White Paper

The Agile Enterprise Architect

Working Effectively with Sprint Teams, Backlogs and Scrum Masters

WP0106 | October 2013

Guy B. Sereff

Guy Sereff is an author, speaker and technology practitioner. His Technology Industry experience includes Application Research and Development, Large-Scale Technology Management, and Global Enterprise Architecture.

As well as a pragmatic blend of Strategy and Tactical execution, Guy also has extensive Architectural Domain experience which covers Business Architecture, Information Architecture, Solution Architecture and Enterprise Architecture.

In today's growing digital economy, organizations are continually challenged to deliver robust solutions under shorter and shorter timeframes. Large organizations, particularly those in heavily regulated industries, often face an even higher burden of ensuring additional risk management and that mitigation controls are in place, which can add additional layers that slow progress down. In order to accelerate productivity, many development communities have turned to Agile Software Delivery methods as a means of increasing their delivery velocity, or to at least keep pace as additional compliance layers are added on.

Implementing Agile in larger, more complex organizations, however, can pose significant problems and execution impediments when trying to scale Agile techniques at the Enterprise level. Challenges erupt between cross-functional service delivery models, traditional project management techniques, conflicting risk management models and efforts to ensure an overarching adherence to Enterprise Architecture policies, guidelines and roadmaps. It can and has been done before, but implementing a successful Agile practice at the Enterprise level that is in harmony with the rest of the institution seldom occurs 'organically' and typically must be done with very deliberate intent.

Enter the Enterprise Architect, who must keep the organization on a critical strategic path and show steady progress against Reference Architecture roadmaps without being perceived as slowing progress down. Agile teams need to be able to move very quickly through their tasks, and terms like 'oversight' and 'governance' often bring a cloud of

resistance with them. Enterprise Architects must find the right balance within their corporate cultural environment to become effective Agile Enterprise Architects.

In this paper, we'll first briefly discuss basic Agile concepts to establish a baseline for our dialog. Next we'll look at common challenges facing organizations that attempt to implement Agile at the Enterprise-wide level to set our scoping context. Finally, we'll discuss the following five principles that potential (and existing) Agile Enterprise Architects should consider in order to increase their effectiveness and likelihood of success:

1. **Study Up on the Use/Misuse of Agile** - Understand the organization's Agile Methodology definition, implementation and level of maturity
2. **Tackle the Hard Stuff** - Address the challenges of Enterprise Agile head-on within the context and culture of the organization
3. **Plan and Prepare Ahead** - Be prepared for rapid deployment ahead of time with reusable EA artifacts and components
4. **Work on the Front Lines** - Spend time with/as a Solution Architect through an entire Sprint or Delivery Cycle to keep a realistic point of view
5. **Reduce Friction** - Provide a means for Low-Friction Reference Architecture Adoption

Agile Concepts
The Art and Science of Agile

Over the past ten to fifteen years, the use of some form of Agile and its subtle variations has been of keen interest across the industry. Delving into all of the aspects and pros and cons of Agile as a discipline is far beyond the scope of this document, as there are numerous resources available on the subject. Our purpose for discussing Agile here in the context of this white paper is to understand the root thinking behind Agile at its origin, establish a baseline set of core concepts, which then sets the stage for a more focused discussion regarding the challenge of Agile at the enterprise level, as well as its impact on an organization's Enterprise Architecture practice.

Much of what we know of today as Agile, in terms of a software delivery methodology, was loosely formalized into a set of value statements in 2001, when a group of seventeen leading software engineering methodology practitioners from different disciplines and viewpoints came together and actually agreed upon what they termed the Agile Manifesto. This manifesto is a declaration of four simple, yet powerful

The Agile Manifesto[i]

The Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and Interactions** over processes and tools
- **Working Software** over comprehensive documentation
- **Customer Collaboration** over contract negotiation
- **Responding to Change** over following a plan

That is, while there is value in the items on the right, we value the items on the **left** more.

unifying values. The purpose of the manifesto was and continues to be collectively designed to significantly improve the software delivery process and to ease the persistent cycle of tension between the community of software consumers and the community of software producers.

Be sure to read the declaration carefully. The authors were not necessarily promoting the abandonment of structured delivery methods or suggesting the elimination of traditional project management in favor of chaos or anarchy per se; instead they were placing an emphasis on valuable activities that had the potential of much better outcomes than traditional methods had produced previously.

