IVAR JACOBSON INTERNATIONAL

Agile Essentials

A Starter Pack of Agile Practices Presented as a Deck of Cards

Creating winning teams.

Introduction

Agile has changed the face of software development fundamentally and for the better. Which is perhaps why Scott Ambler felt justified in making the bold claim that "The process wars are over, and agile has won"[i].

But, as an industry, we are clearly not yet in a position to declare victory over the significant challenges that we face. While the Agile Manifesto, for example, encapsulates some excellent values and principles, it takes many years of experience to understand how best to live by them.

If we are to meet the challenges of the ever-expanding potentials and demands of the software development industry, we need to be able to codify and communicate this knowledge in more secure and scalable ways. We need to be able to progress beyond a growth-restricted "craft" mentality, where knowledge can only be passed on directly from a Master Craftsman to an Apprentice, over many years of close working and one-to-one coaching and intuition.

This pressing need is what is behind the call to plot a strategic path for the software industry to progress "from craft to engineering"[ii]. But the challenge is how best to do this for the subtle and complex art that is software development.

The problem of dilution and contamination

One challenge we face is the danger of dilution and contamination of the essential principles and strategies of agility. The more popular and mainstream agile becomes, the more people feel impelled to brand or rebrand themselves, their processes and their products as "agile". But if existing processes and products are all claiming to be agile already, then agile as a concept loses all meaning and, with it, all its power to achieve genuine transformation. Which is why, for example, Dave Thomas, in his PragDave blog, worries that "The word 'agile' has been subverted to the point where it is effectively meaningless."[iii]

Why we need more than just shared principles

A second challenge, more specific to agile, is inherent in its emphasis on principles over prescriptive process. This is right and proper, and is key to the power and success of agile – that it seeks to empower the local experts, closest to the problem at hand, and allow them to respond to their immediate challenges and circumstances as they see fit. The problem, however, is that abstract principles in themselves are easy to assent to, but equally easy to interpret in radically different ways. For this very reason, abstract principles alone have proved themselves to be relatively ineffective in changing behaviors, and therefore in achieving beneficial outcomes.

For example, it is close to impossible to disagree with any of the four value statements[iv] or twelve principles[v] of the Agile Manifesto. A common danger signal here is the response "Yes, absolutely, it's really all just good common sense, isn't it?" Which effectively translates into "In as much as I have engaged with your abstractions at all, I have interpreted them in such a way that what they now mean to me is that no change whatsoever is needed in my behavior". But unfortunately "no change" clearly also means "no new benefits".

One person's transparency is another person's opacity

Agile rightly puts great store on the critical importance of transparency in all things at all times. To build trust and enable predictability, we need it to be transparently clear to all stakeholders what we are doing, how we are doing it, why we are doing it that way, and how well it is going – i.e. what progress are we making towards achieving the required outcomes.

But we can only achieve transparency if we have a shared language with which to communicate and share these critical messages and facts. And there is a direct tension here between the empowerment of local teams to adopt whatever practices suit their local needs, and the ability to communicate status and progress to stakeholders external to the different development teams. As things stand, governance stakeholders in particular are often left feeling baffled by the latest jargon being bandied around at the team level, and what on earth its implications are for how best to track progress, steer investment and make other key value judgments and governance decisions.

The challenge of building a learning organization

Also central to lean and agile principles and strategies is the concept of a learning organization. In knowledge work of all kinds, including product development and software engineering, we are completely reliant on the creativity and capabilities of our people. We therefore need to be able to sustain and grow these capabilities over time, even as suppliers, contractors and even full-time employees join and leave our teams and our organization. If every new joiner brings their own unique preferred way of working, but deep expertise in applying it is reliant on their presence, and that knowledge exits the organization when they exit, then we will clearly struggle as an organization to continuously grow our capabilities and sustainably improve our performance.

What we mean by "the Essentials"

Here and now we have a significant opportunity to meet many of these challenges head-on. There is an increasingly strong consensus emerging on what constitutes good agile practice, at least in terms of valid practice options. If we can codify this shared understanding, then it will give us some concrete and communicable knowledge to work with, to share, and to build on.

