Industrial-Scale Agile

Challenges and Solution Strategies

Roly Stimson

Abstract

Industrial-scale agile means that agile at any-and-every scale is business-as-usual for an organization, across its entire portfolio, and that this capability is continuously sustained and strengthened.

This paper examines two leading frameworks that provide guidance on how to achieve success within this kind of "complexity at scale" challenge space - David Snowden's Cynefin framework and Max Boisot's I-Space framework. From these it distils three key common critical success factors:

- Having a shared language and conceptual framework
- Evolving a set of "good practices" available for selection and use by experts
- Forging a flexible knowledge dissemination strategy that enables the right approach to be adopted to achieving different communication goals – anything from word-of-mouth within a team, through to more formal codification and broader dissemination of shared practices.

SEMAT and the OMG Essence kernel and language clearly have a key role to play in all three.

Introduction

This paper examines the challenge of reproducing the successes of agile ways of working at ever greater scales – what we characterize as "industrial-scale agile".

There are two separate but related challenges associated with industrial-scale development. The first relates directly to the size and complexity, or complicatedness, of the undertaking – including the sheer size of the code base, the number of people involved in developing it, the numbers of stakeholders and stakeholder types impacted, and the number of unknowns (both known and unknown). The second is the challenge of sustainability and growth – can we replicate the success of one agile team across all our teams, can we embed agility as our business-as-usual way of working, can we sustain it through changes of personnel, and how do we continuously learn and improve as an organization over time?

In this paper we examine three established frameworks and show how these point us toward solution strategies that can enable organizations to succeed in sustainably accruing the benefits of industrial-scale agility:

- David Snowden's Cynefin Framework – which guides us on the different leadership and management strategies that are appropriate for different kinds of problem space
- Max Boisot's Information Space (I-Space) – which helps us devise communication and learning strategies across large endeavors, organizations and communities

- OMG's Essence Language and Kernel - which provides the shared language, conceptual framework and codification mechanisms needed to successfully apply the leadership, communication and learning strategies outlined by the Cynefin and I-Space frameworks.

Why Scale Agile?

Agile has delivered clear benefits and is a key driver for transformations aiming to achieve step-change improvements in development organization performance.

The motivation to scale agile is based on a straightforward desire to increase the scope and level at which these demonstrable benefits are accrued.

Much of agile's success, however, is based on a "small is beautiful" mentality, placing emphasis on "simple solutions"[1], "face-to-face communication"[2] and daily engagement with business users[3], for example. One way in which this philosophy has manifested itself is in the "Development as Craft" movement, which is closely aligned with agile development schools-of-thought[4]. Advocates of Software as Craft are insistent that treating software like an engineering discipline, and putting emphasis on codified and teachable disciplines and practices, breaks the golden rule of empowering teams as the local experts, best placed to solve the problem, and risks killing the golden goose of team innovation and motivation.

It is a simply brute fact, however, that some huge-scale development challenges exist. Software is something that can be and therefore is inevitably applied to every scale of human challenge – everything from relatively simple children's toys, to the enormous scale of nuclear power stations or fighter jets and the staggering complexity of global trading systems, weather systems and leading-edge "Big Science" projects?[5]

Of course, we can look to adopt strategies to avoid or evade the challenges of scaling-up as much and as long as possible, for example by:

- Starting small (e.g. in team size and levels of process ceremony), and scaling up only as and when we have to, and even then aiming to scale as slowly and smoothly as possible
- Use fractal models – e.g. keep each and every team small, and have many small teams.

But still we may inevitably need to face the challenges of scaling up in one form or other, including:

- How do we engage effectively with large and diverse stakeholder communities?
- How do we align multiple teams to advance one large and complex product?
- How do we grow and sustain effective agile delivery teams over extended timescales?

The sheer scale of demand for software-enabled development projects of all shapes and sizes means that there is a strong imperative to enable the effective communication of know-how and learning in a much more scalable way than can be achieved, for example, through a Master Craftsman conveying knowledge on a one-to-one basis to an Apprentice, over many years or even decades.

