An Agile Triple Play: Business Process Reengineering Meets SOA Meets Large-Scale Agile

DENIS DOELLING, Cognizant Technology Solutions
EDWARD GOLDFEHN, Cognizant Technology Solutions

This experience report addresses the challenges and solutions of implementing a Fortune 100 order-to-delivery manufacturing program, with the goal of reengineering and aligning global business processes with Web services and a service-oriented architecture using Agile practices. Our key findings were based on our work in establishing a framework that would support the design required to build an n-tier architecture while providing traceability from business process scenarios to the program and delivery backlogs, using Agile practices to guide a multi-year development and implementation effort.

1. INTRODUCTION

This experience report describes the challenges and solutions of implementing a Fortune 100 order-to-delivery manufacturing program, with the goal of reengineering and aligning global business processes with Web services and a service-oriented architecture (SOA). Large-scale Agile frameworks are fundamentally based on a multi-national and distributed set of delivery teams, working primarily at the user interface level, or the SOA Consumer Layer. Business process reengineering efforts, such as this program, consist of a multi-national and distributed set of teams defining every aspect of multiple SOA layers. This necessitated the authors to combine a set of frameworks and patterns with extensions to create a program-specific Agile delivery environment necessary to build and implement a multi-tiered architecture. This report is a retrospective analysis of these efforts and the results experienced in the program.

The report assumes the reader has a working knowledge of Agile methodologies and a basic understanding of SOA and business process reengineering. The report begins with background information on the project and Cognizant’s Daikibo™ Agile Framework. The background also provides an overview of SOA fundamentals and a comparison of the types of Agile development elements used in traditional Presentation Tier and N-Tier/SOA projects. This is followed by a summary of our assessment and analysis of the client’s Agile development process and the solution we implemented to address the issues identified. Finally, the report offers a retrospective of the results of the solution and a summary of the lessons learned.

2. BACKGROUND

The client’s business process reengineering (BPR) program started in 2011. The goal of the program was to implement a unified end-to-end order-to-delivery process, using a globally aligned business process model to replace four major non-integrated regional systems. This involves migrating from multiple sets of legacy and custom-built order-to-delivery applications across the regions into a single new set of global processes and systems. Following industry best practices, a SOA was selected as the program’s architectural framework.

The BPR program initially followed a traditional waterfall approach. The focus was on gathering requirements across the regional business groups (RBGs). The waterfall approach encountered four significant problems: 1) a global requirements-gathering effort was very expensive and time-consuming; 2) the RBGs had different and often limited documentation for the existing applications; 3) many of the individuals who originally developed or supported the legacy applications were no longer with the company; and 4) even with application information, defining an aligned global process was not a likely result of the effort.

In 2012, the BPR program brought in a consulting company to transition the program to follow Agile practices and principles. One improvement made at that time was the creation of a formal business process alignment team staffed by senior managers around the globe from the program’s impacted departments and process consultants. This team was charged with defining the new globally-aligned business process models that the BPR program would develop and implement.

Denis Doelling, Cognizant Technology Solutions, St. Louis, MO; email: denis.doelling@cognizant.com
Ed Goldgehn, Cognizant Technology Solutions, Santa Fe, NM; email: edward.goldgehn@cognizant.com
Copyright 2015 is held by the author(s).
By the fall of 2012, the BPR program management was not seeing the increased volume of software being built as expected from its transition to Agile. The program engaged Cognizant Technology Solutions to review the current state and recommend improvements. The major challenges that Cognizant identified included the lack of a well-defined backlog of work, the use of Excel workbooks to track progress, a lack of identification of dependencies between the teams within a functional business area, and a lack of ownership of the backlog by the business.

2.1 Cognizant’s Daikibo™ Agile Framework

Cognizant was engaged to restructure the program around its Daikibo™ Agile Framework. Daikibo was a good fit for the nature of this program and the enterprise culture because of its focus on the delivery of a flow of value in a multi-shore delivery environment in which there may be no on-site business representatives.

