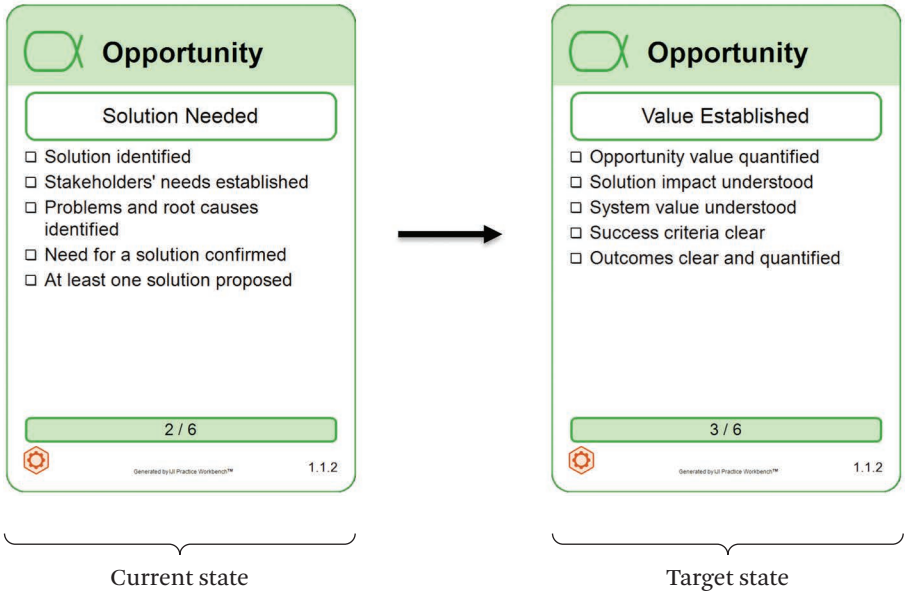


Figure 10.6 Opportunity current and target states.

to generate more traveler interest, leading to increased revenue. This could help them quantify the value of the new system to Dave and Angela.

(Task: Experiment with different ideas to increase business.)

Next, we look at planning from the perspective of the Solution area of concern, comprising the Requirements and Software System alphas.

Requirements: Conceived → Coherent, Addressed (see Figure 10.7).

The team determined two target states for this alpha: Coherent and Addressed. To achieve their objectives in the current iteration, they would need not only to get three requirement items into the Coherent state, but also move these requirement items to the Addressed state. In the discussion that follows, you will see how the team could work toward these two target states at the same time.

Smith already had a simple requirement list. In the first iteration, Smith and his team would attempt to complete the following requirement items:

Req-Item #1. System generates recommendations for a traveler

Req-Item #2. Mobile plug-in displays recommendations

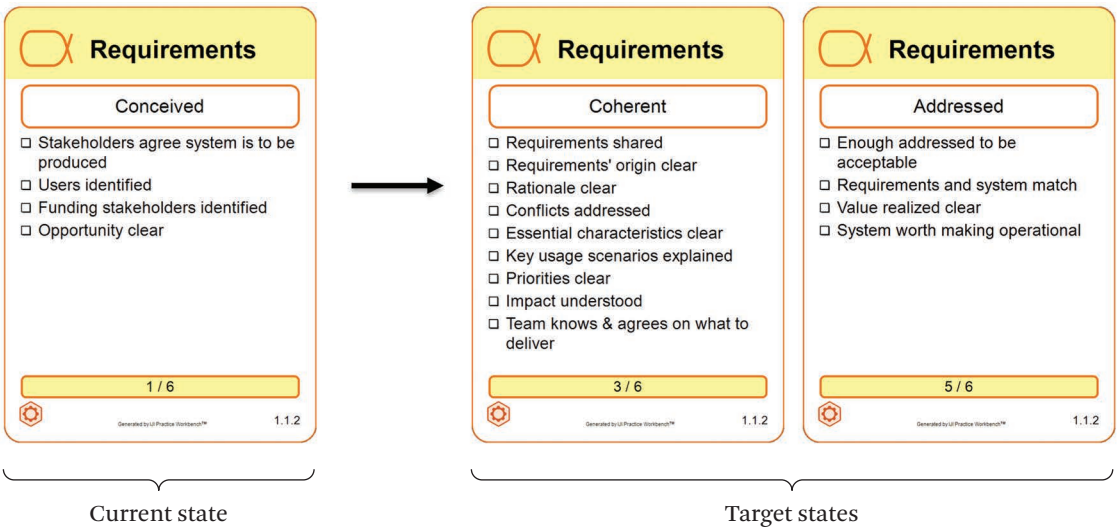


Figure 10.7 Requirements current and target states.