The question still remains, however, as to whether agile strategies are appropriate to these large-scale challenges. Shouldn't we just "flip a switch" and go back to "traditional" management strategies as and when we hit a certain scale and complexity of development challenge?

Our understanding of agile has increased substantially over the past 12 years or so since "agile" was first codified with the values and principles of behavior as set out in the Agile Manifesto[6]. While these guiding principles of behavior still hold good, it has been acknowledged that these are the description of a solution not a problem – they characterize how to behave, but don't tell us why.

Key to establishing the underlying benefits for agile principles of behavior has been the work of Don Reinertsen, whose work emphasizes the economics and the mathematics underlying agile principles of behavior[7].

If we characterize agile, then, not as a set of values and principles, but as a set of strategies for deriving certain benefits, we might summaries the results something like as follows:

BENEFIT		AGILE STRATEGY
Maximize responsiveness and return-on-investment	*by*	reducing time-to-value
Maximize quality and minimize risk	*by*	fast feedback, including through early and frequent testing, demonstration and release
Maximize productivity and predictability of progress	*by*	deploying a dedicated team, with all the skills required to "get the whole job done".

We can see that:

- The benefits derived from the use of agile are potentially that much larger for large-scale endeavors (through what we might call at-scale "benefits magnifiers" – see table below)
- The strategies likewise are equally or more appropriate for large, complex endeavors.

However, what is also clear is that the constraints and challenges associated with applying the strategy in practice also increase with scale, for example as follows:

AGILE STRATEGY	AGILE-AT-SCALE: BENEFITS MAGNIFIERS	AGILE-AT-SCALE: CONSTRAINTS AND CHALLENGES
Reduce time to value	Level of investment is higher Timescales are longer	Identifying opportunities for early value Cost and risk of releases
Use fast feedback	Business risks are higher Technical risks are higher Quality is more critical	Large and diverse user and other stakeholder populations Time and cost of test executions
One focused team	Predictability and productivity are even more important	Depth and diversity of skills needed Sheer number of people to align.

We are left with the conclusion that:

- There is urgent need and valid motivation for scaling agile to an industrial scale
- There are significant constraints and challenges associated with doing so successfully.

What Do We Mean By Small-Scale Agile?

Small-scale agile is relatively easily characterized with reference to the popular and publicly available Scrum framework[8], which has enjoyed much success in this space. Part of the secret of Scrum's success is that it makes certain simplifying assumptions about its application domain, as follows:

- One small, independent team (the "Scrum Team")
- One independent product (small enough to be advanced rapidly-enough by one small team)
- Guided by one genuinely empowered and representative customer (the Product Owner).

Scrum's strategy for keeping it simple might be characterized as having a Plan A that says – "if it isn't as simple as this, then strive to make it so". What is sometimes forgotten, however, is that it can't always be made exactly so. The less "so" we can make it, the more we will need to diverge from "Scrum, pure and simple", possibly by adapting and extending Scrum[9], possibly by mixing in other strategies, possibly to the extent that over time it turns into something very different from Scrum.[10]

What Do We Mean By "Agile at Scale"?

"Scale", of course, is not a simple binary attribute ("Small" or "Large"). Neither is it discretely quantifiable in one single dimension (e.g. "number of dedicated Full-Time Equivalents, or "budget in pounds sterling"). Rather, it varies continuously, and across multiple dimensions:

- Sheer size of product (lines of code)
- Architectural / technical newness / risk
- Non-functional requirements (performance, robustness, security, …)
- Cost of ownership / maintenance
- Number of people actively involved in advancing the system-of-interest or end-product
- Related change (business change, user training, cultural acceptance, …)
- Size and diversity of customer / user base
- Number of other types of stakeholder (security, safety, legal, financial governance, …)

However, if we are looking to characterize a simple "tipping point" when we can reasonably say we are getting into the space of "Agile at Scale", then we can use the same reference point as we used to characterize "Small-Scale Agile" and simply negate any or all of its presuppositions, as follows:

"SMALL-SCALE AGILE"	"AGILE AT SCALE"
One small, independent team	Large numbers of people are required to be engaged in highly interdependent work
One, small, independent product	A very large, complex product or interdependent family of products
Guided by one genuinely empowered and representative customer (the Product Owner)	A large and diverse customer and stakeholder community that can't reasonably be fully represented by just one person.

