“Architecture is the primary carrier of system qualities such as performance, modifiability, and security, none of which can be achieved without a unifying architectural vision” … SEI

More about architecture in the agile space

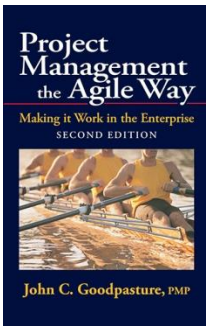
A companion white paper to the “Agile Project Management: Making it Work in the Enterprise, Second Edition

John Goodpasture, PMP

www.sqpegconsulting.com

Forward to the 2nd Edition

In the 2nd edition of *"Agile Project Management: Making it Work in the Enterprise"* there is a concept discussed that underpins all discussion of architecture in agile methodologies. In a few words, it is that agile projects are strategically stationary while tactically emergent.



We use the word "stationary" to mean that the strategic intent of the project, as chartered by the business, remains constant in time and location. No matter when or where you view the project, its strategic intent is clear and present. In this sense, strategic intent is somewhat of a wrapper or framework around the agile project scope.

Tactically emergent means that the project team, and particularly the agile teams, have license to define and develop details not apparent when the strategy was set, but nonetheless support the strategic intent, if not every detail of the strategy

DSDM is discussed in the book as one agile methodology that has architecture built into its methodology. But there are others that have architecture builit-in that the reader should be aware of.

Disciplined Agile Delivery, DAD

DAD is a richly documented "delivery" methodology that has strong appeal to system engineers and architects, but also has agile principles and practices at its core. Scott Ambler[1] is the thought leader behind DAD which he has developed iteratively over a number of years. DAD is not altogether stand alone. It is closely aligned with other Ambler ideas, which he calls collections of best practices representing both philosophy—that is, a point of view—and technique.

Chief among these supporting collections are the Enterprise Unified Process (EUP), Agile Modelling, and Agile Data. The latter two are collections of technical practices. EUP is a management idea to add "production" and "retirement" extensions to what Ambler calls "solution delivery frameworks". DAD is a solution delivery framework in this context.[2]

Architecture in DAD

There are a series of articles in the DAD blog that explain architecture in the context of DAD[3]. It begins with the idea of "enterprise architecture":

[1] Scott Ambler has a number of presences on the web; begin with ambysoft.com
[2] For more on EUP, see: http://enterpriseunifiedprocess.com/. For Agile Modelling: http://agilemodeling.com/; for Agile Data: http://agiledata.org/
[3] See: http://www.disciplinedagiledelivery.com/agile-enterprise-architecture/

- **Enterprise architecture (EA)**. An organization's enterprise architecture consists of the various structures and processes of an organization, including both technical structures and processes as well as business/domain structures and processes. There is always an enterprise architecture, even when it isn't documented.

Even though DAD allows for the EA not to be documented, a role is presumed called the Enterprise Architect who has responsibility for the enterprise architecture in whatever form it takes. Built into DAD is a purposeful collaboration between project architects—which could be a team of architects at both project office and agile teams—and the enterprise architect.

An architect team workflow is envisioned with four major tasks:

1. Envision initial architecture. The outcome is most often a model of the product or process as it will be integrated into the business. Although there could be iteration of this model, the initial model more or less sets the vision
2. Collaborate with business stakeholders. This task is an on-going level of effort which has communication at its core
3. Collaborate with IT stakeholders. Similar to step 3, but targeted more directly at those who have responsibility for the IT infrastructure which will support the product or process over the life cycle.
4. Evolve architecture assets. These assets could include artifacts architecture models, reference architectures, development guidelines, white papers, and structured analysis.

DAD architecture value-add

Architecture in DAD is perhaps best summarized by the value-add given as these five reasons for why have a planned project effort on architecture:

1. Common architecture enables agile teams to focus on value creation.
2. Common technical guidance enables greater consistency.
3. Agile architectures enable disaggregation
4. Common infrastructure enables continuous delivery
5. Enterprise architecture scales agile

SAFe

SAFe (Scaled Agile Framework®) is a framework tool, publically available, directed at

applying agile and leans methods at scale.[4] SAFe is governed by a set of nine principles, more or less universal in application, an idea promoted by W. Edwards Deming[5]

> *" .... The principles that will help to improve the quality of product and service are universal in nature."* W. Edwards Deming

SAFe principles

For purposes of architecture, the first three principles are guiding:

1. Take an economic view
2. Apply systems thinking
3. Assume variability; preserve options

As an architect, SAFe posits that the economic view will involve collaboration and iteration among the factors of system cost and total cost of ownership; cycle time, to include development and production; value in the sense of utility to business and customer; and risk in technical viability.