Req-Item #3. System handles user's selection to view or discard recommendations

Of course, some details needed clarification, such as finding the algorithm (calculation formula) to generate a recommendation, and determining which target set of travelers they would use as a test data set. This meant the next target state for requirements was to get to the Coherent state by working on these issues. The team agreed that this would be achieved by Smith collaborating with Angela to reach agreement on the plan. Once agreed, the team would need to move forward quickly to address these requirements for the upcoming planned demonstration (discussed below under Software System). This means they needed to move their requirement items to the Addressed state.

(Task: Smith to work with Angela to reach agreement on recommendation algorithm, and which set of travelers they would use as their test data set)

Software System: Architecture Selected → Demonstrable (see Figure 10.8).
To get to the Demonstrable state, the team would need to code, test, and integrate critical parts of the system and demonstrate the results to Angela. The team agreed to work on their respective requirement items and integrate their work by Wednesday evening to be ready for a demo to Angela by Friday.

(Task: Team members work on implementing their respective requirement items)

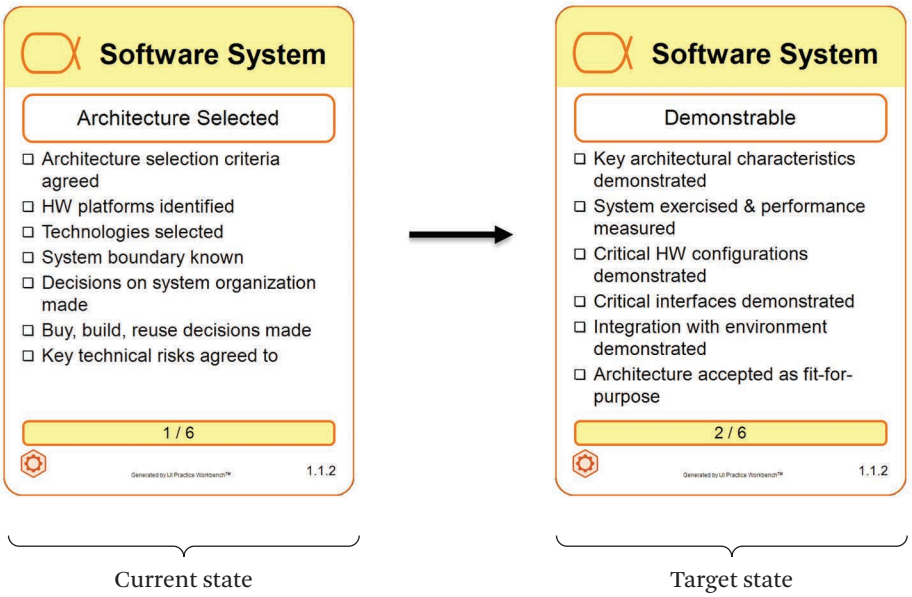


Figure 10.8 Software System current and target states.

Last, we look at this iteration's planning from the perspective of the Endeavor area of concern, comprising the Work, Team, and Way of Working alphas.

Work: Initiated → Prepared, Started (see Figure 10.9).
To get to the Work: Prepared state, the team needed to make sure their tasks were broken down into sufficiently small pieces to fit in the agreed iteration, understand any related risks, and be sure they had a credible plan in place that extended beyond the current iteration. While Tom, Joel, and Grace focused on the planning for the first iteration, Smith reviewed the work the team had agreed to do. As part of Req-Item #1, the team had discussed providing recommendations for both hotels and restaurants, but Smith decided this was too much for the first iteration and suggested the team limit the work for now to just providing hotel recommendations.