What Do We Mean By Industrial-Scale Agile?

Industrial-Scale Agile is different again. It essentially means the ability to sustainably apply agile strategies appropriately to anything and everything that can benefit from them. So this includes:

- Being able to do / be "agile at scale" as and when appropriate
- Doing small-scale agile as and when possible / appropriate
- Being able to grow, sustain and improve these capabilities over time.

The challenges associated with such universal and sustainable agility therefore include:

- Not just using exceptional, hand-picked, "tiger" teams, with special priorities and privileges and subject to "special exceptions" to normal operating procedures
- Not just applying agile strategies to certain, small parts of the value stream ("from requirements to code"), and not to others ("from concept to coding starts" or "from code complete to fully operational and supported live release")
- Being able to propagate agile capabilities to all people, functions and teams within the organization
- Being able to sustain learning and improvements indefinitely over time, despite staff turnover and other organizational change.

We can therefore characterize industrial-scale agile as

- "Agile at any scale"
- "Agile as the rule, not the exception".
- "Agile sustainably, forever", not just as a "one-off".

Industrial-Scale Development: The Challenge of Complicatedness and Complexity

As we scale up, the challenges we face do not just increase in quantity, they also change in nature and quality. That is, we face brand new challenges, not just more of the same. This is the phenomenon of emergent properties, which is a firmly established characteristic of complex systems.

David Snowden's Cynefin Framework provides an invaluable reasoning tool for analyzing the kinds of challenges we face, and adopting appropriate solution strategies accordingly.

We do not have space to fully explore or critically analyze David Snowden's Cynefin Framework in this paper, only to examine some key lessons and conclusions that it draws that are directly relevant to our examination of solution strategies for rising to the challenges of industrial-scale agile. Please see, for example "A Leader's Framework for Decision Making" by David Snowden and Mary (Harvard Business Review Journal November 2007) for an introductory overview[11].

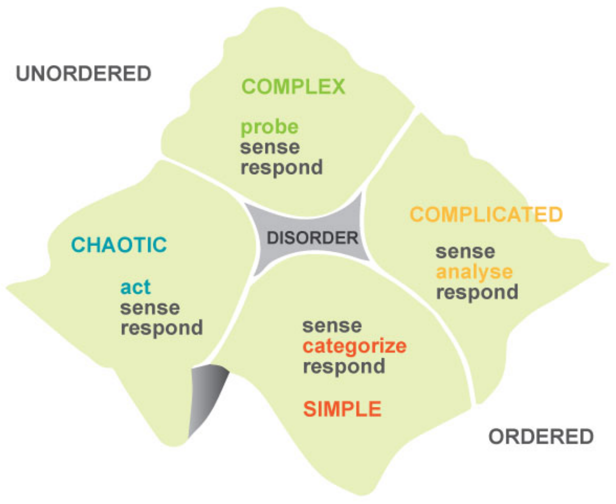


Figure 1: The Cynefin Framework[12]

The Cynefin Framework distinguishes between four different types of complexity "domain" that we might be required to operate within when undertaking any endeavor (such as a software development project, for example), and outlines the very different leadership and management strategies that each one requires:

Simple – where the relationship between cause and effect is self-evident, and an appropriate management strategy is to follow well-documented "best practice"

Complicated – where expertise is needed to select and apply the right approach by selecting judiciously from many available "good practices"

Complex – where even experts can't predict the outcomes of unfolding situations, and so the appropriate management strategy is to "probe, sense, respond", and we need to establish strategies that enable us to experiment safely

Chaotic – where everything is uncertain, and is a space we typically only want to be in when the imperative is to innovate.

