

Article development led by [acmqueue](http://queue.acm.org)
queue.acm.org

Essence can keep software development for the IoT from becoming unwieldy.

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

Is There a Single Method for the Internet of Things?

The Industrial Internet Consortium predicts the Internet of Things (IoT) will become the third technological revolution after the Industrial Revolution and the Internet Revolution. Its impact across all industries and businesses can hardly be imagined. Existing software (business, telecom, aerospace, defense, among others) will likely be modified or redesigned, and a huge amount of new software, solving new problems, will be developed. As a consequence, the software industry should welcome new and better methods.

This article makes the case that to be a major player in this space you will need a multitude of methods, not just a single one. Existing popular approaches

such as Scrum and SAFe (Scaled Agile Framework) may be part of the future, but you will also need many new methods and practices—some of which are not even known today. Extending a single method to incorporate all that is required would result in something that is way too big and unwieldy. Instead, the new Object Management Group (OMG) standard Essence can be used to describe modular practices that can be composed together to form a multitude of methods, not only to provide for all of today's needs, but also to be prepared for whatever the future may bring.

The software world is continuously innovating and opening up new areas of opportunity and challenge. A decade ago developers were busy with trends such as service-oriented architecture and product-line architecture—still very much around, but now a commoditized part of a larger system-of-systems landscape, and also extended to cloud computing with big data and mobile applications. New software development approaches have accompanied these new trends, most of them being agile in different flavors and size: Scrum, Kanban, DAD (Disciplined Agile Delivery), SAFe, LeSS (Large-scale Scrum), and SPS (Scaled Professional Scrum) being among these approaches.

These trends have impacted the software industry in many different ways—producing more pervasive and powerful technology-based products, for example. None of them, however, has had a truly transformational or radically disruptive impact.

The Industrial Revolution in the 19th century moved us from essentially building things as a craft to manufacturing. The Internet Revolution at the end of the 20th century was another such transformation of the world or, as Bill Gates said in 1999, “A fundamental new rule for business is that the Internet changes everything.” The Internet has driven the need for faster turnaround time with less precise requirements—hence, sparking the trend toward light-



weight, empirical, and iterative methods. It has also driven the rise of social networking, which places the Internet at the heart of everyone's life.

Now the Industrial Internet Consortium³ claims that the IoT, building on the cloud, mobile Internet, big data, and so on, is a third such fundamental transformation. The Industrial Internet Consortium was founded in March 2014 by ATT, Cisco, General Electric, IBM, and Intel to remove roadblocks to widespread adoption of the IoT. Its mission is "to accelerate growth of the Industrial Internet by coordinating ecosystem initiatives to connect and integrate objects with people, processes and data using common architectures, interoperability and open standards that lead to transformational business outcomes."

The IoT touches everything. What is it then about the IoT that will dramatically change the business model for all industries? Here is an example: Traditionally, a company sells a product and, as long as all goes well,

doesn't know what happens to it once it has left the factory gate. By connecting the product via sensors to the IoT, the manufacturer can fundamentally change its value proposition. Instead of selling only assets, the manufacturer can sell services, including the assets, which enables it to build long-term relationships with its customers. For example, suppliers of aircraft engines can offer their products as a service (the fee based on the number of flying hours). The effect is that the supplier is now highly motivated to keep the machines running, since otherwise it will lose revenue, and the airline can increase its revenue since it will have reduced downtime. There are many similar examples from both government and industry. Basically, every industry will be affected, including banking, insurance, telecoms, airlines, and defense.

As voiced by Alex Sinclair, CTO of the GSMA: "We believe that with the right standards and regulation in place it will have a fundamental im-

pact on the way we live and work, reducing waste and inefficiencies and delivering major social and environmental benefits in security, health care, transportation and logistics, education and energy, amongst many other sectors of the economy."¹³ The IoT will eventually reach all areas where humans are providing products or services, both today and in the future. Moreover, it will use all of the kinds of systems in use today: communication, mobile, distributed, big data, cloud computing, among others, and it will drive new technologies not yet seen.