This is what we mean by the Agile Essentials. Specifically it means codifying:

1. The critical success factors – the kinds of things that, if teams fail to do, or fail to do well, then they will consistently struggle to be successful
2. The common, useful, modular, agile practices, which deliver real value if deployed appropriately (what we might characterize as the common, basic "tools" that are available for selection from an agile practice "tool-box")
3. The things that we should not need to change, but rather we should expect to build on and reuse as a secure foundation, as we seek to scale the application of agile across our value streams to meet ever larger and more complex challenges.

We need modular practices

Key to achieving this consensus is codifying the most popular and successful agile practices in a sufficiently granular and modular form. Scrum and Kanban, for example, are relatively large practices, that each introduce numerous concepts and principles to address many aspects of a development endeavor. Their territorial ambitions inevitably cause them to come into conflict with each other occasionally, with the result that some see there as being a "Scrum v. Kanban" face-off in the air.

More experienced campaigners realize that this should not be the case, and are clear that we should actually treat these different practices as "process tools" to be selected from a "process tool-box". The problem is, however, that each of these processes is actually more like a "tool-kit", containing the many smaller tools potentially needed to get the whole job done - much as a bicycle repair kit might contain tire-levers, glue, patches and a spanner. That makes for a useful "starter kit", but many of these things can be used individually for other purposes. And if we also buy a "cycle maintenance" starter kit, will we end up with some duplicate tools with essentially similar purposes?

If we make the granularity of our practice thinking slightly finer, however, we will find the usability and applicability of these processes and practices increases, and overlaps and conflicts can be easily resolved. Even the most evangelical Kanban advocates, for example, will readily recognize the usefulness of a daily-stand-up meeting to align the work of a team. And even the most ardent Scrum supporters are open to the use of a Kanban-style board to track the progress of product backlog items through their interim progression states towards being completely done.

If we codify these conglomerate practices as more granular, atomic practices, we can offer teams some simple "adopt or not adopt" options – with practices like Daily-Stand-Ups, Retrospectives, Product Backlogs and the like. Other practice options will be "either, or" decisions – such as either using a Kanban-style "continuous flow" practice option or a Scrum-style "time-boxing" practice option for managing the flow of work through the team (but without feeling obliged to change or abandon the use of Product Backlogs or Daily-Stand Ups, for example, in either case).

This more granular approach to practice codification also opens up options for the much wider reuse and application of practices beyond software engineering, or even product development. All kinds of teams, for example, have found benefits in practices like daily-stand-ups and retrospectives, such as business change teams, even though other concepts like "working software" and "shippable product increments" may have no relevance for them.

Introducing the Agile Essentials card deck

The Agile Essentials card deck codifies a set of popular, standard agile practices in the form of these granular "tools", as opposed to the larger "tool-set" level practices in which such practices are usually bundled. Together, the Agile Essentials practices provide a basic "starter-kit" toolbox that covers all the common and critical aspects of team-based development.