(Task: Team breaks work down to fit in iteration)

Team: Formed → Collaborating (see Figure 10.10).
Smith's team members had successfully worked together before. They each knew their responsibilities and how they would work together, but the team had not yet showed that it was working as one cohesive unit. By setting the goal of integrating

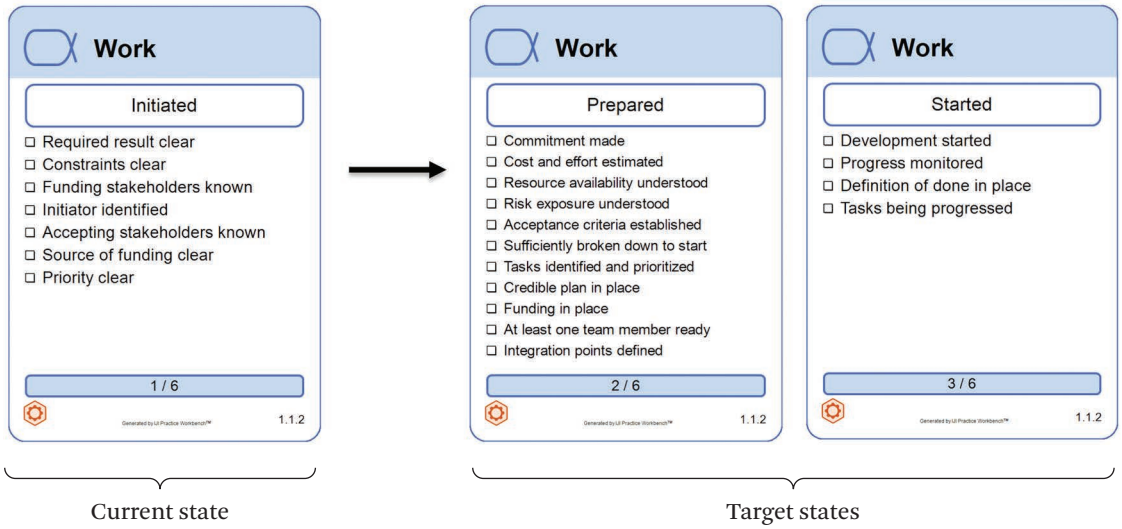


Figure 10.9 Work current and target states.

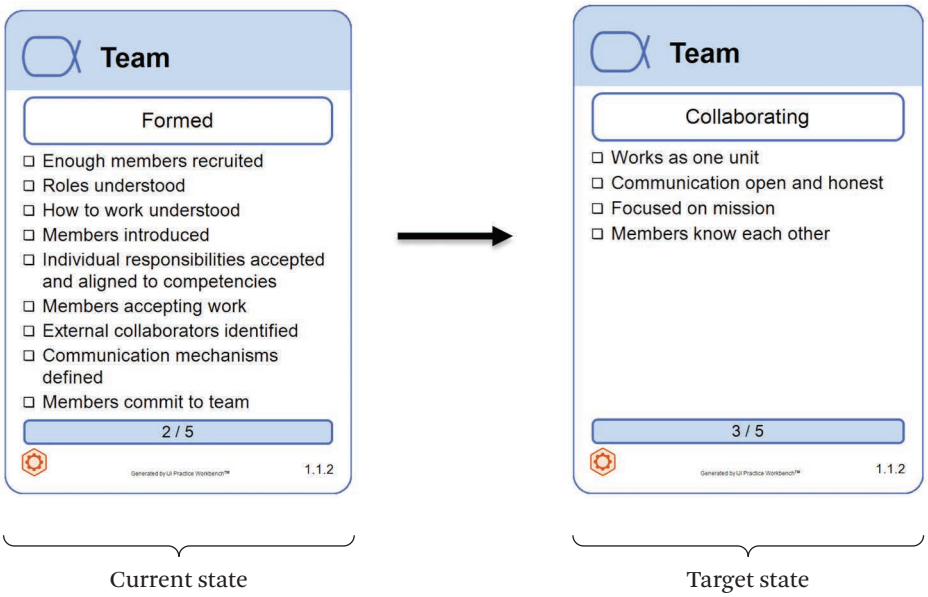


Figure 10.10 Team current and target states.