The two unique characteristics of Daikibo include its focus on large-scale efforts to deliver software, defined as multi-team Agile projects or programs, as well as its focus on distributed or multi-location teams and organizations. These teams may have members within a company or involve multiple organizations and companies; they may also be in different regions, continents, time zones and/or cultures.

In the Daikibo Agile delivery approach, the traditional Agile single-team structure is split into three separate teams to allow for both multi-team and multi-location scalability. The three teams include Concept Teams, Delivery Teams and a Validation Team. Typically, one Concept Team produces user stories from the features within a functional business area or value stream. One or more Delivery Teams consumes those stories and produces tested, potentially shippable code. The Validation Team integrates and validates the total solution from multiple Delivery Teams and business areas. With potentially many Concept and Delivery Teams, this structure is highly scalable and robust across multiple locations. An optional Meta Team is recommended when the program consists of three or more Concept Teams.

Daikibo ensures Agile intimacy through the Cross-Functional Team (CFT) concept, which is overlaid onto the Concept, Delivery and Validation basic team structure. A CFT is a virtual operational group composed of two or three members from the Concept Team who are coupled to a Delivery Team. Another way of saying this is that key individuals participate in both Concept Teams and Delivery Teams as glue. Similarly, there are engaged and empowered individuals who participate in both Delivery and Validation Teams, which avoids the loss of knowledge understanding and transfer between fast-moving related teams.

The Meta Team is composed of one or more of each of the following roles: Architects, Development Leads, Validation Lead, Product Owners, Support Team Leads, the Agile Program Manager and Agile Coach. The Meta Team’s role is to identify and monitor the critical path, prioritize the program epics and separate them into smaller epics and features by business area. This includes management of the program roadmap based on business value and the overall technical, functional and architectural aspects of the project. The Meta Team is also responsible for maintaining oversight of the Agile process employed by the program and planning releases.

Each Concept Team is composed of a Product Owner and more than one Story Author. The team may include Application Architects, Designers, Development or Tech Leads, Testing Leads, Training Analysts, Interface Leads and others in the roles of “directional alignment agents” of cross-functional teams. Concept Teams are intended to remain small. The Concept Team’s role is to decompose features prioritized by the Meta Team into a backlog of groomed user stories in a consumable state for the Delivery Teams. The Validation Team’s role is to consume features and groomed stories and build the test suites necessary to perform system integration, regression and performance testing on software produced by all the Delivery Teams across the program.

2.2 Fundamentals of Service-Oriented Architecture

For a variety of reasons, including leading industry practices, the architectural model selected for the program was a service-oriented architecture. Due to its history, a significant amount of confusion or misinformation often exists about what SOA is or isn’t and the role it plays in delivering a development program or project. This program was no exception to those challenges.
In architectural terms, virtually every Agile project or program is a SOA-based effort that falls into one of three categories: (a) consumer/presentation services, (b) legacy migration or (c) business process reengineering. According to the TOGAF SOA Reference Architecture Technical Standard (The Open Group 2011), SOA is an architectural model that provides a logical representation of capabilities. It is not the same as a technology model, which is how a SOA model is realized. The SOA Reference Architecture (RA) has only been available since 2011, although IBM published the original architecture model, on which it is based, in 2007. In essence, the SOA RA technical standard snuck into the enterprise information technology industry during the recession years. The timing of the publication in that economic environment has left a gap in the working knowledge and the necessary skillsets of both enterprise and small/medium business IT staff.

The vast majority of “traditional” Agile projects fall into the consumer/presentation services category. These types of projects focus only on the Consumer Layer in the SOA RA. This layer includes the effort required to develop user interfaces and integrate with back-end APIs. This is especially true for e-commerce and cloud-based applications (XAAS), or for developing an in-house multi-channel customer experience management (CXM) system. The REST services utilized by these models are simply subsets of SOA focused on the capabilities associated with enabling channel-independent access to the business processes and services supported by the various applications and programs (see http://www.opengroup.org/soa/source-book/soa_refarch/consumer.htm).