A key thrust of David Snowden's work is to highlight the fact that we actually spend most of our time in a fifth conceptual space, that he calls "disorder", which is where we don't fully appreciate which of the problem domains we are actually in. When in this state we tend to interpret our situation and apply strategies that display personal bias, as opposed to the ones that the situation demands.

If, for example, we insist on a management strategy of "just follow the process", when in fact we are operating in a "complicated" environment, where we are reliant on expertise to be successful, we not only fail to leverage the available expertise appropriately, we actually risk alienating the experts, to such an extent that they effectively disengage from the approach and its chances of success.

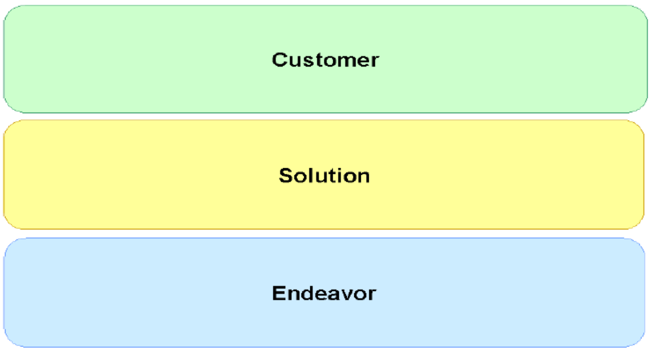
Likewise, if we assume we are in a complex or complicated space, when actually what we face is just a routine, predictable activity, we may waste significant time and effort and increase risk by encouraging experts to debate endlessly over the approach, when the application of a proven practice would have got the job done in an efficient and predictable way to everyone's satisfaction.

David Snowden's conclusions in applying the Cynefin Framework to software development are clear. Even within one specific software development endeavor or project, there are numerous different facets or aspects to the challenges that we face. As David Snowden describes it "Software development is a rich domain, with aspects and activities in all the different [Cynefin framework] domains".[13] What David Snowden calls for therefore is an "a multi-ontological approach" - "taking the best techniques for the various domains, and combining them in an appropriate and flexible manner".[14.15]

This is where the OMG Essence Kernel can help. One thing that this does is to codify the different "dimensions" of the "multi-dimensional" challenge space of software development. Once again we do not have enough space to fully explore this framework, but the reader is referred to articles such as A

New Software Engineering[16], other introductory papers[17] [18], and to the OMG Essence specification itself[19] and related SEMAT publications and initiatives[20].

Of particular relevance to this discussion is the fact that the Essence conceptual framework recognizes the fact that software development operates within a multi-dimensional problem domain, which Essence codifies under three main areas of concern as follows:



- Customer - everything to do with the actual use and exploitation of the software system to be produced.
- Solution - everything to do the specification and development of the software system.
- Endeavor - everything to do with the team, and the way that they approach their work.

We can use these three different areas of concern to help to characterize and understand in broad-brush terms the "complexity landscape" that we are facing, which might, for example be:

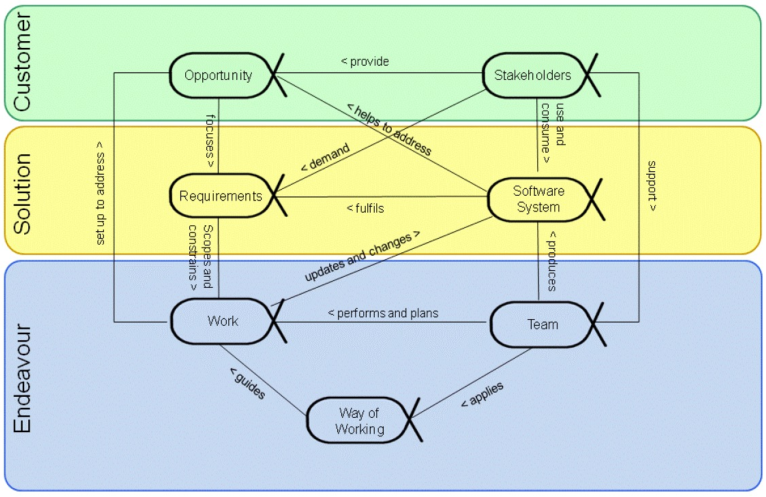
- Endeavor [Simple] – one co-located, multi-disciplined team
- Solution [Complicated] – complicated business algorithms needing expert guidance
- Customer [Complex] – many diverse users and stakeholders, all of them "change-averse".