Systems thinking applies at two levels: the project level and at the enterprise business level.

- The former is where boundaries, optimizations, and allocations of function and feature are considered. These are set in context with product or service value-add that flows through interfaces, a point made strongly in the 2nd edition of "*Agile Project Management".*
- The latter is where the timeline changes scale, from the shorter term of the project to the longer term of the business; where the socialization of the project among business and customers is emphasized; and the value-add is measured and reported.

According to SAFe principles, variability is neither good nor bad. There can be both opportunity and threat as risk components. Each is to be evaluated. The main idea is to reduce variability by thoughtful elimination of possibilities, avoiding local optimizations that run counter to the larger picture.

Enterprise Architect

Like DAD, SAFe envisions a role for an enterprise architect (EA).[6] Although there are many details in the EA role, for purposes of our discussion the these are perhaps most important as "synchronization" tasks as the holistic strategic intent is kept in mind:

[4] The SAFe homepage: http://www.scaledagileframework.com/.
[5] See: http://www.scaledagileframework.com/safe-lean-agile-principles/
[6] See: http://www.scaledagileframework.com/enterprise-architect/

- Synchronizes all non-functional requirements across the enterprise
- Ensures system-to-system value flow
- Ensures standards of quality, security, (and presumably safety) are met

LeSS (Large Scale Scrum)

LeSS is more of a library of best practices that apply SCRUM at scale than it is a methodology or framework, as is the case with DAD and SAFe. Nonetheless, LeSS begins with the framework idea, and indeed two frameworks: LeSS and Huge LeSS, each distinguished by the number of teams supported.[7]

Like all of SCRUM, LeSS is light weight as compared to DAD and even to SAFe. LeSS has a set of principles, among them system thinking and queuing theory. Embedded in these are architecture tasks that provide the unifying vision of the qualities and value-add of the product. The LeSS model does not specifically have a role for system architect, nor a swim lane or sprint dedicated to architecture.

However, LeSS begins with "sprint planning 1" and "sprint planning 2" which are designed to set the larger scope in mind and allocate backlog in general terms. It is also the planning session where prior projects or debt from prior projects is coordinated or linked into the present effort. All such coordination and collaboration is under the hat of an architect role, even is such a role is not explicitly defined.

Public sector and NGO

One sure sign of agile maturity since the 1st edition is the embrace of agile in the public sector and the NGO[8] community. These entities have constituencies that are accustomed to fixed scope which justify taxpayer or donor expenditures. As the idea of an agile scope has made it into their projects the need for architecture to validate the vision, establish the strategic intent, and set the value proposition is all the more important.

To that end, in the United States in the public, the White House Office of Science and Technology Policy (OSTP) has been a top-level leader, along with the General Accountability Office (GAO)[9]. Policy has come from the former re modular, iterative, and agile acquisition in the government generally; project guidance has come from the latter.

These policy and guidance initiatives have provided a friendly context for much of the

[7] For the LeSS framework, see: http://less.works/less/framework/index.html
[8] Public sector: government entities; NGO: Non-government organization, typically not-for-profit
[9] OSTP: See: https://www.whitehouse.gov/administration/eop/ostp; GAO is an independent agency of the U.S. Congress, see: gao.gov

grass roots bottom up initiatives in the executive departments and agencies, as well as their top down policy and guidance.

More about this, with many references, can be found in the 2nd edition companion white paper about agile in the Department of Defense (DoD)

The NGO community, primarily industry associations, university think tanks, and quasi-public corporations, like Mitre have joined in with training, seminars, an white papers on the topic of architecture for agile in the public sector. Of course, in the NGO community, the IEEE and the EIA jointly develop and support standards, IEEE/EIA 12207[10] being the top level standard for software in the enterprise. 12207 does not specify or endorse methodologies, but there are many studies and papers that map 12207 process areas to the various agile methods.

