The Agile Advantage: How Agile Integration Improves Outcomes For Software as a Service (SaaS) In The Corporate Environment

Greg Spehar MBA, PMP, PMI-ACP

spehargreg@yahoo.com

Abstract

More and more enterprises are adopting software-as-a-service (SaaS) "Cloud" applications. These applications are developed and maintained with an Agile Methodology, increasing their quality and performance. How can an Enterprise integrate these SaaS systems in an effective manner without significant risk to the organization and a decrease in application/process quality? An agile data migration and process integration methodology can be defined to ensure a high probability of success with the highest quality delivery when integrating a new system into a corporate environment.

Many of the new applications being developed today are cloud-based SaaS[1] applications developed using agile methodologies that are used by multiple customers or clients. These SaaS applications need to be integrated into the corporate environment in a logical and methodical manner utilizing agile methodologies to ensure that the client organization can meet their performance needs. The problem that needs to be solved is the solution a SaaS application has defined will only include 50% to 80% of the processes implemented by the client organization. For the client organization to fully utilize the SaaS software all impacted processes must be migrated or changed. With the use of change management techniques as well as a thorough understanding of the SaaS agile process, an agile data migration and process integration methodology can be defined to ensure a high probability of success while minimizing the disruption.

This approach was utilized on a Portland based organization, which allowed them to establish their go-live date and meet that date with minimal process impact. This agile integration approach can be standardized and utilized with any organization that is struggling to tame their data migration and process integration efforts into a vendor's agile SaaS implementation.

Biography

Greg Spehar is a Senior Project Manager at CorSource Technology Group Inc. working on agile projects that bring customer value developing and integrating applications. Over the past 15+ years he has been a leader in consulting projects that range from software development to integration of Software as a Service (SaaS) solutions. He has worked with non-profits, healthcare, state/federal organizations, apparel manufacturing, energy, financial and other organizations.

Greg has worked extensively developing and honing his skills in Project Management, Project Leadership and Change Management. He has a BS in Aerospace Engineering from Purdue University, his MBA from University of Texas at Austin and Erasmus University in the Netherlands, and he has earned his PMP and PMI-ACP from the Project Management Institute.

1 Introduction

When working on a migration project there are many challenges and issues that need to be addressed that need to be organized in a fashion to ensure that the project has the highest probability of success. In this paper we will look at using the agile philosophy and approach as a defining framework since in most cases the relationship between the vendor and the client cannot be held within a fixed contract when the vendor is using an agile development approach for their product.

The first step out of the gate that needs to be established is the agile notion of Customer Collaboration over Contract Negotiation[2]. If this approach is not taken from the start the entire project will be in jeopardy. The primary reason is inherent in an agile project, as the train of software updates will not be integrated in a linear process. We shall see in this paper, the integration effort must mirror the development effort to have a significant chance of success. To best illustrate this approach we will first decide on the type of SaaS vendor configuration we will explore in this paper. There are several kinds of SaaS vendors and they can fall into the following three categories as shown in Figure #1.

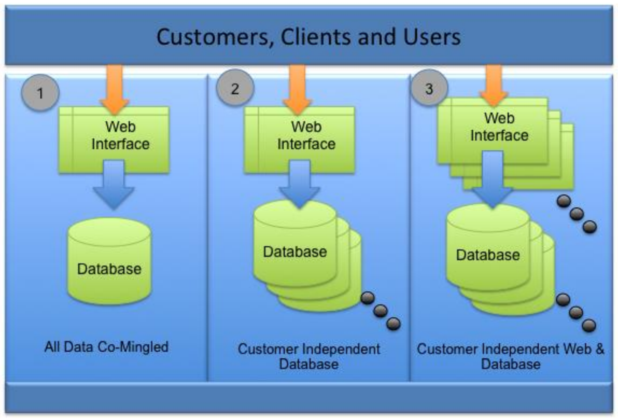
Customers, Clients and Users
1
Web Interface
Database
All Data Co-Mingled
2
Web Interface
Database
Customer Independent Database
3
Web Interface
Database
Customer Independent Web & Database