Legacy migration projects are generally “top-down” in nature because they extend the effort from the Consumer Layer and include new definitions of the back-end APIs in the Services Layer. In these types of projects, the business logic associated with the effort primarily remains in the Operational Systems Layer and is made available by “wrapping” these systems in services. Beginning around 2005, many of these efforts utilized the business process modeling (BPM) tools and technology stacks promoted by industry vendors. Using Agile in these projects meant leveraging the BPM tools to enable a set of SOAP-based back-end API calls for the Consumer Layer. Unfortunately, these technology-based projects often over-promised and under-delivered capabilities. The term SOA continues to be synonymous with BPM technologies across industries and enterprises, fueling confusion and misinformation about the architectural model.

Business process reengineering projects are predominantly “bottom-up” in nature. First, and foremost, they require identifying the business processes that will be reengineered and re-implemented in the Business Process Layer. This includes legacy migration of existing processes and the implementation of new processes. In many cases, this requires a program-level multi-year roadmap that must also incorporate legacy system decommissioning. It is the combination of new processes with legacy system decommissioning that underscores the need for reusability, combined with reductions in the total cost of ownership (TCO). It was SOA’s ability to deliver these benefits that led Forrester Research to call SOA “entrenched” (DeBarros 2010) and PriceWaterhouseCoopers to label SOA as a mainstream strategy for IT performance (PriceWaterhouseCoopers, LLP 2010) in 2010.

2.3 Differences between Presentation Tier and SOA/N-Tier Agile Development Elements

A SOA/N-Tier project requires much more than a user view of the business functionality to be delivered. Each layer in the SOA model utilizes and provides a collection of loosely coupled reusable services that must work together. Combined, they enable a holistic set of business processes. The underlying business functions are based on a set of business logic driven by one or more decision services. As a result, the interactions between Service Layers require a very different set of development elements from an Agile project focused only on the Presentation Tier.

For example, while a Presentation Tier project may utilize wire frames for high-level design and functional requirements elaboration, an N-Tier project would utilize business process flow diagrams. Similarly, whereas Presentation Tier story authoring would be user-centric, N-Tier story authoring should be Service/Consumer
Layer-centric. These types of differences are significant because they underscore a significant set of differences related to backlog development, grooming and prioritization from a “traditional” Agile effort. These differences occur regardless of the size of the effort (large- or small-scale Agile).

The following table provides an overview of the comparisons that Cognizant used with our client and the BPR program. The development elements in this table are not intended as an exhaustive list of differences between the two Agile project categories. However, the content provided in the table serves as a foundation to our approach and experience with the program and the outcomes provided in this report.

<table>
<thead>
<tr>
<th>Development Elements</th>
<th>Presentation Tier / Traditional Agile</th>
<th>SOA / N-Tier Agile</th>
</tr>
</thead>
</table>
| High-level design & functional requirements | • Business function flows  
• Wire frames  
• User interactions                                                                 | • Business process flows  
• Business function models  
• Service Layer interactions (includes API contracts) |
| Performance criteria and non-functional requirements | • UI response time SLAs  
• Business function SLAs                                                                 | • Individual service response times  
• Business function SLAs  
• Business process (end-to-end) SLAs |
| Story authoring              | • Specific user-centric (actor) focus  
• User functions  
• Back-end API interaction                                                                 | • General consumer-centric focus  
• Business function decomposition  
• Techno-functional decomposition  
• SOA Layer interactions |
| Testing strategy             | • User interface actions validation  
• User workflow  
• User management and security                                                                 | • Stand-alone service operations  
• Integrated service interactions  
• Business function operations  
• End-to-end business process  
• Policy management and access control |
| Build and deploy             | • Source configuration management  
• Environment deployment scripts                                                                 | • Source configuration management per tier and per service  
• Data configuration management  
• Decision service deployment management  
• Policy/access control management  
• Environment deployment scripts |
| DevOps                      | • HTTP access and error monitoring  
• Presentation Tier load monitoring  
• Back-end API monitoring                                                                 | • HTTP access and error monitoring  
• Decision rules monitoring  
• Service Layer interaction monitoring  
• Service load monitoring  
• Service component monitoring (includes Operational Systems Layer)  
• End-to-end QoS monitoring |