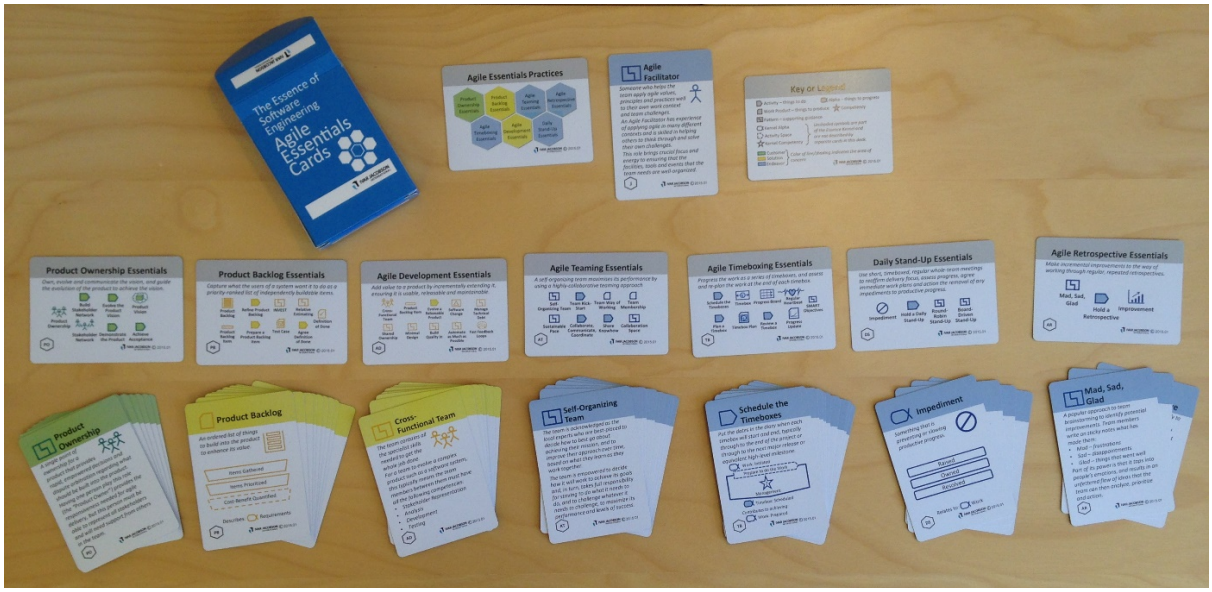

Figure 1 – Agile Essentials deck of Cards containing 7 modular, composable agile practices.

The practices in the Agile Essentials deck of cards are:

- Product Ownership Essentials - own, evolve and communicate the product vision and guide the evolution of the product to achieve the vision
- Product Backlog Essentials - capture what the users of a software system want it to do as a priority-ranked list of independently buildable items
- Agile Teaming Essentials - a self-organizing team maximizes its performance by using a highly collaborative teaming approach
- Daily Stand-up Essentials - use a short, daily, whole-team meeting to reaffirm delivery focus, assess progress, agree work plans and action the removal of impediments to progress
- Agile Development Essentials - add value to a product by incrementally extending it while ensuring it remains usable, releasable and maintainable

- Agile Retrospective Essentials - make incremental improvements to the way of working through regularly repeated retrospectives
- Agile Timeboxing Essentials - progress the work as a series of focused timeboxes, and assess and re-plan the work at the end of each timebox.

Each practice contains a small number of cards that provide useful, structured advice on how to successfully adopt and apply the practice.