Figure #1: Different Integrating Approaches for SaaS Projects

For this paper we will focus the second type of SaaS vendor, The Customer Independent Database Vendor, this vendor has separate databases but uses a common web client code. Within this construct we then have several constraints and levels of freedom at the same time. Those constraints are as follows:

1) Independent Databases allow daily data dumps into a “data warehouse”
2) Not all client change requests will be implemented on face value
3) System Configuration will predominate

As stated before, we have a SaaS system where system changes are managed through the use of an agile approach. Because of the approach dictated by the agile process we have a level of freedom as follows:

1) Contract Negotiations are focused on larger grained features amounting to a dozen or so capabilities (No matrix details are defined).
    - Giving the vendor and client leeway in the kind of features and changes that will occur to meet the needs of the client.

2) Contract Payments are broken in terms of milestones over a period of time with the go-live date plus some post go-live stabilization period amounting to the total system implementation cost.
    - Giving the client the leverage necessary to ensure features are in place before go-live and post go-live.
3) Process mapping meeting change management is driven by backlog priority instead of details in the contract.
    - Giving the client extensive opportunity to change and shift priorities as they discover what is important and what is not important as they learn more about their business as time passes.

Within this context we can define project goals that include the following:

1) Data Migration and Configuration
2) Process Evaluation and Change Management
3) Reporting and Data Integrity
4) Portal Integration into Existing Website/s

We can then define some ongoing project maintenance processes that ensure the project's success:

1) Stakeholder Management and Communications
2) Risk Management and Action
3) Action Management and Follow-up

This paper will not cover in detail the known processes around Project Management body of knowledge[3], but articulate the most critical factors of the PMI process or changes to the PMI process that are relevant to an agile SaaS project's success and quality. Let's dive into the project communications processes first.

2 Communications

2.1 Stakeholder Management

As we all know, when managing a critical project, the onboarding and management of the stakeholders is the #1 job of who ever is responsible for making this kind of change happen[4]. The key items to ensure your success (See Figure #2) with your Stakeholders are the following:

1) Set up routine meetings and status reporting
    - Consistent communications over time ensures comfort for leadership
2) Never delay or do anything that would harm the sense of trust
    - Do not delay or sugar coat bad news if it happens!
3) Establish the most critical aspect of the project
    - They are after new revenue, operational savings, etc.
4) Focus on what can be done and how resources are managed
    - Don't bring problems to solve, bring solutions!

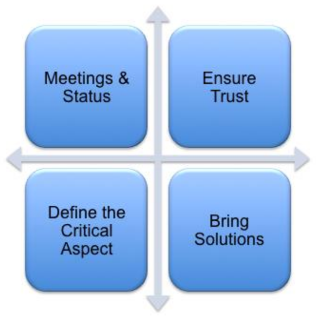

Figure #2: Stakeholder Management Responsibility

The communications on this project will be also critical in the area of the Stakeholders and the immediate team that is working on the migration effort. These groups are the most important to maintain a good solid face time and building an honest open relationship. This kind of environment promotes the immediate identification of risks and then their possible resolution. This component of trust is key in an Agile SaaS Integration effort. Finally, the rest of the organization that will be impacted from this change will be explored within the Change Management section.

2.2 Risk Management and Action

Risk management is another area that is not taken as seriously as some people might think it is needed[5]. The challenge which can appear in an integration project that is run in an agile like method is the necessity to maintain and mange the risks that will inevitably appear on the project. The need for the project team to identify them early and as quickly as possible allows the team to make important decisions that can have many months impact or ultimately kill an integration effort. The steering committee working with the team must trust that the team will provide the necessary decisions to ensure the technology can perform with their organization in the best way possible; even if that includes the difficult decision of cancelling the project entirely.