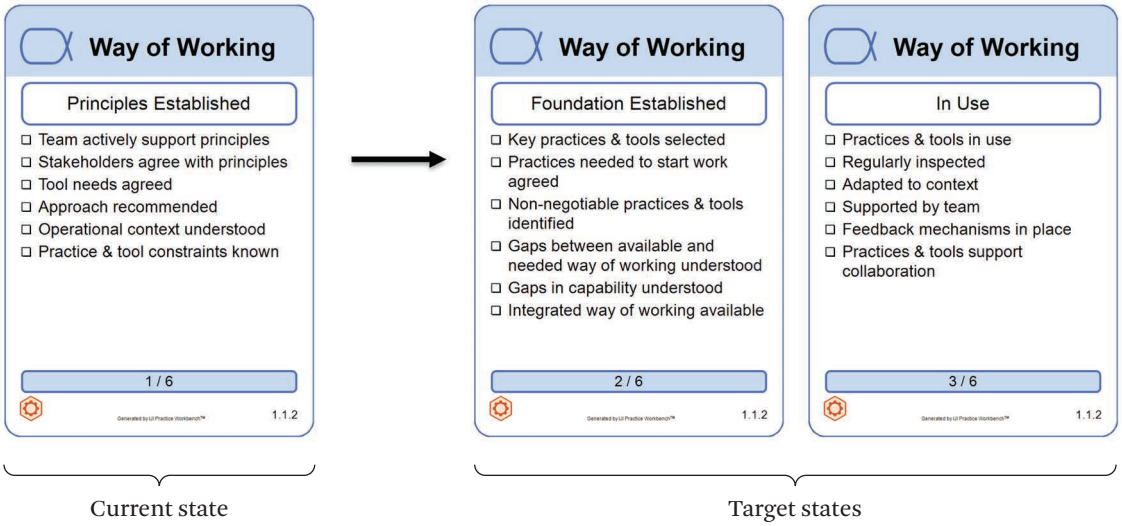


Figure 10.11 Way of Working current and target states.

their work by Wednesday, they would be able to verify that they were working as one cohesive unit (Task: Integrate work by Wednesday).

Way of Working: Principles Established → Foundation Established, In Use (see Figure 10.11).

To get to Foundation Established, the team needed to establish a development and test environment. Tom agreed to set up the development environment that included the team's chosen repository version control tool and the test environment. Grace agreed to prepare the test environment and supporting scripts. Following the setup of the environment, the team would start to use it to complete their tasks during the first iteration (Task: Establish development and test environment).

10.2 Doing and Checking with Essence

With the goals (expressed as target alpha states) and the tasks identified, Smith's team proceeded to work on their respective tasks.

Smith and his team members were co-located: sitting near each other in the work place. Angela's work area was located on a different floor, near Dave. TravelEssence had an internal corporate chat application for collaboration, so all of them could access each other when needed. In general, work went rather smoothly, as the members were familiar with each other and the technology they were using.

On Friday afternoon, they did indeed achieve their goals and demonstrated the implementation of the designated requirement items to Angela. They reviewed their health and progress by playing the Chase the State game again and comparing the results to the previous time. In the following results, the target state for each alpha is listed in parentheses, with the actual results of the team's effort described afterward.

From the perspective of the Customer area of concern:

Stakeholders (Involved). Following the Friday demo, Smith had a side meeting with Angela and Dave, where they discussed and agreed to their involvement in future demonstrations.

Opportunity (Value Established). The Friday demonstration was successful and convinced Dave and Angela that the system could potentially produce significant user interest, leading to increased business. As a result, Dave was ready to move forward and fund the effort.

From the perspective of the Solution area of concern:

Requirements (Coherent, Addressed). The team had successfully clarified the open issues related to the agreed requirements, and then they successfully addressed those requirements in the Friday demonstration.

Software System (Demonstrated). The team successfully demonstrated the critical parts of the system that had been agreed to for the Friday demonstration.

From the perspective of the Endeavor area of concern:

Work (Prepared, Started). The team had successfully broken their agreed tasks down for the first iteration, understood the risks, and moved forward coding, testing, and integrating the pieces in preparation for the Friday demonstration.

Team (Collaborating). The team successfully integrated their work on Wednesday in preparation for the Friday demo. This activity verified that the team was working as one consistent unit.

Way of Working (Foundation Established, In Use). The team had successfully gotten their environment set up and used it during the first iteration to complete the work for the Friday demonstration.

10.3 Adapting a Team's Way of Working with Essence

The kernel clearly helped the team capture and apply the essence of software engineering. It reminded them to

- involve key stakeholders;
- think about the opportunity;
- break the work down to fit in the agreed way of working;
- think about risk;
- clarify requirements;
- integrate each team member's work with teammates' work; and
- focus on the most important things first.