3. TRANSPARENCY UNCOVERS THE REAL IMPEDIMENTS

Denis Doelling joined the program early in 2013 as an Agile Coach working under a Senior Agile Coach. The initial goal of the coaching was focused on training teams on the Daikibo Agile Framework and assisting Meta and Concept Teams in creating and preparing the product backlog for the program’s first “release” planning event in May of that year. Because the focus was on transitioning the previous backlog of use cases into features and stories, we recommended quarterly planning events, knowing that we would address the creation of the product roadmap and the release schedule post transition. The remainder of 2013 was focused on helping the teams improve their delivery of software.

Edward Goldgehn initially joined the program in mid-2013 as part of the Implementation Planning Team and was charged with performing an architectural assessment of the program’s global rollout and implementation strategy. In late 2013, he was subsequently reengaged as a Consulting Architect, with the responsibility of driving the adoption of leading industry practices, maturity improvement and consistency with SOA patterns across the overall program.

In early 2014, the business program director requested assistance in defining and tracking the delivery of business value in alignment with the Application Lifecycle Management (ALM) tool. It was that request that led us down the path of an assessment and gap analysis, followed by improvement recommendations and implementation of the recommendations in the program.
3.1 Assessment of High-Level Design, Requirements and Story Authoring

During our initial assessment, we observed a lack of iterative and recursive discussions between the stakeholders and teams. Discussions were frequent, but they are used for firefighting the current hot spots. Historically, the program used use cases and other traditional waterfall documentation methods. There was no evidence of contextual diagrams, models or “good-enough” documentation practices. This left a void after the structure of the waterfall methodology. With the conversion to Agile, the only documentation processes that were created resembled a Presentation Tier development project, based entirely on user interfaces and business user actors.

In one work stream, a user interface for limited back-office functions was being used to drive design discussions. Unfortunately, the back-office functions represented one business actor’s view of a narrow set of business capabilities. In other areas, we observed teams getting lost in their discussions due to the lack of a user interface on which to focus the discussion.

Service definitions were defined on an ad-hoc basis using entity definition models associated with the Operational Systems Layer. This created a series of conflicts between each of the Concept Teams involved in a Service Layer interaction. We consistently found API definitions being developed without collaboration by both consumers and providers. This resulted in a consistent quantity of spikes, blockages, rework and inter-team conflicts.

The lack of contextual diagrams also impacted the Sprint demonstration. Without user interfaces, the functionality being demonstrated was restricted to an entity-centric stream of data. Without contextual diagrams, the business was unable to associate the user story and demonstration with a higher level business function or business capability.

Functional requirements were collected in the form of confirmation criteria in textual list formats in the user stories. This led the Design Team to create in-depth functional decompositions and detailed linear “story maps” based on a single tightly coupled business function. This decomposition became the basis of highly complex user stories that were extraordinarily large chunks of inter-dependent functionality. The design time requirements for these story maps were excessive and led to a “just-in-time” product backlog.

The program lacked any traceability from business processes to the thin product backlog, which resulted in a lack of functional and integration test cases. There was no evidence of focus on business processes, business functions or service-centric technical criteria. This lack of a common understanding of the context of the work between Product Owners, Concept and Delivery Teams created a series of tactical solutions that had limited association with future-state business processes.

Non-functional requirements were also restricted to a conceptual (non-documented) User Interface Layer only. We observed that the teams were not comfortable with actors that were generic service consumers of a loosely coupled function. This led to the discovery that the teams had not been identifying or documenting the discrete service timings or interactions outside of the linear and inter-dependent process flows defined in the story maps. Unit and integrated testing, therefore, lacked any performance criteria from which to determine whether delivered functionality was operating within acceptable parameters.