Typical risk management tools can be made and defined, but for this project the risks that should be closely identified are as follows:

1) Client Project Team Personnel Risks
2) Change Management Risks for Client
3) Staffing Issues with Vendor
4) Delivery of Feature Sets in a Timely Fashion (Burn rate)
5) System Performance once the system is in place
6) All normal technical challenges such as bandwidth, desktops, etc.

2.3 Action Management and Follow-up

The most interesting aspect of any integration effort with systems that are pre-defined and configured on site is the lion share of the activities required at the client site will be focused on the simple question:

Can the software do what I need it to do to meet my process objectives?

If not, what changes (Requirements) can we define to help the Vendor conceive of the most effective approach within their software?

NOTE: In this example of a Customer Independent Database Vendor, the vendor is considering many other vendor's needs and the impact to them when using the software.

The majority of the actions being defined for the client team will be around how to ensure that the new system can meet their demands at the process level. If not, what steps are needed to augment the system and will those changes meet the needs of the client processes. In some cases the changes will NOT be adequate and the VENDOR may not be willing to make the changes. In these situations the client must be prepared to change business processes to meet the methods that are outlined by the vendor or make a decision to abandon the effort.

Experience has shown that the latter option never happens and a compromise can be developed to the benefit of both parties. The usual approach we have been witness is the willingness of the client to change their processes if the Vendor can state a good case as to why the software cannot be changed. In most cases the client's proposed changes fundamentally change the assumptions of the SaaS system and thus will have a wide impact to other clients the SaaS vendor is attempting to satisfy. Usually in these kinds of situations the client discovers that their process approach will be more effective if they adopt the Vendor's technique.

All these items need to be in place to communicate and track the details to lead and ensure the project success. The following sections are the constructs of the overall effort.

3 Migration

3.1 Data Migration and Configuration

As with any integration of a system into an environment, the data becomes one of the key factors of success (See Figure #3). The movement of the old data into the new system includes several areas of concern:

1) Data that does not translate cleanly into the new system
2) Configuration data
3) Leaf Data - Dropdowns
4) Data that is new or is in places hard to get

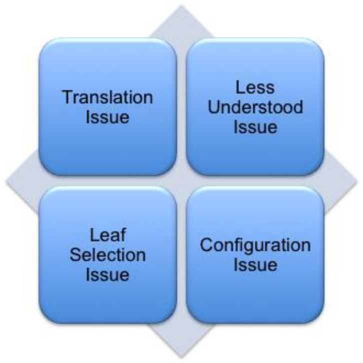

Figure #3: Data Issues to Resolve

Data that translates cleanly will not be data that you will normally have challenges with and in most cases your technical team will have an easy time getting to that data and translating it successfully into the new system. The challenges include data which one might think needs to be translated in the same way turns out to not be the same in the new system. Most of the vendors will assist in this work of ensuring the data is matched perfectly, but the general approach to management of data transfer is the agile approach to putting things up in the system as soon as possible for the client and their business people to review.

The following is a general approach to pushing data up to a test system over time:

1) Test System Release 0 – Generally has a few hundred records or less for as much of the system data that is easily identified.
2) Test System Release 1 – This release will consist of over 70% of the necessary data if possible. This will be the first time the users of the system will see how things work and begin to really evaluate their processes.
3) Test System Release 2/3/4 – These releases are all being performed to hit the goal of near 100% of the data being scripted.
4) Go-live Migration – Usually occurs over a weekend or an entire week depending on the vendor at hand.

The most critical notion around this approach is that the client should be purchasing 50% to 80% of the features[6,7] already in place, so moving 50% to 80% of the data should be something that is very doable in a short period of time. (Within a 6 month period of time realistically.) Once this Release 1 happens, the most severe risk has been managed and the team can dig in to fully understand the gap that exists in data and system functionality.