But there will still always be better ways of doing things. So after the successful Friday demonstration, the team decided to discuss what went well, what did not go so well, and how they could do better during their next iteration.

During this discussion, Smith reminded the team of their agreed-on target alpha states from the first iteration; see Figure 10.3.

Smith then asked the following questions:

- What went well with our planning, doing, and checking related to the above alpha states?
- What did not go well with our planning, doing, and checking related to these alpha states?
- What can we do better with our planning, doing, and checking related to the alpha states?

Joel said, "We achieved a successful demonstration and now Dave is going to fund our full endeavor, so that certainly went well."

Tom said, "The way to achieve the Requirements: Addressed state was not clear to me at the start of the iteration. I learned that I had to talk to Angela and get her to agree to the requirement items to be implemented. I didn't understand this just by looking at the state checklist."

Grace said, "Actually, Smith, for me to do my job better, I would like to have better guidance regarding how to work on a requirement item."

Smith first considered Tom's request. That was easy; all Smith had to do was to supplement the state checklist with some additional guidance. He scribbled two

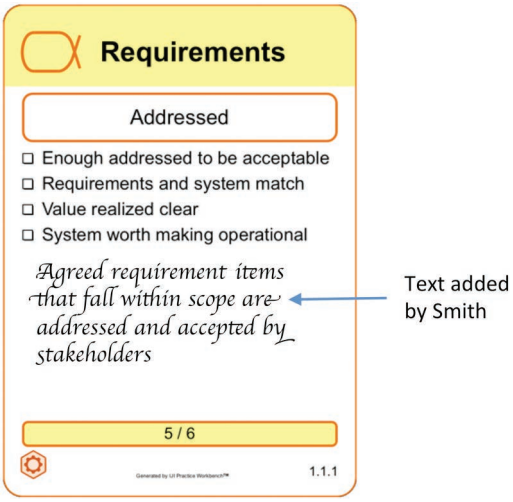


Figure 10.12 Additional guidance beyond the standard on achieving a state.

lines of text onto the Requirements card as follows (see Figure 10.12):

- gain agreement on requirement items that are within scope of Addressed; and
- implement these requirement items.

These notes were additional guidance on how to achieve the Addressed state.

Then Smith considered Grace's request. This request was not as easy. It meant making the way of working on all requirement items more explicit. We will discuss how to do this in Part III.

The way of working affects all team members, and every team member can contribute. This is another area where the kernel is useful. By talking about the alpha states after their successful demonstration, as we have just observed, the team came up with a number of good ideas on how they could do better during the next iteration.

10.4 How the Kernel Helps Adapt Their Way of Working

The kernel helps teams adapt their way of working in multiple ways.

10.4.1 Helping a Team Reason about Their Way of Working

First, it helps a team reason about their way of working and decide if there are improvements they should make.

Developers who come straight from an educational program often know more about programming than developing software, and more about developing software than working as a team and improving their way of working. Because their experience is limited, they often need a little help. The alphas and their states can help a team reason about their way of working as they try to improve.

When reviewing their way of working, we make the alpha states visible to the team members to help them think about their "process." If you are conducting a review of your way of working for an iteration, you only need to make visible those states relevant to the current iteration (i.e., the iteration's target states). This helps the team stay focused on reviewing just their way of working related to that iteration.

By visualizing the states, a mental transition takes place. The team is now looking at the "process." We then look at each state specifically, and ask the same questions.

- What went well during this iteration, and have we achieved this alpha state?
- What did not go well during this iteration, and do we know what is keeping us from achieving this alpha state?
- What can we do better in the next iteration that will help us achieve this alpha state?

10.4.2 Making Changes to the Way of Working

The daily contact your team has with the alpha states (and hence the kernel) help you find simple improvements to adapt your team's way of working. This may mean adding additional items to the alpha state checklist to meet your team's needs. Teams can also define new alphas or add checklists to help team members, such as the text Smith added to Requirements: Addressed to help Tom. How to extend the kernel elements further with more explicit practices is discussed in Part III.

Keep in mind that the team should only add information to the kernel elements and their checklist items. Changing the information that is already there would undermine the value that we gain through the use of a standard kernel. The standard is the basis for essentializing methods and practices, a subject discussed in Part I. You will learn more about essentializing in Part III, namely how to express explicit practices using the Essence language.