3.2 Technical Debt Accumulates

When initially defined, the program’s releases were based on a high-level five-year architectural roadmap. The plan was to deliver a production release every year focused on core business capabilities required for reengineering the business. The plan also included shifting the primary focus on SOA provider and consumer capabilities in alternating years.

The program’s inability to shift from waterfall use cases to an iterative requirements discovery and elaboration process led to significant gaps. One of those gaps was an inability to translate the architectural roadmap into functional designs with definitions. A series of tactical solutions resulted, based on a limited set of knowns within program visibility. The program intentionally set aside the unknowns to a future date. This was especially true when IT was confronted with a need for the business to define the capabilities and globally aligned processes necessary to meet the future state. At this point, all aspects of strategy and critical path prioritization also became purely line-of-sight. To accommodate this approach, the program elected to focus on tightly coupled deliverables in what is best described as an entity-centric approach using a modified Rational Unified Process (IBM, Inc. 2001).
Inevitably, the tactical solutions were handed off to the Concept Teams without sufficient high-level design elaboration. The process also drove the need for story authors to pull designers into a series of meetings to discuss the details of the intended functionality during backlog development iterations. This resulted in a quantity of technical debt in both design realization and solution delivery. Combined with the lack of traceability to the reengineered business processes, delivery commitments were either entirely missed or had no identifiable business value.

3.3 The Solution: An Agile Design Elaboration (ADE) Framework

In response to the assessment, we developed a lightweight elaboration framework in order to increase the iterative and recursive design and development discussions between the Product Owners, Concept and Delivery Teams. The goals of the framework were to provide the necessary business context around the development effort described in user stories. The intent was to enable a consistent generation of intuitive design documents using iterative and collaborative communications and enabling “good-enough” documentation to support all phases of the release to production lifecycle.

The ADE framework established both transparency and traceability from the future-state business processes, down to the features and user stories. This portion of the framework provided the Meta Team with visibility into a much more strategic view of the existing backlog in order to effectively establish development commitments and release planning goals. For the Concept Teams, each feature could now be mapped directly to a business function and its associated business process. This also enabled the identification of the functional test cases for discrete service interactions, as well as end-to-end business functions. Finally, the ADE framework helped to align the design elements in the ALM tool, including reporting of completed features.

With the ADE framework in place, we were able to address a requirement to map the reengineered and globally aligned business processes to a business capability model provided by Enterprise Architecture (EA) for C-level value stream reporting related to regional and global deployment and adoption. Because the BPR effort would be delivered in incremental phases, we extended the ADE framework to support a series of interim releases of business process models and change management processes in concert with the EA guidelines. This multi-release became the basis for the development of a comprehensive multi-year roadmap that is used to define budgetary resource allocations and architectural enhancements to support the business goals.

Finally, we place metrics to measure the flow or cycle time of creating the models and the cycle time of feature to user story decomposition. As of this writing, these metrics are being manually collected and tracked due to ALM tool limitations measuring only execution flow. A statistical view from these metrics is still pending and may serve as part of a future experience report or case study.

3.4 RESULTS

The ADE Framework has been instrumental in instilling a balance between functional teams attempting to implement functional (business) stories in vertical slices versus component teams delivering technical stories needed to implement the N-Tier architecture. The contextual models have increased collaboration and discussion. They also have allowed for good-enough high-level feature design that supports evolutionary design by the Delivery Teams. In other words, better definition of the scope of the feature has allowed the teams to fill in the details without incurring technical design debt.

The value of having models was very apparent in a major implementation planning discussion with the first market, in which the use of an interim capability model was instrumental in identifying gaps in knowledge between the program and the market.

The program also found new opportunities to implement small slices of functions and capabilities. These implementations not only resulted in a return on investment but were also used as opportunities to test the implementation/ adoption process on a bite-sized scale.