The Essence Kernel additionally codifies seven key dimensions or "Alphas" within these three Areas of Concern, which:

- Capture the key concepts involved in software engineering
- Allow the progress and health of an endeavor to be tracked and assessed by tracking the status and progression of the Alphas
- Provide the common ground for the definition of methods and practices.

These seven key alphas are as shown and described below:

1. Opportunity - the reason for the creation of the new, or changed, software system
2. Stakeholders - the people, groups or organizations who affect or are affected by a software system
3. Requirements - what the software system must do to address the opportunity and satisfy the stakeholders
4. Software System - the primary product of any software engineering endeavor



5. Work - everything that the team does to meet the goals of producing a software system
6. Team - a group of people actively engaged in the development, maintenance, delivery, or support of a specific software system
7. Way-of-Working - the tailored set of practices and tools used by a team to guide and support their work.

The Kernel Alphas can help us to further "home in" on the different dimensions of the project and independently assess the kinds of challenges and the levels of complexity that we face within each, and on this basis guide us in the selection of appropriate strategies for navigating our way to a successful outcome.

Another key thing that the Essence Kernel and Language provides is a way of expressing modular, composable practices. These are exactly the kind of thing that David Snowden calls "good practices" – there is no "one right way", but good practices, in the hands of experts, become the selectable tools that help to get complicated jobs done in a controlled and effective way.

Thus, Essence-compliant practices and methods can be selected and adopted in different problem domains at different times depending on different complexity landscapes encountered. Extending the fictional scenario sketched out above, we might crudely characterize some options as follows:

- Endeavor [Simple] – one co-located, multi-disciplined team – the team simply use Scrum[21] as a pre-defined "best-practice framework" that we are used to using in this kind of space
- Solution [Complicated] – complicated business algorithms needing expert guidance – try using the Use Case 2.0 practice[22] to identify the algorithmic processing scenarios and devise a design, build and test strategy to implement the business algorithms incrementally
- Customer [Complex] – many diverse users and stakeholders, all change averse – try a combination of storyboarding, prototyping and early demonstration in support of a rapid, iterative, feedback-driven "probe, sense, respond" strategy.

What Essence enables and supports in this case is:

- The process of defining of a library of options of "good practices" available to experts as tools to use to attack complicated domains
- Guidance in selecting good practices, based on the fact that is clear which dimensions (areas of concern, alpha etc.) each practice operates within and across
- Help with fitting together different practices to provide an overall cohesive way of working that operates across the different dimensions of the problem domain
- Composing practices into larger-scale "best practice frameworks", or methods, for covering off relatively simple, well-understood problem domain spaces[23]
- Evolving an approach "in-flight", based on learning or changes that have the effect of taking projects into less well-charted waters (e.g. into "complicated" or "complex" domains).

Industrial-Scale Agile: The Challenge of Sustainability

The other challenge that we face is the problem of achieving sustainability though knowledge and information dissemination and learning. The prize here is to become the kind of organization that is consistently successful at tackling software development challenges of all scales, and continuously learning and improving over time, despite the "coming and going" of both personnel and of different types of challenge.