You can just think of "essentializing software engineering" as representing the way your team is working using the Essence language and the Essence kernel common ground. This can help your team understand more clearly what missing elements they need to focus on in order to be successful in their endeavor.

What Should You Now Be Able to Accomplish?

After studying this chapter, you should be able to

- explain the terms iteration and iterative development;
- explain the activities involved within iterative development (i.e., planning, doing, checking, and adapting);
- explain the use of individual card games within the planning, doing, checking, and adapting parts of the iteration cycle, and be able to apply them;
- explain the role of discussion and team agreement during these activities;
- explain the role of "healthy alpha" and its use in examining alpha states; and
- give examples of the "Way of Working" and means to adapt them during development, meeting the needs of the team and the endeavor.

References

Alpha State Card Games. 2018. https://www.ivarjacobson.com/publications/brochure/alpha-state-card-games. 99, 153

S. Ambler and M. Lines. 2012. *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*. IBM Press. 297, 339, 346

K. Beck. 1999. *Extreme Programming Explained: Embrace Change.* Addison-Wesley Longman. 203, 285, 346

K. Beck. 2003. *Test-Driven Development by Example*. Addison Wesley. 346

K. Bittner and I. Spence. 2003. *Use Case Modeling.* Addison-Wesley Professional, 2003. 222, 285

G. Booch, J. Rumbaugh, and I. Jacobson. 2005. *The Unified Modeling Language User Guide.* 2nd edition. Addison-Wesley. 222, 285, 345

F. Brooks. 1975. *The Mythical Man-Month*. Addison Wesley. 342

M. Cohn. 2004. *User Stories Applied: For Agile Software Development.* Addison-Wesley Professional. 204, 247, 285

E. Derby and D. Larsen. 2006. *Agile Retrospectives: Making Good Teams Great.* Pragmatic Bookshelf, Dallas, TX, and Raleigh, NC. 196, 198, 284

E. W. Dijkstra. 1972. "The Humble Programmer." Turing Award Lecture, *CACM* 15 (10): 859–866. DOI: 10.1145/355604.361591. 342

D. Graziotin and P. Abrahamsson. 2013. A web-based modeling tool for the SEMAT Essence theory of software engineering. *Journal of Open Research Software*, 1,1(e4); DOI: 10.5334/jors.ad. 147, 153

ISO/IEC/IEEE 2382. 2015. Information technology–Vocabulary. International Organization/International Electrotechnical Commission, Geneva, Switzerland. https://www.iso.org/obp/ui/#iso:std:iso-iec:2382:ed-1:v1:en. 343

ISO/IEC/IEEE 12207. 2017. https://en.wikipedia.org/wiki/ISO/IEC_12207 345

ISO/IEC/IEEE 15288. 2002, 2008, 2015. Systems and software engineering—System life cycle processes. International Standardization Organization/International Electrotechnical Commission, 1 Rue de Varembe, CH-1211 Geneve 20, Switzerland. 345

ISO/IEC/IEEE 24765. 2017. Systems and software engineering–Vocabulary. International Organization/International Electrotechnical Commission, Geneva, Switzerland. https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:24765:ed-2:v1:en. 343

M. Jackson. 1975. *Principles of Program Design*. Academic Press. 344

I. Jacobson. 1987. Object-oriented software development in an industrial environment. *Conference Proceedings of Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 87)*. DOI: 10.1145/38807.38824. 221, 285

I. Jacobson and H. Lawson, editors. 2015. *Software Engineering in the Systems Context*, Systems Series, Volume 7. College Publications, London. 345

I. Jacobson and E. Seidewitz. 2014. A new software engineering. *Communications of the ACM*, 12(10). DOI: 10.1145/2685690.2693160. 347

I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard. 1992. *Object-Oriented Software Engineering: A Use Case Driven Approach*. ACM Press Addison-Wesley. 345

I. Jacobson, I. Spence, and K. Bittner. 2011. Use-Case 2.0: The Guide to Succeeding with Use Cases. https://www.ivarjacobson.com/publications/whitepapers/use-case-ebook. 169, 222, 226, 233, 285

I. Jacobson, P.-W. Ng, P. E. McMahon, I. Spence, and S. Lidman. December 2012. The essence of software engineering: The SEMAT kernel. *Communications of the ACM*, 55(12). http://queue.acm.org/detail.cfm?id=2389616. DOI: 10.1145/2380656.2380670. 30, 95