4. WHAT WE LEARNED

We learned that a disciplined approach is needed when business process reengineering is driving the software development. You need to be able to link the new business capabilities and business process flows within each capability with the product and program backlogs. The approach is not to solve the problem but to facilitate the discussions regarding the sequencing of the business capabilities to enable adoption earlier than later.
More thought and planning of releases is required when implementing changes to an enterprise’s business capabilities. The client’s concept of frequent releases and their timing remains challenged by the perceived complexities and lack of experience of implementing services and service layers across a global enterprise. Don’t be surprised, as in this case, if the minimal viable product requires a long development cycle before a sufficient set of capabilities can be implemented.

Feature and story writing at scale requires supporting contextual artifacts such as the elaboration framework we developed for this program. The contextual artifacts have helped the program in guiding the conversations necessary to decompose epics to features and features to stories. A program of this scale and complexity from both a business and technical perspective requires that knowledge be captured in some way in order to facilitate the flow of information and stop the introduction of business and technical design debt. Relying solely on people’s ability to remember the important architectural, technical and business elements over a five-year period is not practical. Models and other contextual information should be lightweight and good enough to support implementation discussions or refactoring due to the addition of the next increment of an interim business capability. The addition of information such as business rules taken from the business process reengineering effort also provided the necessary linkage between the business process and the service functions.

Business and IT must think in terms of business capabilities instead of thinking in terms of the user-centric interaction when implementing enterprise business capabilities in a multi-tiered SOA. Thinking of functionality from the user point of view will lead to design debt, which increases the cost of development and adds risk to the implementation.

A noteworthy experience was the opportunity to test the elaboration framework on a regional application development project. The application would be one of the future consumers of the new global services. The regional project’s Concept Team was trained on the elaboration framework; however, after a couple of weeks, the client’s project leader suspended work because the epic and feature packages were not being completed. This full stop of the project was beneficial in ensuring the elaboration framework was implemented well so that we could show its value to the larger global program.

Changes in our thoughts were that a repeatable and disciplined approach based on models fosters conversation and captures the really important decisions that are essential to guiding programs of this scale. Although something like this could be considered big upfront, allowing each area to figure out its approach to feature and story elaboration raised the risk of building the wrong things, in the wrong way and for the wrong reasons. In addition, consistency in the overall contents of each epic and feature package provided a consistent and valuable collection of information essential to supporting incremental, multi-release implementation plans.

4.1 Summary and Outlook

Have a plan for when and what contextual artifacts and models are required to support collaboration and implementation, especially when embarking on a transition to Agile. With increased complexity and risk comes the need for lightweight documentation to provide traceability, reinforce release planning, and support incremental implementation across the enterprise.

Release on-demand may not be possible when replacing legacy enterprise functionality with new enterprise capabilities implemented as services. Planning the contents of a release will be required since implementing only a slice of a capability or feature on-demand may not be possible due to the scale of the implementation. In this case, the process models were the tools that established the release goals.

Finally, SOA presents new challenges that are not easily managed or solved using user interface-centric Agile practices. It takes time to get comfortable with delivering business value without a user interface. Realize that SOA may put some constraints on the evolution of design at a system level, but the models created by the Architecture Team should provide clarity to the Delivery Teams so that they can evolve the detailed design.

The outlook for the BPR program has improved considerably compared with the state of the program during our original assessment. Challenges still exist in the SOA design processes that are related to skillset improvement instead of process adoption. However, avoidance of additional technical debt and conformance to the SOA model for new development are now program directives. A multi-year roadmap has been developed that is being used to drive a strategic backlog with critical path management by the Meta Team. Concept Teams are engaged in elaborating requirements using the ADE Framework, with traceability to business value and enterprise business capabilities. And while there are still several improvements to make to achieve predictable velocity with quality, the maturity of the program is trending in the right direction.
5. ACKNOWLEDGMENTS

Thanks to:

- Aita Saloosa for her insights and for encouraging us to submit this report.
- Jutta Eckstein, our Shepherd, who provided us with lots of questions and challenged us to refine our thoughts.
- Jeff Boyle and Cognizant Technology Solutions, for funding our participation and attendance.

REFERENCES