To be successful, companies will need to be able to respond quickly to the changing demands of the network while maintaining appropriate levels of engineering discipline, particularly for the cloud-based services upon which the distributed devices will depend. Moreover, the space to be addressed covers all levels of complexity—from very simple software running on basic sensors and other simple devices to

the high-performance, highly reliable, highly governed, secure, resilient, scalable systems needed to process, analyze, and respond to the vast amounts of data they produce, and everything in between. Not only that, the rate of change and the need for innovation will never have been higher.

The IoT Needs Everything

The IoT does not lack methods. Researching the space shows clearly, and not surprisingly, that there is not a one-size-fits-all approach. Instead, methods for waterfall and Agile, methods for small applications (apps) and for complex systems of systems, and methods for systems engineering (that is, for systems with hardware and software integrated) are all still needed. What is really new is that a larger vendor needs all this at the same time and with compressed time scales, which increases complexity significantly. Thus, for larger vendors a multitude of methods are needed. A smaller vendor needs a more specific and focused approach, but one that can grow as new products evolve and new problems emerge. Thus, methods such as Rational Unified Process (RUP) and SAFe, and practices such as Scrum, user stories, and use cases are all being applied. As always with any new trend, new branded methods are born. Literature regarding methods for the IoT is extremely sparse at the time of this writing. We have found two methods within the domain: Ignite¹³ and the IoT Methodology.²

The Ignite IoT Methodology. Ignite is an enterprise methodology for a major player in the IoT. It is a “big method” covering all aspects of developing for the IoT. It has two major practice areas. (In this article, *practice* is defined as a repeatable approach to doing something with a specific purpose in mind.⁹ Practices are the things that practitioners actually do.) These areas are *strategy execution* and *solution delivery*. Strategy execution is about agreeing what to build (that is, the solution) and involves the practices of opportunity identification, opportunity management, and initiation. Solution delivery is about delivering the solution to users, and it has a life cycle consisting of planning, building, and running (that is, operating the solution). Planning involves project initiation, whereas building and running are carried out through parallel project workstreams.

Project initiation is a set of practices that results in a number of different artifacts, including solution sketches, a milestone plan, user interface mockups, and software architecture. Project workstreams consist of a complementary set of practices (called workstreams): project management, cross-cutting, solution infrastructure and operations, back-end services, communication services, on-asset components, and asset preparation.

At a high level, these might seem to all be very general practices, but embedded within are two domain-specific practices: project dimensions (PD) and asset-integration architecture

(AIA). The intention is that the PD practice should be used to conduct project self-assessments, compare different IoT options, and select the solution architecture and technologies to be used in a project. The AIA practice is then used to identify the devices, gateways, and services, and their responsibilities for an enterprise solution. Ignite provides a set of technology patterns (such as machine-to-machine connectivity, and sensor networks, among other).

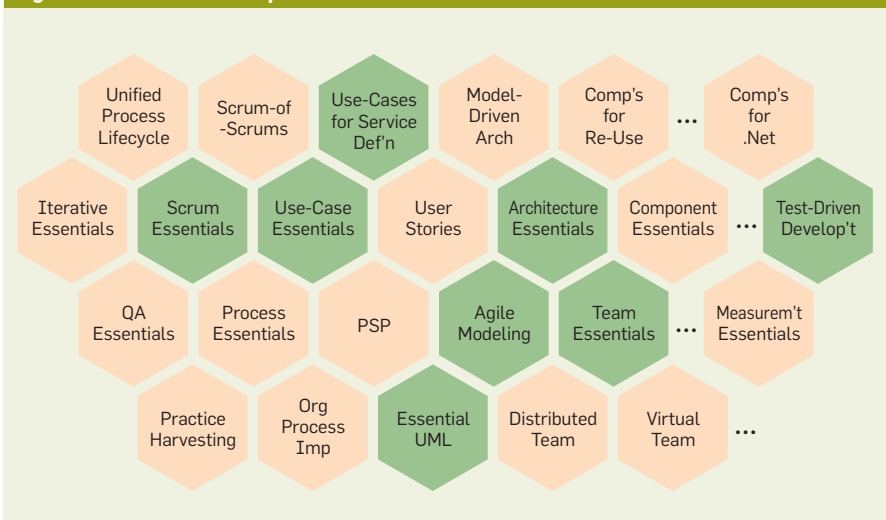
The benefit of Ignite is that it is based on real-world experience, capturing this experience and best practice in a well-thought-out and comprehensive methodology. Naturally, the first thought of the authors of the methodology was not so much about the modularity of the practices described but about the completeness and relevance of the method as a whole.