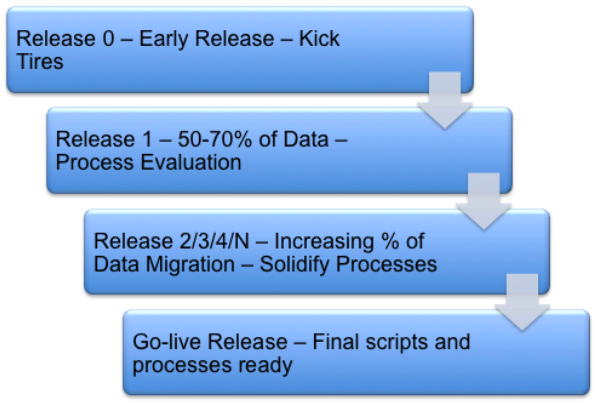
Release 0 – Early Release – Kick Tires
Release 1 – 50-70% of Data – Process Evaluation
Release 2/3/4/N – Increasing % of Data Migration – Solidify Processes
Go-live Release – Final scripts and processes ready

Figure #4: Agile Based Migration of Data Approach

One of the most notorious and misunderstood component of SaaS integrations in this realm is the necessity of configuration management. Since the system is a shared system, there may be hundreds or thousands of configuration settings that need to be turned off or on depending on your specific set of use cases. The vendor will do their best up front to manage this transition but will not fully understand the ramifications of the setup till the software is put under stress.

When evaluating process steps and process approaches, the project team should always ask the vendor integration team if the software does something one way or can it be done another way. That maybe the case as the configuration setting needs to be updated properly. In many cases there will be a feature request that comes back to the team as a configuration setting. As systems can become complex, the SaaS vendor may not remember all of the configuration settings; only when the problem is encountered can the developer discover that a configuration setting exists for the solution. NOTE: As many of the SaaS systems grow organically over time, the usage of the systems becomes more important than the documentation of the system, hence the sheer lack of documentation for larger SaaS systems will occur since the system changes much too rapidly for any documentation to be maintained cost effectively.

The management of Leaf Data is also an activity that is not entirely dealt with in the context of its importance. The need for the integration team to properly populate the data is clear. Mischaracterized leaf data results in improper loading of data that requires that the process be repeated. Proper review and care should be taken in analyzing all relevant dropdowns and the resulting updates in the configuration files should be effective. A couple of important notes on this effort:

1) Try to do it right the first time
    - Improperly loaded leaf data can have an impact to migration if discovered late in the process, more due to the business agreements required to finalize the data to be used.
2) Make feature changes right away if there are system limitations
    - These structural changes can take some time to make and these risks need to be mitigated early in the process or you will find you will need to delay the project since groups of people are not able to agree on a specific way of representing the leaf data.

Data that is new or in places hard to get will be the most challenging of the data to migrate. The project should expect this kind of data to be discovered and migrated in some cases near the start of the process. The lion share of this challenge will appear at the tail end of the project when the “Necessary” data is discovered in new processes and new excel spreadsheets that “Have to be integrated” into the system at the last minute. A gentle question should help establish perspective, “If it was so important, why are we only discovering it now? And if so, can it not wait till after the integration is complete?”

More importantly this discovery tends to fall into the general overall challenges of unknown processes, new processes or little understood processes that will require the creation of the transaction data as well as the creation of new configuration changes and drop down leaf data. These items should be properly evaluated and determined to be included or not in the overall integration process. We will explain more in the process evaluation and change management section on how that can be done within this agile approach without derailing the project.

3.2 Process Evaluation and Change Management

The most telling issue of any integration project is the ability for the team to properly evaluate their processes to ensure that the team has the ability to implement these processes into the new SaaS software. There are several schools of thought around process re-engineering efforts but they fall into the following two groups[8,9]:

1) Perform process re-engineering after the migration has happened
2) Perform process re-engineering before the migration happens

Most people move their decision making to #2 since they think that there is a better chance of making true change occur since a new system is being implemented. This makes sense at the surface but we should also consider the other perspective that the additional activity of re-engineering while switching software systems is one fraught with risk.