One of the key additional challenges of industrial-scale development relates to knowledge sharing and learning. Traditionally the models used for this were based on a relatively crude distinction between "tacit" and "explicit" knowledge, which seemed to leave us with a more-or-less binary choice as follows:

- Either keep practices implicit – i.e. leave each team to decide how to work (often characterized as "agile")
- Or provide teams with prescriptive processes to follow (often crudely characterized as "non-agile").[24]

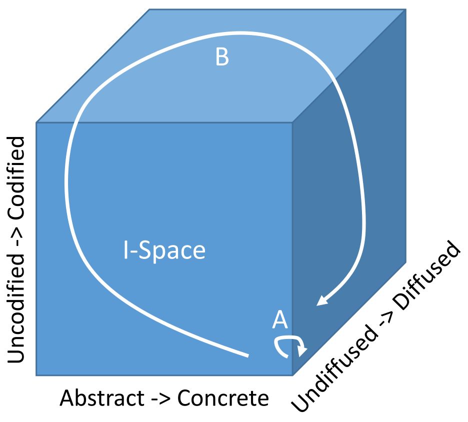
What starts to become clear is that we need a more subtle and sophisticated model for identifying acceptable / good approaches to capturing and communicating knowledge and learning within a multi-dimensional complexity landscape.

To analyze how we might go about meeting these communication and learning challenges at large scales, it is informative to look at how these challenges were approached on what is perhaps the largest and most complex single collaborative development venture in the history of mankind. I refer to the Large Hadron Collider (LHC) at Cern, and the experiments that it houses and runs and, in particular, one of the two main experiments jointly credited with the discovery of the Higgs Boson, namely; the Atlas experiment[25]. Like everything relating to the LHC, the scales are staggering. Designing, constructing, preparing and running the Atlas experiment involved a collaboration of some 4000 physicists, engineers and Ph.D. students from 174 research institutions located in 38 countries and using around 400 suppliers[26]. I think we can reasonably adopt this as an example of collaborative development "at scale"!

A model that was used to manage this enormous-scale communication challenge is Max Boisot's information Space or I-Space.[27] As Max Boisot describes it: "information and knowledge flows constitute the lifeblood of all organizational processes" and his Information-Space or I-Space framework was devised to "help us explore the nature of these knowledge and information flows".[28]

Once again, it is beyond the scope of this paper to fully explore, understand, interpret or verify this model. Here we can only offer some tentative proposals as to what insights it might give us into the challenges and solution strategies associated with industrial-scale development.

The I-Cube plots information in three dimensions as follows:



- Codification – the process of extracting useful data from surrounding noise ("what have we learnt in this case?"
- Abstraction – the process of establishing broader sets of cases to which the learning applies ("to what more general cases does this new learning apply?")
- Diffusion – the process of communicating knowledge to more and more people across broader knowledge communities.[29]

A learning-loop journey through I-Space involves both an outward journey of codification, abstraction and diffusion, but also a "homeward" journey of application by new teams and individuals to new concrete situations.

Journeys through I-Space can take different trajectories, and can be short or long, for example, in the above diagram:

- Journey A - might be a team discussing something that has happened and as a result start behaving and working better in future (application of team learning to new concrete circumstances, but no diffusion of learning beyond the team)
- Journey B - might be enshrining the learning from a new type of project as a fully documented full-lifecycle process for all future projects of this type to follow within the organization.

Longer journeys are clearly more costly in terms of time and effort to codify, teach and learn to apply successfully, but they have the potential to deliver much more value by reaching much broader and more dispersed communities much faster, including dispersal across time (to future team members or organization employees). As Boisot puts it; "the greater the degree of codification and abstraction achieved for a given message, the larger the population of agents that can be reached by diffusion per unit of time".

As we have already seen, modular Essence-based practices give us the tools that we need to codify and disseminate learning in a tailorable and flexible way, thus enabling us to plot many different journeys through I-Space to meet the exact knowledge propagation and learning needs and circumstances of each situation, for example:

- An organization invests in a standard default process framework for managing relatively simple and well-understood problem domains (e.g. a Scrum-like framework for one-small-team, one-independent-product type developments)

- An organization builds a library of practices over time, each representing good practice, for selection and application by the experts that operate in complicated problem domains
- A team innovates a new approach to a specific challenge; the result is captured as a small, modular practice and made available through the organization's practice library.