The IoT Methodology. In comparison, the IoT Methodology² is a lightweight method highly inspired by lean startup¹² and design thinking.¹ It involves the following iterative steps:

1. *Co-create.* Communicate with end users and stakeholders to identify pain problem areas in a nontechnical way.
2. *Ideate.* Simplify discussions to communicate requirements to designers, implementers, and project managers.
3. *Question and answer.* Translate soft concepts into hard requirements, analyze solutions, and brainstorm options.
4. *Map IoT OSI* (Open Systems Interconnection). Map requirements to a valid architecture, infrastructure, and business frameworks, similar to the layered approach used in the ISO/OSI model.
5. *Prototype.* Use standardized toolkits to build prototypes and iterate toward minimal viable products.
6. *Deploy* continuously to close the feedback loop and improve the products.

Like Ignite, this seems to be a very generic method at the high level. What’s so special about IoT Methodology is its use of an IoT Canvas and an IoT OSI reference architecture. The IoT Canvas is an adaptation of the business model/lean canvas used in brainstorming sessions to validate minimal viable product requirements for IoT projects. The IoT OSI reference model is an adaptation of the

Figure 1. An abundance of practices.



seven-layer ISO/OSI reference model for use with IoT solutions. This IoT OSI reference model consists of five layers, with endpoints at the bottom, connectivity, middleware, IoT services, and, finally, applications at the top. Stakeholders and developers use the IoT Canvas and IoT OSI reference model to co-create and co-evolve a solution definition before prototyping.

The IoT Methodology has taken agile thinking as a starting point but is also a monolithic method.

New Practices Are Needed

It is clear from these methods, and our own experience handling emerging technologies, that new domain-specific practices will be needed to handle the very nature of the IoT—particularly practices to handle these concerns:


- ▶ *Distributed.* These systems are typically far more distributed than most other software systems. Experience from the development of telecommunication systems will come into play: new failure modes (due to communications), reliability engineering, redundant systems development, and so on.

- ▶ *Mobile.* Again telecommunication vendors have practices to develop mobile systems, which are applicable. For example, these systems have to degrade gracefully, security is critical, and they must be robust.


- ▶ *Human out-of-the-loop.* The whole idea of the IoT is to sense/analyze/activate without a human in the loop—for example, self-driving cars, automated trading systems, and population health integration systems. There may be practices to be designed here, around reliability, failure management/failover, and exception condition management.

What isn't needed are new management practices.

Both Ignite and IoT Methodology are monolithic methods that reuse many existing generic practices, combining these with new innovative practices specifically for the IoT—sadly, in a way that makes the new practices difficult to reuse and share. This issue can be easily fixed, however, by taking them to the next level by *essentializing* them and freeing their practices. This means capturing the essence



Essence provides a common framework for describing all practices and then composing them into many methods.



of a practice, which consists of the things to work with, things to do, and competencies and patterns to provide minimal explicit guidance to apply the practice effectively. This does not just make the practices more accessible, but it also makes them easier to learn, change, and use for teams that adopt them. Later, we look at how one of them—the Ignite Methodology—could be essentialized.

The IoT Needs Essence

As discussed previously, the IoT requires many methods and practices, some of them specific to the domain and others that are generally accepted good software development practices. For example, they need to deal with specific problems of distribution and mobility, yet at the same time they must be grounded in sound architecture practices.

Essence and practices. The software development world has already identified and described hundreds of different practices, some of which are shown in Figure 1. Those shaded in green are selected for an IoT team. In an ideal world teams would be able to select the set of practices they need to address their current situation and easily assemble them into a method. For example, a team building software for the IoT with a high level of engineering complexity and a high rate of change may choose to base their method on the practices highlighted in green using Use Case and Architectural Essentials to provide the required engineering rigor, and Scrum and Agile Modeling to cope with the high rates of change.