Elaborating further on the Agile Manifesto, the Agile Alliance organization went on to supplement the initial value statements by releasing the *Twelve Principles of Agile Software*[ii]. As you read through these principles, you'd correctly find them to be quite intuitive and rather insightful. Ironically, you may also find them to be quite provocative and potentially contradictory to traditional views on risk management and somewhat counter-intuitive to predictive engineering methods, depending on the context and culture of the engineering discipline present within your organization today.

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done - is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The Agile Alliance's Twelve Principles of Agile Software

Agile Software Delivery implements a rapid iterative approach within a set timeframe with the objective of creating a functioning product at the end of each cycle. There are a few 'styles' or methods of implementing

Agile, but at their core they represent an adaptive iterative approach that produces incremental working deliverables within a fixed period of time.

Requirements are typically captured as User Stories (textual description of a desired noun (actor)/verb (action) capability) or Use Cases (textual description plus primary and alternative path delineation with human actor/system actor integration information), depending upon the level of information granularity desired. The inventory of the requirements is typically placed into a backlog, with each discrete unit of work assessed a level of difficulty, assigned a delivery priority and potentially given a high-level realization effort estimate.

The time allotted for 'final' product delivery is broken into fixed-length cycles, or sprints. Each sprint is typically a self-contained loop or mini-project, where analysis, design, development test and deployment activities occur with minimal delay. Sprints are often sized in days or weeks in order to keep an aggressive delivery cadence going – sprints lasting months or quarters tend to morph into pseudo-waterfall efforts, looking less and less like true agile development as the initiative progresses.

Sprint mobilization typically involves assessing the current requirements backlog and identifying what work will be produced within the current cycle. Traditional project managers are often tempted to map out all of the User Stories or Use Cases by sprint through the end of the program engagement. While this can be a helpful starting point for the first iteration, this plan usually gets very 'muddy' and rapidly out of date after several sprints have been completed, due to the adaptive nature of Agile delivery. Flexibility around requirements clarification and their interpretation means that effort sizing estimates can (and will) vacillate wildly from their initial assessment, and there will invariably be work tasks that spill over from one sprint back into the backlog, which may or may not be addressed sequentially (i.e. unfinished work is not guaranteed to automatically continue during the next sprint).

More and more system capabilities are delivered with each iterative sprint, theoretically reducing the backlog over time. The 'burn down rate', or pace at which items from the backlog are being completed over time, helps assess the overall velocity of the team and level of functional completion by the targeted end date. Once the amount of completed work is sufficient from both a quantitative (level of functionality) and qualitative (robustness of the working system), the product is declared ready for release. Rigor around release candidate readiness assessment is based on the planned scope of deployment (i.e. Proof of Concept, Friendly User Test, Pilot, General Availability, etc.), with the intent to get the product out the door as quickly as possible. Backlog items not completed or only partially completed at the time of product release are either jettisoned or queued for consideration in a future release.