Thus we see that Essence breaks down the crude binary distinction between "word of mouth" (crudely characterized as "agile"), and documented process (crudely characterized as "prescriptive, therefore non-agile"), and offers a flexible range of options to suit the exact needs of the organization and the complexities of the situation, endeavor, or specific aspect of a larger endeavor.

By way of a footnote, another important conclusion that Max Boisot draws is that "the dynamics of diffusion is sensitive to how the target population is … partitioned into different groups" which itself is a function of the degree to which the population share the same language and conceptual framework. Or, as David Snowden summarizes it: "the less the shared context the higher the cost in money, time and effort of creating a knowledge artefact of artefacts which will successful allow knowledge to diffuse without the direct mediation of the knowledge holder."[30]

A primary goal of the SEMAT community and the OMG Essence Kernel and Language standard is to provide exactly this – a shared conceptual framework and language to enable more cost-effective sharing of learning within the software development community.

Conclusions

The need for organizations to be able to sustainably practice industrial-scale development is increasingly ubiquitous and urgent, and the motivations to replicate the benefits of agile development strategies at an industrial-scale are strong. The challenges, however, are considerable, particularly those that relate to knowledge and learning dissemination at scale.

In rising to the challenges we need to remember:

- One size does not fit all
- Communication is key
- Effective Learning is key to long-term success.

From the Cynefin framework we learn that the problem space we face is variable and multi-faceted, with different strategies called for in different places and at different times, even within the same organization, programme or project. Key to success is being able to understand the different dimensions and complexity domains and identify what kind of solution approach is appropriate in each case.

From I-Space we learn that there is an infinite variety of possible communication and learning strategies and journeys. So, again, it is clear that no one-size solution will fit all circumstances. Key to success is a flexible set of knowledge dissemination tools that allow the appropriate journeys to be plotted through the Information-Space in each case.

From both these framework authors we also hear a clear message that for learning to work well within and across we need a shared language and common conceptual framework.

OMG's Essence Kernel and Language provides us with:

- A shared language and conceptual framework: a prerequisite for effective communication and learning
- A framework to help select the right process strategy and practices for the right problem domains and complexity landscapes.

A practice-based approach that helps us adopt and execute the right tailored knowledge-dissemination strategies, including options such as:

- Wholesale adoption of predefined methods, practice frameworks or families
- Adopting different modular practices in response to different challenges
- Innovating new practices in response to novel and complex challenges
- Codifying practices as good practice options within an organizational learning library.

To summarize in the language of the "craft" and "engineering" paradigms we briefly explored earlier:

Essence provides a powerful set of process and practice tools for skilled craftsmen to use in meeting the complex challenges of industrial-scale development endeavors and organizations.

References

[1] "Simplicity--the art of maximizing the amount of work not done--is essential." [http://agilemanifesto.org/principles.html]
[2] "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation." [http://agilemanifesto.org/principles.html]
[3] "Business people and developers must work together daily throughout the project." [http://agilemanifesto.org/principles.html]
[4] See for example http://manifesto.softwarecraftsmanship.org/ and "Software Craftsmanship - The New Imperative" by Pete McBreen (2002).
[5] Later in the paper we will look at the example of the Large Hadron Collider (LHC) Atlas project – a leading-edge science and engineering collaboration of some 4000 physicists, engineers and Ph.D. students from 174 research institutions located in 38 countries and using around 400 suppliers
[6] www.agilemanifesto.org
[7] See The Principles of Product Development Flow by Donald G. Reinertsen(2009)
[8] See for example http://www.scrumguides.org/scrum-guide.html
[9] For example, using "Scrum of Scrums" (see http://guide.agilealliance.org/guide/scrumofscrums.html)
[10] Of course, we shouldn't need to stress that "other agile frameworks are available", as the popular media proviso goes, including other starting points (such as Kanban), and other scaling strategies, such SAFe (http://www.scaledagileframework.com/), DAD (http://www.disciplinedagiledelivery.com/), Less