The problem is these practices come from different sources and do not share the common ground needed to allow them to be readily composed into an effective method. This isn't a problem unique to the IoT; it is a problem that has been plaguing the software industry since its inception and one that gets worse with every advance in technology.

How can teams be empowered to own and control their methods while providing them with the guidance they need to be successful, and reflecting the owning organization's need for governance and compliance? How can teams benefit from the growing

number of proven practices while continuing to innovate and rise to the new challenges that they face every day? These are issues that particularly affect companies moving into the IoT, as they will need a variety of methods.

What is needed is some concrete common ground that the practices can share, providing both a shared vocabulary for practice definition and a framework for the assembly and analysis of methods.

This will allow organizations to prepare a library of practices suitable for their industry/domain—practices that teams can easily share, adapt, and plug and play to create the innovative ways of working that they need to excel and improve.

This common ground has already been prepared in the form of the Essence kernel, part of the new OMG

standard Essence,⁹ which provides a foundation that allows teams to share and free the practices from the shackles of monolithic methods.

Essence provides the following:

- ▶ A kernel of elements that establishes a common ground for carrying out software engineering endeavors and assembling methods

- ▶ A simple, easy-to-understand, visual, intuitive language for describing practices that can be used both to represent the kernel and to describe practices and methods in terms of the kernel

By combining these capabilities, Essence provides a common framework for describing all practices and then composing them into many methods.

The power of Essence in addressing the method complexity inherent in developing software for the IoT comes from its ability to enable the composition of practices into methods; help clearly define life cycles and checkpoints, enabling practice-independent governance; and support the creation of practice libraries from which practices can be selected to be composed into methods.

Let's now look at each of these in more detail.

Composing practices into methods.

In the past, different methods have primarily been described as isolated, conceptual islands. Every method is basically a unique phenomenon, described in its own language and vocabulary and not standing on any widely accepted common ground. Any method, however, may be considered to be composed from a number of practices.

For example, the agile method of extreme programming (XP) is described as having 12 practices, including pair programming, test-driven development, and continuous integration. Scrum, on the other hand, introduces practices such as maintaining a backlog, daily scrums, and sprints. Scrum is not really a complete method, though; it is a composite practice built from a number of other practices designed to work together. Scrum can itself be composed with other practices from, say, XP, to form the method used by an agile team. This composition is typically done tacitly, as Scrum and XP are not provided in a format that allows them to be explicitly composed.

As discussed previously, Essence provides a framework and language for describing and composing practices. This framework provides a practice architecture where, as shown in Figure 2, both generic and domain-specific practices are described and assembled on top of the Essence kernel.

Now individual practices can be described using Essence. A practice can be expressed by extending the kernel with practice-specific elements, by describing the activities used to progress the work and the work products produced, and by describing the specific competencies needed to carry out these activities.

Liberating practices in this way is very powerful. Once practices are codified in Essence, teams can take ownership of their way of working and start to assemble their own methods. This can start with even a simple library of practices, as shown in Figure 3.

This capturing and sharing of practices, both generic and domain-specific, in a way that lets them be applied alongside popular management practices (agile or otherwise), provides the raw materials that teams need to compose their own ways of working.

Figure 2. The essence practice architecture.