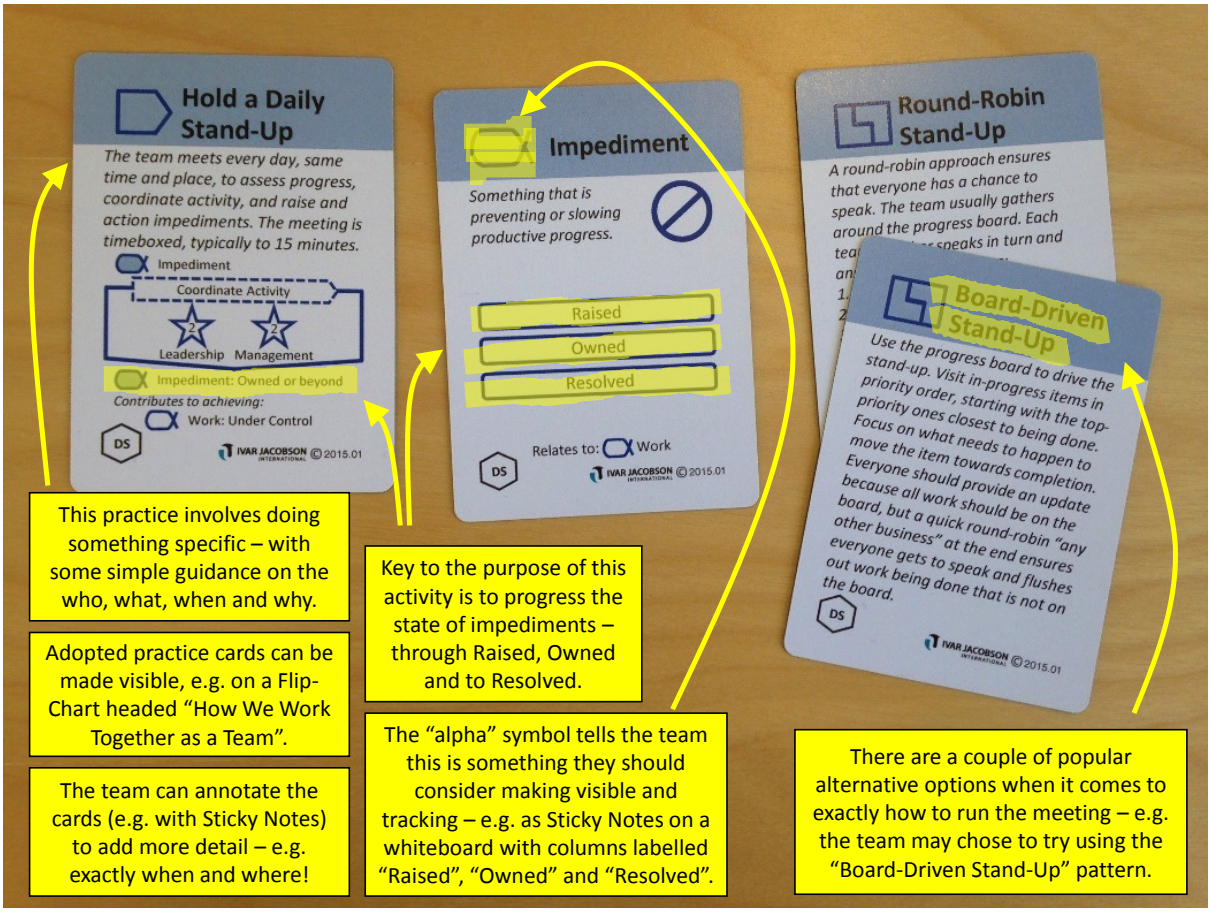

Figure 2 – Agile Essentials cards from the Daily-Stand Up practice

The science of mixing and matching

Experienced teams have become relatively adept in the art of mixing and matching practices, but again it is something that currently takes a lot of experience and confidence to do well. Here again, a small amount of codification can help to make this art more of a science-informed art and less of a mystical art.

In a box of tools, for example, we can naturally distinguish between tools that do a similar job, and should therefore be considered as competing alternatives (e.g. a spanner and a mole-wrench) and others that are naturally complementary (e.g. a hammer and a chisel).

If our practice "tools" came pre-labeled in a standard way that describe what kind of job they are useful for, we are considerably helped in identifying naturally complementary practices, and naturally competing alternatives, and so are able to mix-and-match with much more assurance.

Using the Essence Kernel to define interchangeable practices

The Agile Essentials practices have been codified using the standard Essence Language and Kernel[vi]. This means they support the process of selecting, mixing and matching practices, and also checking whether the assembled set of practices contains all the tools that we need for the job at hand.

A key thing that the Essence Kernel provides is a map of the scope and structure of the territory that we operate within and across as an industry. It provides maps of what it is we work with and what it is that we do, as follows:

- 3 Areas of Concern: Customer, Solution and Endeavour - color-coded green, yellow and blue
- 7 key concerns that we need to manage and progress (known as Alphas): Stakeholders, Opportunity, Requirements, Software System, Team, Way of Working and Work
- 15 Activity Spaces, which describe the general kinds of thing that we need to do well if we are to progress and complete a software development endeavor.

These maps enable us to compare and contrast practices, simply by characterizing which spaces they play in, and which spaces they do not touch on at all.