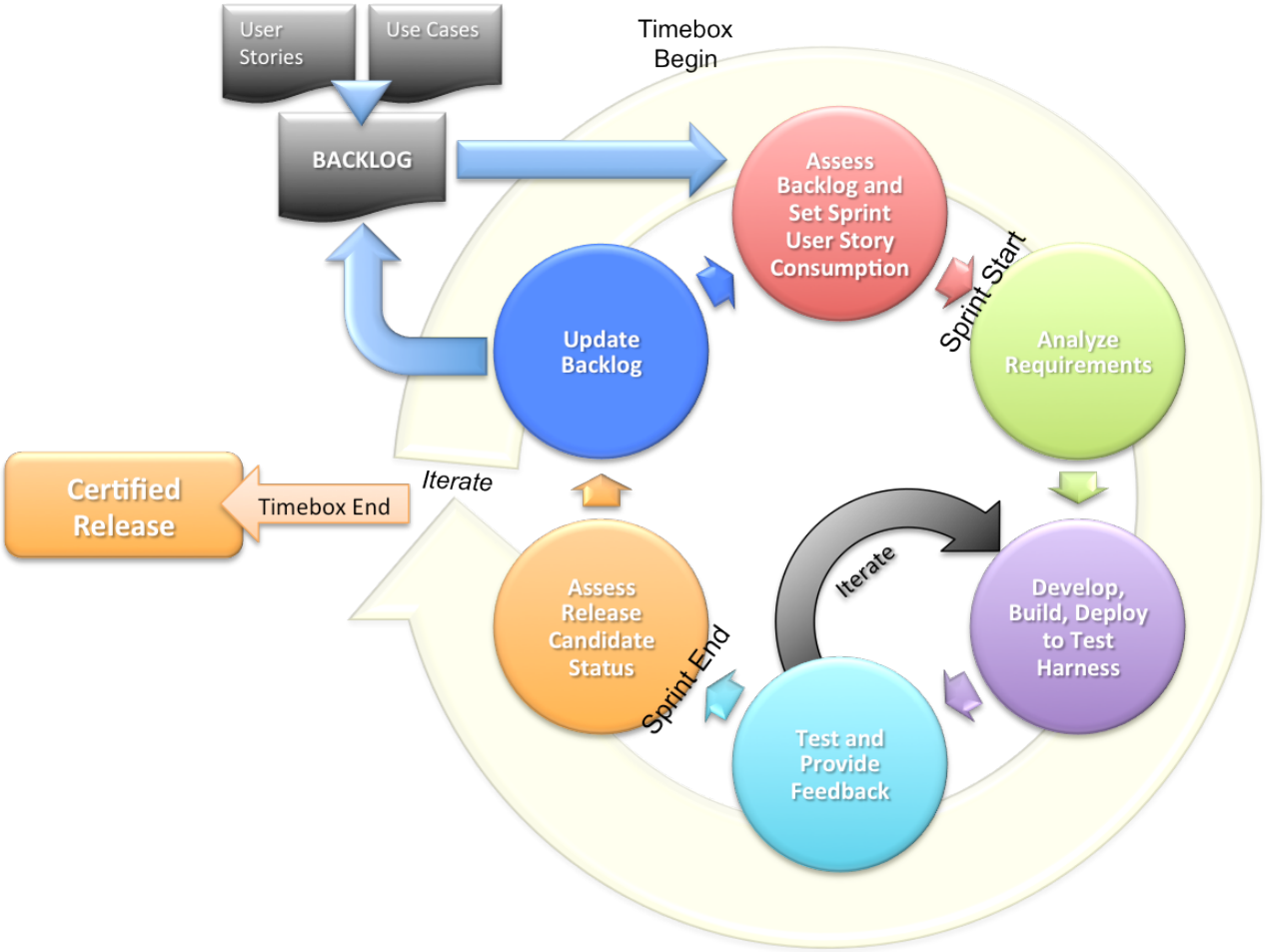

Figure 1 - Generalized Agile Delivery Life Cycle

Figure 1 depicts a generalized view of the Agile process as was previously described. The formal starting point for the Timebox depends on the organization's current level readiness to engage (i.e. does the backlog already exist or not) and the length of time mobilization will require. Cross-functional communication is critical to the success of Agile, as teams need to provide and receive fast and reliable feedback during and between task iterations.

Technicians and end-user representatives iterate back and forth between requirement definitions and test results. Designers and developers iterate back and forth between approach and implementation details. Numerous sprints are conducted within the overall project Timebox, producing some form of a 'working' system at the end of each sprint cycle. Simultaneous sprint teams can be leveraged, although they often require additional levels of cross-functional platform integration and validation work.

By and large, Agile can be very effective in the right operating environment when supported by appropriately skilled resources from all requisite disciplines (Project Management, Business Analysis, Architecture, Engineering, Certification, etc.). For those familiar with Agile already, these generalized concepts will likely resonate with current knowledge and practices at some level. There are several

industrial-strength Agile methodologies available and in practice today; our purpose here was only to point out a few common threads and concepts. For those who may be less familiar with Agile, you'll find additional background materials listed in the reference section that can potentially help you obtain a deeper level of understanding.

Challenges to the Agile Enterprise

From my perspective, the term Agile evokes an image of a light, nimble being or organism that is capable of quickly scurrying from one place to another, deftly accomplishing tasks and moving on to the next. The term Enterprise tends to evoke an image of a very large entity of great potential strength and scale, but not typically known for being particularly fast or adaptive to change. Linking the terms together, either as Enterprise Agile or Agile Enterprise, seems paradoxical at best. Large, global organizations are typically complex and struggle with rapid transformation. Large-scale risks grow into complicated policy safety nets designed to protect the stakeholders and comply with external regulators. Yet some firms are finding their way to harness the power and promise of Agile approaches and reaping the benefits.