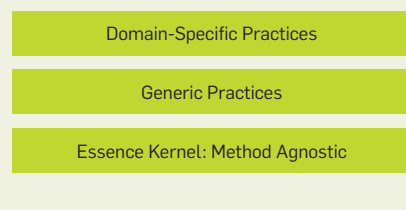
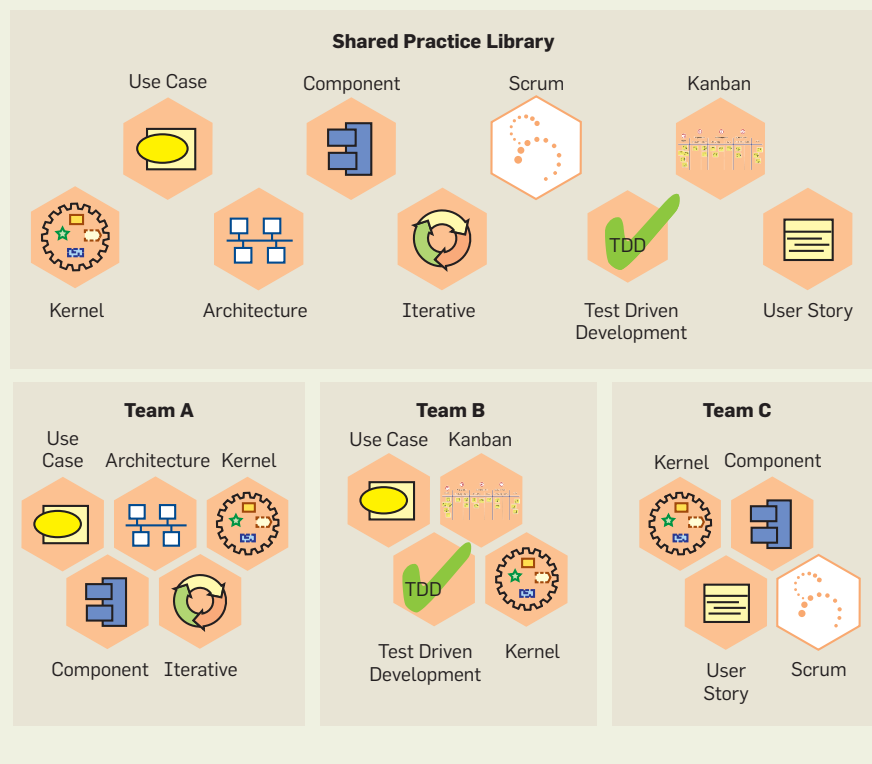


Figure 3. Three teams sharing a simple practice library.



Bringing a set of practices into this common system also allows gaps and overlaps to be more easily identified. The gaps can then be filled with additional practices and the overlaps resolved by connecting the overlapping practices together appropriately.

Governance and compliance. The Essence kernel allows you to define life cycles easily in a practice-independent way. Having a selection of different life cycles is incredibly useful when tackling a domain as complex as the IoT—particularly when the life cycles can be combined with whichever set of practices the team wants to use, ensuring that appropriate governance is applied without compromising the other aspects of the team’s way of working.

Using the Essence kernel makes it very easy to assemble a number of life cycles, each built using the same building blocks but addressing a different context and containing its own contextualized checkpoints. For example, Munich Re¹¹ defined a family of life cycles, each addressing a different context:

- ▶ *Exploratory*: A lightweight agile development life cycle for experiments, proof of concept, and small creative endeavors

- ▶ *Feature growth*: A rigorous engineering life cycle to support rapid feature growth with a strong architectural foundation

- ▶ *Maintenance and small enhancements*: A lightweight life cycle to enable the continuous flow of small enhancements and bug fixes for a fixed, funded period of time (typically a year)

- ▶ *Support*: A support-focused life cycle to aid in the transition between the development and support organizations

The ability to capture checkpoints and life cycles in a practice-independent way is incredibly powerful. It liberates the practices, allowing them to be used where appropriate and not constraining them to any predefined type or style of development. It also makes it possible to address the entire IoT methods space with a minimal, extensible, evolving set of practices, and allows teams to get the help they need without compromising their agility or engineering rigor.

Introducing Essence

The new Object Management Group (OMG) standard Essence⁹ is designed to support organizations and communities in becoming learning organizations with empowered teams that own their own ways of working and share their practices.

In addition to liberating the practices by enabling them to play well together, Essence does the following:

- ▶ Makes methods significantly lighter by focusing on the essentials.
- ▶ Helps teams measure progress and health in a method-independent way.
- ▶ Allows organizations to build a library of practices from which teams can select the ones needed for a particular solution (some teams need a “big” method, while others need only a small one).
- ▶ Helps organizations build “forever” learning organizations.

Essence provides a foundation for software engineering methods. This foundation helps in two ways: enables teams to understand and visualize the progress and health of their endeavors, regardless of their ways of working; and, allows teams to easily share, adapt, and plug and play their practices to create the innovative ways of working that they need to excel and continuously improve.^{6,7}

It guides *developers* in achieving measurable results and reusing their knowledge in systematic ways.

It helps *executives* lead programs and projects in balanced ways, without more governance than necessary, and develop learning organizations.

Note that Essence is generic enough to support a waterfall life cycle, as well as agile approaches.