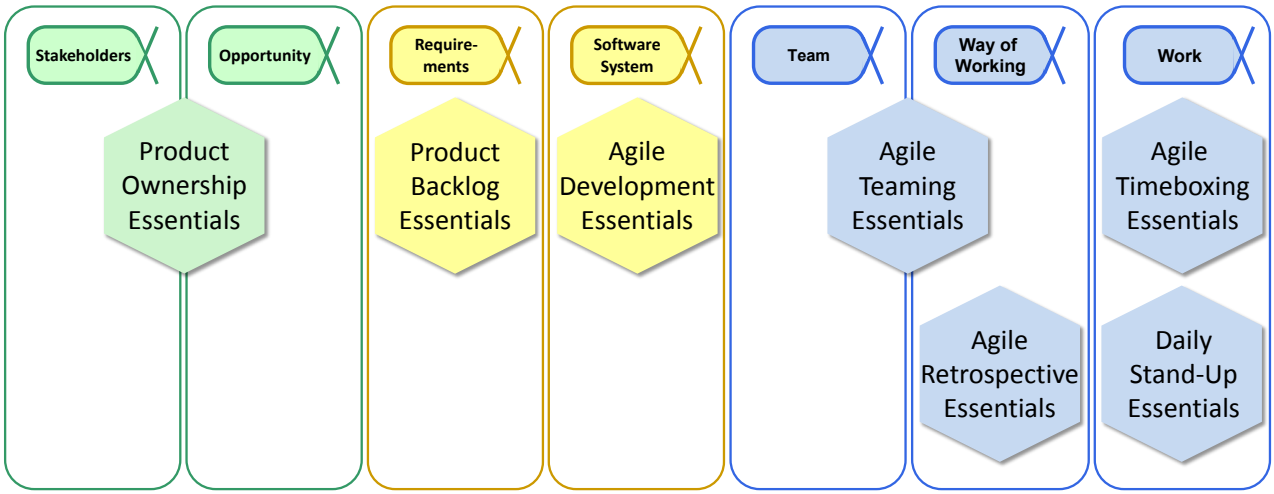

Figure 3 – Agile Essentials practices, showing coverage across Essence Kernel Alphas

Figure 3, for example, shows us that Product Ownership, Timeboxing and Retrospectives all operate in different spaces – and are therefore not competing alternatives, and may well be complementary.

Figure 4 shows a drill-down of the coverage offered by the Agile Essentials deck of cards in terms of the Activities it describes and how they map to Essence Activity Spaces. While this indicates a reasonable level of coverage, there are clearly some spaces that are not covered at all, including:

- Shape the System (which is all about agreeing the technical solution strategy, including the overall architecture of the solution)
- Deploy the System and Operate the System – which are the spaces where, for example, a DevOps or similar practice might be deployed.

If we are to scale up to more complex endeavors, therefore, we may well need to consider adopting additional practices to mix-in with the Agile Essentials practices to cover off spaces such as these.