For our purposes, we'll define an Agile Enterprise as an organization that has been able to successfully implement Agile methods in scale, specifically in terms of software development and solution delivery. In this context, 'in scale' means that a significant portion of software delivery follows a defined Agile methodology to deliver critical or strategic solutions. Repeatable patterns of success are evident and spread across more than one part of the organization.

Large organizations that are perhaps dabbling with Agile on a small pilot basis to deliver a non-critical application may be on the road to becoming an Agile Enterprise. However, many organizations are unable to successfully get Agile out of the lab environment and into the mainstream of their technology discipline. It's not that it can't be done; it's just that it isn't always intuitive and certainly not easy to do.

In The Enterprise and Scrum, Ken Schwaber discusses the concept of 'Muscle Memory Friction', in which the organization's progression to Agile, or Scrum in particular, is potentially hindered by four organizational memories:[iii]

- **Waterfall Thinking**
  Classic cascading project management techniques that are deeply embedding in the organization's psyche and contrary to iterative delivery approach methods

- **Command and Control**
  Managers believe they know best about everything and make dictation from 'on high' regardless of input from the community of individual contributors

- **Commitment to Defying the Laws of Nature**
  Engineers and solution providers yield to the pressure to make promises that they will deliver humanly impossible results...again

- **Hiding Reality**
  Continually communicating an untrue or overly optimistic status in hopes that someone figure something out before the truth gets out and the problems will magically go away

The potential threat of these points of hardened organizational thinking should not to be taken lightly. Organizations that lack a culture of transparency or don't tolerate the delivery of bad news well will be disappointed with their ability to adopt Agile in scale. Hiring a certified Scrum Master doesn't make the organization Agile; becoming an Agile Enterprise requires a commitment from top to bottom. Senior technology and business leadership must be willing to support real process change and invest in adoption. Developers and team members must be ready, willing and able to do 'Agile' and not have it forced on them. Too many 'voices of doom' or 'devil's advocates' around the table will lead to a self-fulfilling prophecy of failure, or at best lackluster results.

The following table, adapted from Ambler and Lines's book on Disciplined Agile Development, highlights common Agile scaling or gating factors that should be considered and addressed.[iv]

Geographic Distribution	Colocation is optimal; Where will the team be physically located?
Team Size	Optimal teams size is typically small (7-10 members); How big will the teams be and how many of them will there be?
Regulatory Compliance	Agile teams tend to resist Governance; What governing aspects are non-negotiable?
Domain Complexity	Agile complexity grows with the complexity of the domain; How complex are the problems to be solved with Agile?
Technical Complexity	Agile can be applied to new or legacy platforms of varying technical levels of difficulty; How complex are the technical environments that will be involved in an Agile effort?
Organizational Distribution	Cross-functional project teams can be challenging to manage even without Agile; What is the organizational topology view of the required Agile participants and stakeholders?
Organizational Complexity	Collaboration is critical to Agile; What is the current collaboration culture in the organization and how hard is it to change?
Enterprise Discipline	Agile takes time to do well and should not move the organization away from its strategic vision; How will fundamental enterprise principles regarding architecture, reuse and strategic alignment be incorporated?

Enterprise-Level Agile Scaling Factors to Consider

All of these gating factors can and should be addressed at some level ahead of time. The problem is that many organizations don't fully address them until the symptoms manifest themselves, often masking their root cause. Rather than taking a planful approach designed to establish a robust and scalable Agile practice from the onset, organizations plunge headlong into implementation. These same organizations are often later disappointed with their Agile results, incorrectly assuming that Agile simply is not a good approach after all.