Building a practice library. It is easy to see how the use of Essence would readily allow the assembly of a comprehensive practice library containing all the practices needed for a particular domain in a way that empowers teams to select just the practices they need to build their methods. Over the past few years, working in many areas of software development, including embedded systems, financial systems, telecommunications, modems, and many other areas affected by the IoT, Ivar Jacobson International (IJI; <https://practicelibrary.ivarjacobson.com/start>) has built an Essence-based practice library. Its library caters to both the craft and engineering ends of the development spectrum.

The practice library is constantly evolving as more and more practices are captured in the Essence language. At press time, IJI has essentialized close to 30 practices, including:

- ▶ Agile essentials, such as daily stand-ups, product ownership, and agile retrospectives,
- ▶ Common agile practices such as Scrum, user stories, and continuous flow,
- ▶ Proven architectural practices such as Use-Case 2.0, architectural essentials, and component-based development, and
- ▶ Life cycles such as the ones defined by Munich Re, discussed earlier.

Parallel to these efforts, existing methods such as dynamic systems development method (DSDM) and the Unified Process are being essentialized.^{8,10} An essentialized method is first structured in terms of its inherited practices, and then each practice is essentialized without changing its original idea.

All of these practices are built on top of the kernel and can be assembled to prime the pump for the methods that your teams will use. For example, organizations have used these practices to create lightweight agile methods, robust software engineering methods, pull-based flow methods, and flexible method families. They have been used to create both agile and waterfall methods that share many of the same practices but apply them with a very different emphasis.

What is powerful here is that these methods all share the same foundation and can adapt to changing circumstances by dropping and adding practices. The methods can also share practices, helping the teams—and the software they produce—to align and collaborate with one another.

To make the practices accessible and easy to learn, they are all available in card and electronic formats. Easy-to-use tools are available for practice and card creation, for method composition and publication, and for practice exchange and community building. These tools make it easy to extend ex-

isting practices to meet your needs and local standards, add your own practices, define practice-independent life cycles, and build your own frameworks and methods.

This allows you to leverage not just the industry best practice captured in the IJI practices, but also your own best practices, be they technical, financial, motivational, or managerial.

Building a Practice Library for the IoT

Examining the practices found in Ignite helps illustrate how to add domain-specific practices to a practice library.

Ignite describes a number of IoT-specific practices, including the AIA practice discussed earlier. Today, the generic practices in Ignite are not described in any detail, a gap that can easily be addressed by reusing the generic practices available in the IJI practice library.

Essentializing Ignite in this way helps distinguish the IoT-specific practices in a way that allows them to be adopted separately and applied alongside whatever generic practices the team or commissioning organization deems to be the most appropriate.

New domain-specific practices. By their very nature, the practices in the IJI practice library are very generic and applicable to many software-development domains. These generic practices are useful for many kinds of software, including for the Internet of Things.

The specific practices from Ignite and IoT Methodology are useful domain-specific practices that help address specific challenges for IoT applications. In addition, practitioners would have to work with specific technologies such as EPC (Electronic Product Code) to identify smart objects over an RFID network communicating with REST (representational state transfer) interfaces.⁵ Thus, there would be other domain-specific practices to use EPC and REST correctly.

Let's take a peek at how domain-specific practices are added to the practice architecture. A method has many aspects, such as team collaboration, how to manage requirements, architecture, and so on. In the discussion to follow, as shown in Figure 4, the focus is on architecture aspects because IoT ap-



The IoT will eventually reach all areas where humans are providing products or services, both today and in the future.



plications, with their high distribution and ubiquity, require serious attention to architecture.