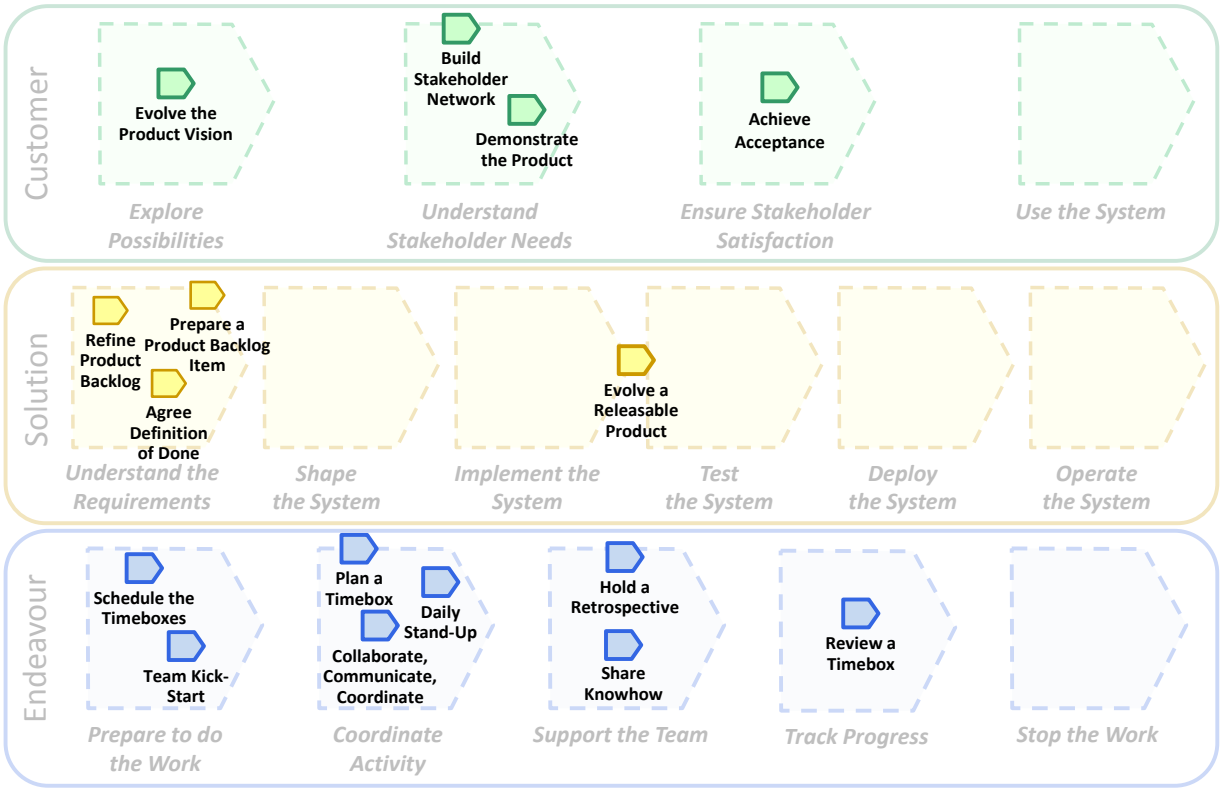

Figure 4 – Agile Essentials activities, showing coverage across Essence Kernel Activity Spaces

Summary

Well-defined, granular practices enable significant freedom in mixing and matching practices to build effective ways of working. Using the Essence Language and Kernel to define practices provides standardized, codified support for the process of practice selecting, mixing and matching.

For software practitioners, professionals and teams this means practical help in rapidly agreeing to a good way of working.

For leadership it enables empowerment of teams to select their practices, but without loss of visibility and control of what is being done, how and why. It also enables the organization to sustainably grow corporate learning through the evolution of libraries of modular practice.

[i] Forward to Disciplined Agile Delivery by Scott Ambler and Mark Lines [IBM Press 2012]

[ii] For example Real Software Engineering by Ivar Jacobson and Ed Seidewitz, [CSI Communications August 2014]

[iii] http://pragdave.me/blog/2014/03/04/time-to-kill-agile/, dated 4 March 2014.

[iv] http://www.agilemanifesto.org/

[v] http://www.agilemanifesto.org/principles.html

[vi] http://www.omg.org/spec/Essence/1.0/.

About Ivar Jacobson International

IJI is a global services company providing high quality consulting, coaching and training solutions for customers implementing enterprise-scale agile software development.

IJI improves the performance of software development teams by introducing new practices, and removing barriers to their wider adoption.

Through the provision of high calibre people, innovative practices, and proven solutions, we ensure that our customers achieve strong business/IT alignment, high performing teams, and projects that deliver.

www.ivarjacobson.com

Sweden

+46 8 515 10 174 info-se@ivarjacobson.com

United Kingdom

+44 (0)207 953 9784 info-uk@ivarjacobson.com

Asia

+8610 82486030 info-asia@ivarjacobson.com

Americas

+1 703 338 5421 info-usa@ivarjacobson.com