(http://less.works/) etc. We use Scrum here not to indicate any innate preference for it, but purely as a simple and commonly known example of an approach designed and proven for use with small-scale endeavours.
[11] https://hbr.org/2007/11/a-leaders-framework-for-decision-making/ar/1
[12] Taken from "A Leader's Framework for Decision Making" by David Snowden and Mary -   Harvard Business Review Journal November 2007 [https://hbr.org/2007/11/a-leaders-framework-for-decision-making/ar/1]
[13] David Snowden [http://cognitive-edge.com/blog/is-software-development-complex/]. Different "activities and aspects" listed in this paper from brainstorming sessions with different projects include things like "changing requirements", "task estimation", "knowing when a task is done", "fixing the build" etc.
[14] ibid.
[15] An approach that he says "would also go a long way towards resolving the infighting and bickering now taking place between the "Agile" and "Lean" communities." (ibid.)
[16] Ivar Jacobson and Ed Seidewitz, "A New Software Engineering," Communications of the ACM, Volume 57, Issue 12, December 2014. Pages 49-54 (http://queue.acm.org/detail.cfm?id=2693160)
[17] Ivar Jacobson, Pan-Wei Ng, Paul E. McMahon, Ian Spence and Svante Lidman, "The Essence of Software Engineering: The SEMAT Kernel," Communications of the ACM, Volume 55, Issue 12, December 2012. And ACM Queue (Oct. 24, 2012); http://queue.acm.org/detail.cfm?id=2389616
[18] [3] Ivar Jacobson, Pan-Wei Ng, Ian Spence and Paul E. McMahon, "Major-League SEMAT: Why should an executive care?" Communications of the ACM, Volume 57 Issue 4, April 2014. Pages 44-50 [http://queue.acm.org/detail.cfm?id=2590809].
[19] "Essence - Kernel And Language For Software Engineering Methods" Version 1.0 Release Date: November 2014 [http://www.omg.org/spec/Essence/1.0/PDF/].
[20] http://semat.org/.
[21] http://www.scrumguides.org/scrum-guide.html
[22] http://www.ivarjacobson.com/Software_Use_Case_Essentials/
[23] See for example http://www.ivarjacobson.com/Agile_Essentials/
[24] Of course there are ironies and contradictions that arise from any crude and simplistic characterization of agile as believing "prescriptive process bad". Scrum, for example has some very clean, clear and simple rules, i.e. concrete, prescriptive process guidance – e.g. Sprints of one month or less, Daily Stand-Up meetings of 15 minutes or less, exactly one Product Owner etc. (see for example http://www.scrumguides.org/scrum-guide.html)
[25] See for example http://en.wikipedia.org/wiki/ATLAS_experiment
[26] See "Collisions and Collaborations": The Organization of Learning the Atlas Experiment at the LHC" edited by Max Boisot, Markus Nordberg, Said Yami and Bertrand Nicquevert,
[27] Ibid, especially Chapter 2 "A Conceptual Framework: The I-Space" by Max Boisot and Markus Nordberg.
[28] Ibid, Page 28.
[29] Description text and quotation marks the authors' own summary of "Collisions and Collaborations": The Organization of Learning the Atlas Experiment at the LHC" edited by Max Boisot, Markus Nordberg, Said Yami and Bertrand Nicquevert Chapter 2 "A Conceptual Framework: The I-Space" by Max Boisot and Markus Nordberg, but see also "Information Space: A framework for leaning in organizations, institutions and culture by Max Boisot (1995). The diagram is a simplified version of diagrams in these same sources.
[30] "The Origins of Cynefin - Part 1" by David Snowden [http://cognitive-edge.com/blog/part-one-origins-of-cynefin/]

About Ivar Jacobson International

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

We are passionate about improving the performance of software development teams, and maximizing the delivery of business value through technology.

Whether you are looking to transform a single project or program or your entire organization with lean and agile practices, we have solutions to suit your needs.

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
+1-703-434-3344
info-usa@ivarjacobson.com