One last observation we'll discuss on the challenges of creating an Agile Enterprise is the common disconnect from sound Enterprise Architectural principles. Case in point, the Ivar Jacobson International organization makes the following observation in the context of performing Enterprise-Scale Agile Software Development:

> ***Focus on Architecture***
> *Ensuring that the solution delivered is maintainable, extensible, and high-performing. Many agile approaches ignore architecture, or assume it can be derived by merely refactoring. This results in well-structured code but ignores the bigger picture. Focusing on the architecture is also essential to co-ordinating multiple teams working together.*[v]

Delving deep into the challenges of scalable Agile software delivery at the Enterprise level is a topic large enough to fill multiple white papers. Our purpose for touching upon it here is to highlight some of the common barriers to wide-spread adoption of Agile across larger organizations. This will help the Agile Enterprise Architect recognize potential issues that may be preventing their organization from enjoying the significant benefits Agile has to offer.

Becoming an Effective Agile Enterprise Architect

Now that we've briefly reviewed Agile software delivery methods and potential obstacles to achieving effective Enterprise-scale Agile proficiency, we turn our attention to the role of the Enterprise Architect. Regardless of the particular Enterprise Architecture framework an institution follows, architecture is generally broken into the following common domains: Business Architecture, Information Architecture, Solution Architecture, Application Architecture, and Platform (or Technical) Architecture.[vi]

The Business Architecture domain within Enterprise Architecture often aligns very nicely with the principles of Agile, focusing on the promise of rapid deployment of strategic business capabilities. However, when progressing through the other more technical Enterprise Architecture domains, challenges can erupt as attempts to align Agile teams to

architectural standards and reference architecture roadmaps is often perceived as 'heavy handed deceleration' and an 'innovation killer'. Yet allowing an Agile program to deliver solutions that are in conflict with, or perhaps even diametrically opposed to the organization's architectural standards is not desirable in the end run, no matter how fast the solution was delivered.

Enter the concept of the Agile Enterprise Architect – a proactive, pragmatist who focuses on maintaining the spirit and benefits of Agile while finding ways to apply sufficient architectural controls from a risk-based model to ensure that the broader interests of the enterprise are protected. We can further describe this concept in terms of a role definition such as the one below:

> *An Agile Enterprise Architect is an actively engaged Enterprise Architect who effectively guides and influences organizations through the Agile Software Delivery process, providing the appropriate level of design oversight and reference architecture governance without impeding the velocity of solution delivery or the level of delivered functionality.*
> ***Sereff, 2013***

Becoming an effective Agile Enterprise Architect requires planning and preparation in the anticipation of needs within the organization. Jumping into the middle of an active Agile engagement armed with only a legacy delivery solution mindset and limited tools will only foster disharmony and create tension. The following suggestions provide some practical steps to serve as a starting point to begin exerting more architectural influence into the Agile Software Delivery process.

1 - Study Up on the Use/Misuse of Agile

Understand the organization's existing Agile Methodology definition, implementation and maturity level

First and foremost, the Agile Enterprise Architect needs to have a full understanding of how the organization is executing what it considers to be Agile Software Delivery. Some organizations may follow all of the suggesting steps of a particular Agile methodology with academic precision. Other organizations may follow an adaptive hybrid Agile method that works best for the organization. Still other organizations may follow an obscure set of practices that are Agile in name only and look more like mini-waterfall iteration prototyping. While improving an organization's Agile implementation approach may be a future objective, the immediate requirement is to obtain and demonstrate a solid, functional knowledge of the current practice.

The reason for this level of knowledge is to allow the Agile Enterprise Architect to effectively engage in Agile sprint activities without disrupting the flow of progress. Collaborative teams do not always quickly welcome an 'outsider', particularly if they perceive that person has not done their homework or 'doesn't get it'. Building credibility is critical if one is to

influence behavior. The axiom of 'seeking first to understand, then to be understood' holds true in this instance.[vii] One way to gain hands on learning is to participate in an Agile cycle from start to finish, observing the process, roles and collaboration dynamics. Advice can be offered if solicited, but the real objective is to learn how the process flows and what the interaction model of the participants is today. This way one will be better prepared to be an active participant in the cycle.

2 - Tackle the Hard Stuff

Address the challenges of Enterprise Agile head-on within the context and culture of the organization

The effective Agile Enterprise Architect must be bold enough to tackle the more difficult issues head on. This doesn't imply an abrasive or dictatorial style, but rather a keen awareness of where the most significant architectural problems are and then focusing the team's energy on addressing those issues first. There is often a preference for taking the path of least resistance to keep up with the committed backlog burn down rate. However, architectural 'short-cuts' or plans for delayed resolutions are often very difficult to correct in any type of project. Since the goal of each Agile iteration is to produce a working production-ready candidate solution, the impact of regressive design can be even more difficult.