I. Jacobson, P.-W. Ng, P. E. McMahon, I. Spence, and S. Lidman. 2013a. *The Essence of Software Engineering: Applying the SEMAT Kernel*. Addison-Wesley. xxvi, 30, 90, 95, 336

I. Jacobson, I. Spence, and P.-W. Ng. (October) 2013b. Agile and SEMAT: Perfect partners. *Communications of the ACM*, 11(9). http://queue.acm.org/detail.cfm?id=2541674. DOI: 10.1145/2524713.2524723. 30, 96

I. Jacobson, I. Spence, and B. Kerr. 2016. Use-Case 2.0: The hub of software development. *Communications of the ACM*, 59(5): 61–69. DOI: 10.1145/2890778. 169, 222, 226, 285

I. Jacobson, I. Spence, and P.-W. Ng. 2017. Is there a single method for the Internet of Things? *Queue*, 15.3: 20. DOI: 10.1145/3106637. 335, 339

P. Johnson and M. Ekstedt. 2016. The Tarpit—A general theory of software engineering. *Information and Software Technology* 70: 181–203. https://www.researchgate.net/profile/Pontus_Johnson/publication/278743539_The_Tarpit_-_A_General_Theory_of_Software_Engineering/links/55b4490008aed621de0114f5/The-Tarpit-A-General-Theory-of-Software-Engineering.pdf. DOI: 10.1016/j.infsof.2015.06.001. 91, 92, 96

P. Johnson, M. Ekstedt, and I. Jacobson. September 2012. Where's the theory for software engineering? *IEEE Software*, 29(5). DOI: 10.1109/MS.2012.127. 84, 87, 96

R. Knaster and D. Leffingwell. 2017. *SAFe 4.0 Distilled: Applying the Scaled Agile Framework for Lean Software and Systems Engineering.* Addison-Wesley Professional. 297, 339

P. Kruchten. 2003. *The Rational Unified Process: An Introduction*. 3rd edition. Addison-Wesley. 345

C. Larman and B. Vodde. 2008. *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum.* Pearson Education, Inc. 346

C. Larman and B. Vodde. 2016. *Large-Scale Scrum: More with LeSS.* Addison-Wesley Professional. 297, 339

D. Leffingwell. 2007. *Scaling Software Agility: Best Practices for Large Enterprises.* Addison-Wesley. 346

P. E. McMahon. January/February 2015. A thinking framework to power software development team performance. *Crosstalk, The Journal of Defense Software Engineering*. http://www.crosstalkonline.org/. 96, 154

NATO. 1968. "Software Engineering: Report on a conference sponsored by the NATO Science Committee." P. Naur and B. Randell, editors. Garmisch, Germany, October 7–11. 342

S. Newman. 2015. *Building Microservices.* O'Reilly Media, Inc. 250, 285

P.-W. Ng. 2013. Making software engineering education structured, relevant and engaging through gaming and simulation. *Journal of Communication and Computer* 10: 1365–1373. 99, 153

P.-W. Ng. 2014. Theory based software engineering with the SEMAT kernel: Preliminary investigation and experiences. *Proceedings of the 3rd SEMAT Workshop on General Theories of Software Engineering*. ACM. DOI: 10.1145/2593752.2593756. 30, 96

P.-W. Ng. 2015. Integrating software engineering theory and practice using Essence: A case study. *Science of Computer Programming*, 101: 66–78. DOI: 10.1016/j.scico.2014.11.009. 96, 152, 154

Object Management Group. Essence—Kernel and Language for Software Engineering Methods (Essence). http://www.omg.org/spec/Essence/1.1. 63

OMG Essence Specification. 2014. http://www.omg.org/spec/Essence/Current. 95, 346

D. Ross. 1977. Structured Analysis (SA): A language for communicating ideas. In *IEEE Transactions on Software Engineering*, SE-3(1): 16–34. DOI: 10.1109/TSE.1977.229900. 344

K. Schwaber and J. Sutherland. 2016. "The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game." Scrum.org.

Index

Author Biographies

Ivar Jacobson

Dr. Ivar Jacobson received his Ph.D. in computer science from KTH Royal Institute of Technology, was awarded the Gustaf Dalén medal from Chalmers in 2003, and was made an honorary doctor at San Martin de Porres University, Peru, in 2009. Ivar has both an academic and an industry career. He has authored ten books, published more than a hundred papers, and is a frequent keynote speaker at conferences around the world.