Summary

The three agile approaches described in this forward and the four others in the 2nd edition, are seven among the many practice sets and methodologies that claim the title "agile". However, there are few that really bear down on the role of architect and the need for architecture viewed from the windshield as opposed to looking back at the rear view mirror for emergent architecture.

The public sector and the NGO community have provided robust and continuing leadership regarding agile methods generally and the need for architecture and systems engineering more specifically.

In this whitepaper, we make the case and explain more fully the need to look forward before opportunity has passed and the only thing available is the look backward.

John Goodpasture

Orlando

August 2015

More about architecture in the agile space

Architecture is the glue that binds all the features, functions, and infrastructure into a working and workable product

[10] See: https://en.wikipedia.org/wiki/IEEE_12207

"If a project has not achieved a system architecture, including its rationale, the project should not proceed …"

Dr. Barry Boehm

Every system, every product, every program has architecture, whether written or not, whether thought about or not. So, it's not a matter of architecture or not, it's only a matter of how architecture is represented and whether putting effort into expressing architecture pays off. In this book, we buy into Boehm's thinking: architecture is to be thought about and given authority.

In simplest terms, architecture is a conceptual description of a system, different from a business story or product vision because architecture reveals structure and structure relationships. In this usage, a system could be a product, process, or an assembly of cooperating components. The architecture descriptors are picked and organized for reasoning about, and appreciation of, the properties of the system. Architecture is a rendering of a system by mapping of functionality onto hardware and software, and by specifying the interaction between these components and the users.[1] Interactions normally occur between interfaces. System architecture is primarily concerned about identifying objects within a context, object behavior at interfaces, and object interface properties and performance, both internally and externally.

The world is flat

Tom Friedman has famously captured the spirit and the reality of architecture concepts in his bestseller, *"The World is Flat, A Brief History of the 21st Century"*.[2] Friedman describes 10 phenomena that he declares to be 'world flatteners'.

But the more important point for purpose of our architecture discussion is the three convergences he discerns. The first is a coming together of the 10 flatteners in a way that creates a *common platform* to enable really anyone anywhere to freely collaborate.

Friedman's second convergence is *'horizontalization'*, by which he means creating access to the platform by all manner of flanking functionalities and operating units. Look only to the phenomenon of the 'smart phone' to see how platform and horizontal access have transformed the device into a mobile usefulness far beyond making a phone call. Where a transaction path used to go up, over, and then down, grab something, and come back, now it just goes straight to and from the target, at ever increasing speed and intrusiveness.

And, lastly, especially important to the agile community, the third convergence is about *new customers in extraordinary numbers*, enabled by many and different functionalities, cultural democracy, and ubiquitous access to services and products.

New customers are coming from every corner. They are signing onto the global platform in truly stunning numbers. Look

only to the growth of social networks in just their first year or two: a few hundred, perhaps a thousand or two users at launch mushroomed 10,000-fold, in some cases 100,000 times, in just the blink of an eye.

What project methods can deal with such a dynamic customer base? What's needed in an architectural framework to handle application evolution? Think of the needs and wants such a group can generate!

	The world is flat
A project management tip	• Universal platforms, flanking functionalities for disparate access, and unknowable numbers of users with demands requires change be embraced and even encouraged • Architecture coupling must be maximally loose to accommodate services and structure to meet evolutionary requirements

Architecture communicates

Architecture establishes boundaries, especially the boundaries between interconnected services. In fact, there is an entire body of knowledge around service-driven architecture, formally known as SOA.[3] But the architecture of electronic connectivity has added ambiguity where there was certainty. Where exactly is the business boundary when there are all manner of documents, service requests, and responder data flowing between nodes of different interconnected enterprises?