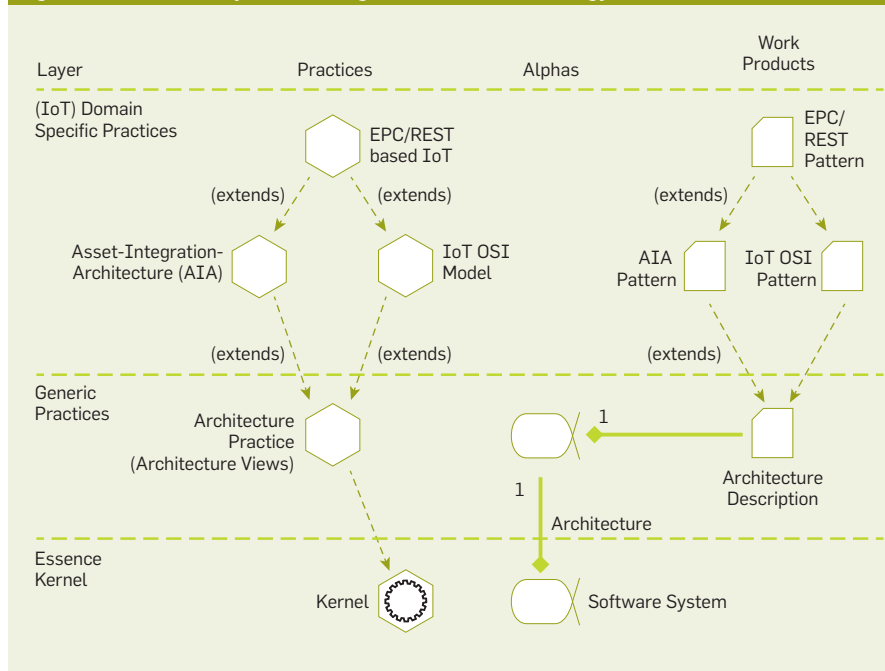
At the kernel layer, Essence provides guidelines for working with the software system. IJI's generic library has a practice for working with architecture, including guidelines for creating a sound architecture description (a work product) in an agile and lightweight manner. The Ignite method recommends using AIA as a way to describe architecture, and IoT Methodology recommends using its IoT OSI model. An application that uses EPC and REST would have technology specifics about how to name products and connections and so on.

Let's dive into the practices identified in Figure 4. The Essence language specifies a number of constructs. For brevity, this article illustrates only alphas and work product. An *alpha* is "an essential element that is relevant to an assessment of the progress and health of a software engineering endeavor."⁹ The alphas provide descriptions of the kinds of things that a team will manage, produce, and use in the process of developing, maintaining, and supporting software and, as such, are relevant to assessing the progress and health of a software endeavor. "A *work product* is an artifact of value and relevance for a software engineering endeavor. A work product may be a document or a piece of software."³ Practices are a kind of package consisting of these elements.

The Essence kernel, which stands at the bottom of Figure 4, is made up of a number of elements. The figure specifically shows the Software System alpha. The Essence kernel does not have an explicit notion of architecture because in simple development, this is left for teams to define. For more sophisticated development, the architecture practice fills the gap by providing explicit guidance on creating an intentional architecture. The architecture practice introduces an Architecture alpha that is described by an architecture description work product. The Architecture alpha provides guidance on how to determine architecture goals and how to identify and validate architecture scenarios.

The two domain-specific practices—namely, AIA practice and IoT OSI practice—provide specializations on

Figure 4. Architecture practices in Ignite and IoT Methodology.



how an IoT application architecture is described. The way teams work on an IoT architecture is similar to the way they work on other kinds of architectures. Thus, they do not introduce a new Architecture alpha but reuse the Architecture alpha and description from the generic architecture practice. There are specific considerations peculiar to IoT applications, however. Hence, each of these domain-specific practices introduces a pattern for describing an IoT application. A pattern provides domain/technology-specific stereotypes to model the IoT application. In Unified Modeling Language (UML) speak, this corresponds to a UML profile.⁴ UML profiles are a common approach to describe domain-specific architectures, and IoT is one such domain. The AIA practice introduces an AIA pattern for the architecture description, whereas the IoT OSI practice introduces an IoT OSI pattern. At the very top is a technology-specific architecture practice for EPC/REST-based IoT applications. This contains a specific pattern for EPC/REST.⁵

The layering of practices helps practitioners understand what is truly different when working with IoT-based applications, as opposed to a more general application. Understanding this difference helps practitioners quickly pinpoint the specifics they need to be aware of and, hence,

learn a domain quickly. This practice separation is in contrast to monolithic methods where salient aspects of such methods often drown in the sea of generic information. It also helps practitioners differentiate methods—for example, Ignite and IoT Methodology—from the way they work with architecture and to understand if they are truly different. Practice separation also helps practitioners pick the best parts from different methods, provided they have been decomposed, as shown in Figure 4. This mix-and-match approach helps teams become innovative with methods, as well as the solutions they produce.