When identifying critical architectural issues, convey the downstream impact to the team to help them understand the level of difficulty in realignment in future iterations. Consider how the team can address the architectural issues while still being able to maintain the cadence required by the program drivers and stakeholders. Be prepared with workable alternatives that respect both the architectural requirements and the needs and pressures facing the delivery team.

3 - Plan and Prepare Ahead

Be prepared for rapid deployment ahead of time with reusable artifacts and components

Many Enterprise Architecture organizations produce a number of models reflecting various aspects of their organization. These models often reflect repeating business patterns or deployment solutions representative of the organization's current state. Yet many times, these images are not in a central repository or lack an adherence to a common set of standards or guidelines, making them difficult to quickly locate or reuse. Digging through old project artifact folders to locate an exported bitmap or JPEG file might help a little, but it lacks the attributes required for rapid reuse, extension or refactoring in the type of tight timeframes typically required by an Agile team.

Something as simple as having a series of UML 2.0 Sequence Diagram templates prepopulated with common lifelines in an engineering-grade modeling tool can speed the creation of contextual models, helping to accelerate design activities while conveying critical message sequences between platforms. Establishing an over-arching architectural approach at program initiation can quell in-flight debates during the delivery

process and focus efforts in the right direction as opposed to launching a realignment initiative at a later post-deployment date.

On the more complex end of the spectrum, having a collection of scalable Service Oriented Architecture (SOA) services available to expose core platforms from the organization's Reference Architecture technical specifications can quickly increase roadmap progression and accelerate solution delivery even further. The key is that this can't be done in parallel with the Agile activities themselves – that is not the time to introduce high-risk, critical path dependencies. Having proven, working modular solutions already available ahead of the Agile program demand curve not only provides an opportunity for rapid architectural alignment, it may even provide an overall solution delivery accelerant.

4 - Work on the Front Lines

Spend time with/as a Solution Architect through an entire Sprint or Delivery Cycle to keep a realistic point of view.

The successful Agile Enterprise Architect understands the value of hands-on engagement with the Agile delivery teams. I refer to this direct engagement approach as 'Architecture By Wandering Around', an adaptation of Tom Peters' recognition of the value of getting out from behind the desk and getting directly involved with the layers of the organization where production occurs.[viii] The key here is engaging in active collaboration and problem solving down in the proverbial trenches where critical decisions are made in real time, rather than simply watching things unfold from afar, resting comfortably whilst rendering architectural proclamations from the metaphorical EA Ivory Tower.

In this context, this is much more than monitoring daily scrum calls while multi-tasking and listening for occasional architectural questions. It is also more than being on 'stand by', waiting to engage only from a reactive 'as needed' basis. This is about full-on engagement as an Agile team participant – being recognized as a valuable contributing resource, working through simple and complex design issues, providing architectural insight and expert guidance, and above all being held equally accountable to both the Agile and Enterprise Architecture stakeholders.

Does this sound difficult? That's because it is. Does it seem like this has the potential to cause a lot of conflict? No doubt. But in the absence of direct engagement, the question arises as to how architectural standards are making their way into solutions at the cadence of Agile. This approach may be an uncomfortable stretch for some Enterprise Architects, and it may be the status quo for others, depending on the culture of the current environment and cross-functional working dynamics in place. However, this level of engagement allows the Agile Enterprise Architect to be a catalyst for change, building credibility and socializing the criticality and value of architectural alignment at the grass-roots level.

5. Reduce Friction

Provide a means for Low-Friction Reference Architecture Adoption.

The final recommendation for becoming an Agile Enterprise Architect is to aggressively work to reduce friction within the organization when it comes to its ability to adopt and deploy Reference Architecture components and design patterns. Many firms suffer from the situation where trying to follow the prevailing Reference Architecture definition takes dramatically longer than spinning up or perpetuating a non-compliant legacy environment. Discussions around the 'greater good of the Enterprise' are important, but having bottlenecks or chokepoints that discourage Reference Architecture adoption will generally not get the required buy in across the rest of the company, particularly within the Agile community.