The architecture of modern systems has introduced enormous security concerns into business and personal domains. Now there must be careful attention to authentication and authorization, encryption and disguise, solicitation and misrepresentation, intrusion to what used to be sanctuary, and all manner of Trojan horses. Indeed, the thoroughness by which agile methods treats security is key to its use in many industries, especially public service, defense, and intelligence.[4]

On the other hand, these same architectural features have dramatically changed the pace of business decision-making and operational transactions. They have raised personal productivity and introduced social structures into business that never existed before; they have challenged traditional relationships among the leadership and managers. Informality exists where it never did before. And perhaps very profoundly, sharing opportunities have been created where none were anticipated or intended. Purpose and mission:

For all the reasons discussed, a rendering is necessary. In summary, the mission of architecture is:

—To show interconnectedness and relationships;

—To show system structure and services,

even if intangible, in order to mitigate Weaver's organized complexity;

—To show context or environment, especially if there are legacy systems or applications to be interfaced or supported; and

—To establish a framework—hopefully a persistent framework that carries over planning horizons—upon which to hang product detail.

	Architecture brings out the best
A project management tip	Architecture is the means to bring cohesiveness to disparate requirements. It provides form and shape and connectedness. Coherence amplifies individual effects by harmonizing alignment. Coherency wrought by architecture can provide the 'ah hah!'

.

Scope fits architecture

Scope must a fit within architecture; even the scope of the lowest level business case must fit architecture. Architecture tells us the topology of the system, product, or process.[5] Topology tells us about hierarchy, interconnectedness, and whether nodes are reached by point-to-point, hub-and-spoke, or some mesh circuitry. In some cases, architecture gives the protocols, that is: the rules, by which elements of the system tie together.

Architecture gives form to requirements. It tells us whether we build in layers, tiers, and subsystems. Architecture gives guidance on how loosely coupled components can be, and how cohesive they need to be for good maintenance and operability.

Views and view points

To see and represent all these ideas, more than one point of view is often required. The standard for how to represent architecture, not what architecture must contain, is given in IEEE 1471-2000.[6] Chapter 5 of 1471 explains 'views'.

All architecture has views. View is what is seen from a view point. The customer, user, sponsor, project manager, lead developer, supplier, and post-go live support all have a point of view and see the architecture differently.

It follows that each view is modeled uniquely to show relationships between the system's structure and services, and its environment as viewed by a constituent; however, all views must be cohesive—that is, demonstrate that all elements fit together, even if loosely coupled.[7]

Agile makes demands on architecture

In the agile space, architecture more often than not is represented by a model that is easily transformed into executable code. Indeed, one way to represent architecture is with a class model in code. However, models are a means to communicate, to document, and give persistence to concept.

Therefore, the choice of tools should be governed by purpose and audience. There are many modeling tools from whiteboard sketches to UML [Unified Modeling Language] diagrams that are useful for scoping out architecture because models in different forms address the points for why have a rendering of architecture.

What we've defined as architecture is intended to be a big-picture view. Architecture is formed at three levels, mindful of the principle that governs all agile activity: just enough; just in time:

1- System level: business story, product vision and prominent actors, actions, and messages that interact with top level system components. Stakeholders and users engage at this level to communicate needs and concerns not obvious from the business story.

2- Planning wave: the system level model given richness and detail that is actionable over just a few months. Feedback from completed iterations give guidance to evolutionary insights to be accounted for in future iterations.

3- Iteration or sprint: the planning wave model that given class detail actionable over just a few weeks. Class diagrams and UML diagrams of various forms, CRC cards, data models, and code models are not easily understood by users; therefore, iteration detail should build on use cases and user stories with user collaboration before committing to more technical representations.

Care should be taken that architects leave generous room for maneuver as the teams plan iterations and SCRUM sprints.

In the agile space, project management expects the picture to develop incrementally. Incrementalism puts demands on coupling; it must be loose to accommodate additions and modifications. Incrementalism puts demands on interface design: The services and functionalities of the system must be made accessible to many subscribers, many of whom will not be known or appreciated until operational experience is had with the early releases.

And incrementalism is enabled by conformance to standards. Standards are open specifications. Openness promotes economies by reuse of design and reliance on performance already proven.

Does all this mean that the architecture is at the whim of every iteration and release? Yes, in the sense that a change to the framework may be envisioned at a particular iteration. No, in the sense that architecture is not only a description but it is also a protocol—rules and principles that guide additions. Consider building a house. It might be decided to add a room at the last

minute, but the architect will demand conformance to design protocols nonetheless. The roof over the new room will not be shingle if the baseline is tile; gables will not appear if they are not elsewhere.