Thus, architecture is one area that needs special attention when building IoT applications. Security and privacy also need special consideration. The IoT opens the world to new ideas and use cases, and, as such, product idea generation and formulation also need special considerations. Each of these areas require generic practices and domain-specific practices.

Welcome to the Future

The IoT promises a new dawn for all sorts of industries, fundamentally changing the basics of everyday life. Let's make sure our software-engineering practices do not get left behind. Let's stop producing inflexible, monolithic methods that are not easy

to adopt. Instead, the focus should be on essentialized practices that provide an incremental and safe path for teams and organizations to evolve and grow their ways of working.

By using Essence as the foundation for a new practice library, we can liberate the practices and provide development teams with the guidance they need to innovate, improvise, and excel. We can avoid the traps of the past and enable software-engineering methods to evolve at Internet speeds while building on established, proven practices. ■

References

- Brown, T. Design thinking. *Harvard Business Review* 86, 6 (2008), 84.
- Collins, T. A methodology for building the Internet of Things; <http://www.iotmethodology.com/>
- Evans, P.C., Annunziata, M. Industrial Internet: Pushing the boundaries of minds and machines. GE, 2012; www.ge.com/docs/chapters/Industrial_Internet.pdf.
- Fontoura, M., Pree, W., Rumpe, B. *The UML Profile for Framework Architectures*. Addison-Wesley Longman Publishing, 2000.
- Guinard, D., Mueller, M., Pasquier-Rocha, J. Giving RFID a REST: Building a Web-enabled EPCIS. *Internet of Things*. IEEE, 2010, 1–8.
- Jacobson, I., Ng, P.-W., McMahon, P. E., Spence, I., Lidman, S. The Essence of software engineering: The SEMAT kernel. *Commun. ACM* 55, 12 (Dec. 2012); and *acmqueue* 10, 10; <http://queue.acm.org/detail.cfm?id=2389616>.
- Jacobson, I., Ng, P.-W., McMahon, P. E., Spence, I., Lidman, S. *The Essence of Software Engineering: Applying the SEMAT Kernel*. Addison-Wesley, 2013.
- Jacobson, I., Ng, P.-W., Spence, I. The Essential Unified Process. *Dr. Dobbs's Journal* (Aug. 2006); <http://www.drdoobbs.com/architecture-and-design/the-essential-unified-process/191601687>.
- Object Management Group. Essence—Kernel and language for software engineering methods, 2014; <http://www.omg.org/spec/Essence/>.
- Page, V., Stimson, R. Essentializing the DSDM Agile Project Framework. Agile Methods Conference, London, 2016. Ivar Jacobson International; https://www.ivarjacobson.com/sites/default/files/field_jji_file/article/essentializingdsdm_1.pdf.
- Perkins-Golomb, B., Folkjaer, P., Rauch, F., Spence, I. Ending method wars: The successful utilization of Essence at Munich Re. Ivar Jacobson International, 2015; https://www.ivarjacobson.com/sites/default/files/field_jji_file/article/essence_munichre_0.pdf.
- Ries, E. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Random House, 2011.
- Slama, D., Puhlmann, F., Morrish, J., Bhatnagar, R. *Enterprise IoT: Strategies and Best Practices for Connected Products and Services*. O'Reilly, 2015.

Ivar Jacobson, chair of Ivar Jacobson International, is a father of components and component architecture, use cases, the Unified Modeling Language, and the Rational Unified Process. He has contributed to modern business modeling and aspect-oriented software development.

Ian Spence is CTO at Ivar Jacobson International and the team leader for the development of the SEMAT kernel. An experienced coach, he has introduced hundreds of projects to iterative and agile practices.

Pan-Wei Ng coaches large-scale systems development involving many millions of lines of code and hundreds of people per release, helping them transition to a lean and agile way of working, not forgetting to improve their code and architecture and to test through use cases and aspects.

Copyright held by owners/authors.
Publication rights licensed to ACM. \$15.00.