If the organization felt strongly enough to establish a Reference Architecture position and corresponding roadmap, then hopefully it is also willing to invest in a process that supports rapid provisioning and deployment of said Reference Architecture in scale. Having gone through the Agile delivery process first hand as recommended above, the Agile Enterprise Architect is now equipped with an accurate understanding of where the process of adoption works well, and where it breaks down. Ready-to-Consume architectural components must also be Easy-To-Consume as well. If not, these strategic architectural components may be quickly passed over by an Agile team who must stay on track with the demands of their sprint cycle.

One way to assess where the 'hot spots' are when it comes to consuming strategic architectural components and aligning solutions to the prescribed architectural standards within the organization is to engage the Business Architecture and/or the Business Analyst community. Assign this group of process strategy experts a charter to build a detailed Business Process Model that accurately reflects all of the steps required when it comes to fully implementing a given set of Reference Architecture definitions. This model should include all steps, hand-offs and wait states from the component consumer perspective (i.e. an Agile engineer).

The main reason for doing this is based on the notion that software engineering is a business process in and of itself, regardless of whether or not an Agile methodology is being followed. As such, the same set of process evaluation and optimization tools and techniques used to help streamline business operations, such as Lean Six Sigma, can also be used here to optimize the integration of Reference Architecture components. Once complied and assessed, this information on low-friction process optimization can be used to demonstrate the dramatic leverage that can be achieved across all of the solution delivery channels within the organization.

Conclusion

Becoming an Agile Enterprise Architect is not easy work, and the principles shared within this document are by no means exhaustive. Some of the effort requires process changes that may or may not be hard to adopt, depending on the size of the organization and its ability to change. The core message is to take a realistic view of what is happening within the organization, engage in an impactful way, and be prepared to remove roadblocks.

For those that accept the challenge of becoming an effective Agile Enterprise Architect, potential benefits they'll be driving into their organization include:

- Better alignment of stakeholder needs across the organization;
- Stronger ability to provide architectural influence to Agile delivery teams in real-time rather than at post-activity reviews and checkpoints;
- Establishment of best-practices that benefit the Agile community as well as other parts of the Enterprise;
- Acceleration of Reference Architecture adoption;
- Strengthened Enterprise Architecture credibility and accountability.

Recommended Reading

The Agile Enterprise: Reinventing your Organization for Success in an On-Demand World
Pai and Pantaleo (2005)

Collaborative Enterprise Architecture: Enriching EA with Lean, Agile, and Enterprise 2.0 practices
Bente, Bombosch and Langade (2012)

Building the Agile Enterprise: With SOA, BPM and MBM
Cummins (2008)

Lean Architecture: for Agile Software Development
Coplien and Bjørnvig (2011)

Scaling Software Agility: Best Practices for Large Enterprises
Leffingwell (2007)

[i] Beck, Cockburn, Fowler et al. (2001). The Agile Manifesto.
http://www.agilealliance.org/the-alliance/the-agile-manifesto/.
Accessed September 14, 2013.

[ii] Agile Alliance. (2013). The Twelve Principles of Agile.
http://www.agilealliance.org/the-alliance/the-agile-manifesto/the-twelve-principles-of-agile-software/. Accessed September 14, 2013

[iii] Ken Schwaber (2007). The Enterprise and Scrum.
Redmond, WA, USA: Microsoft Press

[iv] Amblin, Scott and Mark Lines (2012) Disciplined Agile Delivery:
A Practitioner's Guide to Agile Software Delivery in the Enterprise. P. 22.
Boston, MA, USA: IBM Press

[v] Ivar Jacobson International (2013). Enterprise-Scale Agile Software Development.
http://www.ivarjacobson.com/enterprise_scale_agile_software_development/.
Accessed September 18, 2013

[vi] Sereff, Guy. (2013). Building an Enterprise Business Architecture Practice within the Broader Enterprise Architecture Context. P.3.
London, England, UK: Orbus Software

[vii] Covey, Stephen R. (1989). The Seven Habits of Highly Effective People.
P. 239. New York, NY, USA: Simon and Schuster.

[viii] Peters, Tom. (2005). tompeters! - MBWA After All These Years.
http://www.tompeters.com/dispatches/008106.php
Accessed September 21, 2013.

Orbus Software
3rd Floor
111 Buckingham Palace Road
London
SW1W 0SR
United Kingdom

+44 (0) 870 991 1851
enquiries@orbussoftware.com
www.orbussoftware.com

orbus software