At the bottom line is Steve McConnell's answer to the question: Why have an architecture? "Because the quality of the architecture determines the conceptual integrity of the system. That in turn determines the ultimate quality of the system." McConnell really has said it all. Architecture is the gateway to quality.[8]

Summary and take-away points

Architecture is the glue that binds all the features, functions, and infrastructure into a working and workable product. All systems have architecture whether or not it is written down or even thought about.

There are four big reasons to render architecture:

1. To show interconnectedness and relationships;
2. To show system structure and services;
3. To show context or environment; and
4. To establish a framework upon which to hang product detail.

Every beneficiary and constituent of the system has a unique point of view. Each view should be modeled to ensure all concerns and needs are accounted for at the top level of abstraction.

It's expected that as the project unfolds, architecture will evolve. To do so economically, architecture must be loosely coupled.

In a word, architecture is concept and form, submitted to communicate structure and relationships.

End Notes

1 "Users" could be other systems, services, or individuals that address the system through interfaces. Typically, an interface itself is an active component, potentially authenticating the user, accepting commands or inquiries or data on behalf of the system, and then processing the subsequent system responses into information compatible with the user.

2 Friedman, T.L., *"The World is Flat, a brief history of the 21st century"*, Farrar, Straus and Giroux, New York, 2nd edition, 2006

3 SOA, service oriented architecture, proposes that business activities are packaged as services and are called upon by users to deliver services in standard ways. SOA spans organizational boundaries, both business-to-business, and between and among business units in an organization

4 Baldwin, K. *"Agile in the DoD Environment"* Presentation to Lockheed Martin, posted May 2007, http://www.acq.osd.mil/sse/briefs/Kristen-Baldwin_Dallas-LMI-Agile-SW-Dev-Conf_2007-05-15.pdf, retrieved August 2009

5 A lot of words will be used interchangeably to represent the project outcomes: product, system, and deliverable. The outcome could be tangible, like a consumer product, or intangible, like a service. The project outcome could be a new process applied internally in the enterprise. Except in the most trivial cases, most outcomes depend upon being part of a system, whether legacy or new-to-the-world. In this sense, consumer devices like telephones are systems.

6 See FAQ about IEEE 1471-2000 at http://www.iso-architecture.org/ieee-1471/ieee-1471-faq.html

The formal title for the standard is ANSI/IEEE Std 1471-2000 :: ISO/IEC 42010 "*Recommended Practice for Architectural Description of Software-Intensive Systems*."

7 For more discussion on cohesion and coupling, see Chapter 4 of Goodpasture, J. (2010) "Project Management the agile way—making it work in the enterprise", J. Ross Publishing, Ft. Lauderdale

8 McConnell, S. *"Code Complete"*, Microsoft Press, Redmond, WA. 1993. pg 35

About the author

John C. Goodpasture, PMP is a program manager, instructor, author, and project consultant specializing in technology projects

For many years, he has been one of the instructors for an online distance learning course in Agile project management. He was project director of an E-Business application development unit at Lanier Professional Services where his team delivered a number of successful projects using agile principles and practices.

He is the author and contributing of four other technical books in project management, numerous magazine and web journal articles in the field of project management, and has been an invited speaker at many professional project management events.

After graduating with a master's degree in engineering, John was a system engineer and program manager in the U.S. Department of Defense leading high technology programs. Subsequently, he managed numerous defense software programs while at Harris Corporation in Melbourne, FL., eventually finishing his corporate career as operations vice president for a document imaging and storage company.

He has coached many technology teams in new product development and functional process improvement, both in the United States and abroad, in industries as diverse as semi-conductor manufacturing and retail mortgages.

For more on the subject of project management and Agile methods, check out these websites: John blogs at johngoodpasture.com, and his work products are found in the library at www.sqpegconsulting.com.

Many of his presentations on agile methods are found at www.slideshare.net/jgoodpas.
John maintains a professional profile at www.linkedin.com/in/johngoodpasture