Ivar Jacobson is a key founder of components and component architecture, work that was adopted by Ericsson and resulted in the greatest commercial success story ever in the history of Sweden (and it still is). He is the creator of use cases and Objectory—which, after the acquisition of Rational Software around 2000, resulted in the Rational Unified Process, a popular method. He is also one of the three original developers of the Unified Modeling Language. But all this is history. His most recently founded company, Ivar Jacobson International, has been focused since 2004 on using methods and tools in a smart, superlight, and agile way. Ivar is also a founder and leader of a worldwide network, SEMAT, whose mission is to revolutionize software development based on a kernel of software engineering. This kernel has been realized as a formal standard called Essence, which is the key idea described in this book.

Harold "Bud" Lawson

Professor Emeritus Dr. Harold "Bud" Lawson (The Institute of Technology at Linköping University) has been active in the computing and systems arena since 1958 and has broad international experience in private and public organizations as well as academic environments. Bud contributed to several pioneering efforts in hardware and software technologies. He has held professorial appointments at several universities in the USA, Europe, and the Far East. A Fellow of the ACM, IEEE, and INCOSE, he was also head of the Swedish delegation to ISO/IEC JTC1 SC7 WG7 from 1996 to 2004 and the elected architect of the ISO/IEC 15288 standard. In 2000, he received the prestigious IEEE Computer Pioneer Charles Babbage medal award for his 1964 invention of the pointer variable concept for programming languages. He has also been a leader in systems engineering. In 2016, he was recognized as a Systems Engineering Pioneer by INCOSE. He has published several books and was the coordinating editor of the "Systems Series" published by College Publications, UK.

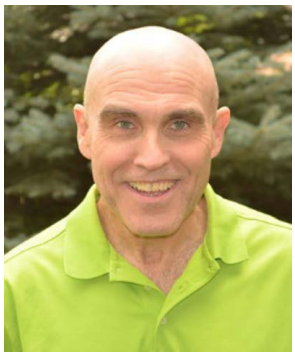
Tragically, Harold Lawson passed away after battling an illness for almost a year, just weeks before the publication of this book.

Pan-Wei Ng

Dr. Pan-Wei Ng has been helping software teams and organizations such as Samsung, Sony, and Huawei since 2000, coaching them in the areas of software development, architecture, agile, lean, DevOps, innovation, digital, Beyond Budgetings, and Agile People. Pan-Wei firmly believes that there is no one-size-fits-all, and helps organizations find a way of working that suits them best. This is why he is so excited about Essence and has been working with it through SEMAT since their inception in 2006, back when Essence was a mere idea. He has contributed several key concepts to the development of Essence.

Pan-Wei coauthored two books with Dr. Ivar Jacobson and frequently shares his views in conferences. He currently works for DBS Singapore, and is also an adjunct lecturer in the National University of Singapore.

Paul E. McMahon



Paul E. McMahon has been active in the software engineering field since 1973 after receiving his master's degree in mathematics from the State University of New York at Binghamton (now Binghamton University). Paul began his career as a software developer, spending the first twenty-five years working in the US Department of Defense modeling and simulation domain. Since 1997, as an independent consultant/coach (http://pemsystems.com), Paul helps organizations and teams using a hands-on practical approach focusing on agility and performance.

Paul has taught software engineering at Binghamton University, conducted workshops on software engineering and management, and has published more than 50 articles and 5 books. Paul is a frequent speaker at industry conferences. He is also a Senior Consulting Partner at Software Quality Center. Paul has been a leader in the SEMAT initiative since its initial meeting in Zurich.

Michael Goedicke

Prof. Dr. Michael Goedicke is head of the working group Specification of Software Systems at the University of Duisburg-Essen. He is vice president of the GI (German National Association for Computer Science), chair of the Technical Assembly of the IFIP (International Federation for Information Processing), and longtime member and steering committee chair of the IEEE/ACM conference series Automated Software Engineering. His research interests include, among others, software engineering methods, technical specification and realization of software systems, and software architecture and modeling. He is also known for his work in views and viewpoints in software engineering and has quite a track record in software architecture. He has been involved in SEMAT activities nearly from the start, and assisted in the standardization process of Essence—especially the language track.