There are three re-engineering approaches that can be utilized:

1) Make re-engineering efforts prior to migration and implement processes as best one can with the old system
2) Make re-engineering efforts during the migration efforts
    - Only effective with organizations that are able to absorb the risks of broken processes and significant change management endurance. Organizations that are very stable and people have embedded themselves into their processes will find that the other two options may result in better performance.
3) Make re-engineering efforts after the migration is over and endure the rework that might be necessary to change the software

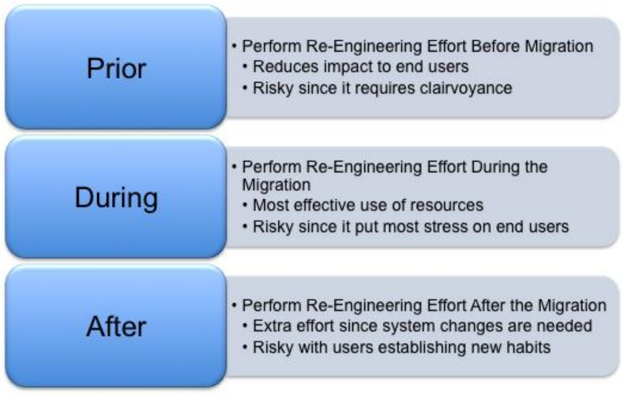
Prior
• Perform Re-Engineering Effort Before Migration
• Reduces impact to end users
• Risky since it requires clairvoyance
During
• Perform Re-Engineering Effort During the Migration
• Most effective use of resources
• Risky since it put most stress on end users
After
• Perform Re-Engineering Effort After the Migration
• Extra effort since system changes are needed
• Risky with users establishing new habits

Figure #5: Re-Engineering of Processes Options

Either path results in necessary changes to the vendor SaaS system and the need for a way to evaluate the importance of those changes. The first step in defining the methods to building a good product backlog for the vendor to manage is to not think of the Agile SaaS Integration effort as a waterfall effort. The reason being is that as time passes the client and vendor will have more opportunity to understand what is working in the system and what is not working in the system. This agile flexibility that can be experienced should become a pattern that will be adopted throughout the project. An approach that can be adopted (Figure #6) is were there are several levels in which the system can be evaluated and they are as follows:

1) First pass evaluation – Does the system meet the most basic features?
    a. Generally occurring during “Test System Release 0”
    b. Focused on finding show stopper features
    c. Opens door to contract negotiations if major issue found
2) Multi-Pass Evaluation of Feature Changes
    a. Generally occurring with “Test System Release 1 through go-live”
    b. As-Is process evaluation
    c. To-Be process evaluation
    d. Training Documentation (Also workbook)
3) Fully Acceptance Testing Walkthrough of all To-Be documents
    a. Generally occurring with “Test System Release Pre-go-live”
    b. Last determination for Go/No-Go Decision

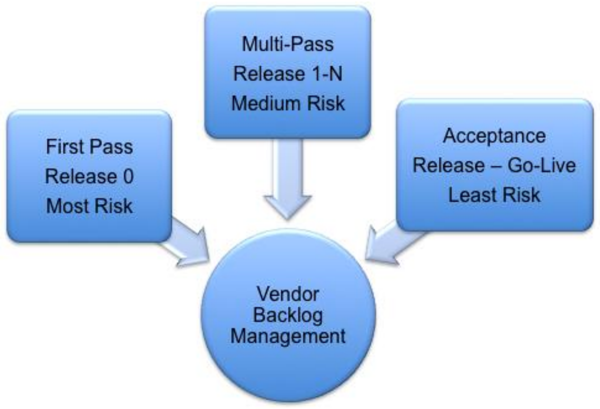

Figure #6: Process for Eliciting Backlog Changes to Vendor

In this approach there is the notion that large high-risk issues be discovered up front and properly scheduled for system updates or the determination that the risks are so great that the project be terminated. The later passes discover more and more detail upon each pass giving users a more structured process for evaluation and reporting of issues.

Once these issues are identified they are logged and given a priority that should be at three levels of importance:

1) Level 1 – Must have for go-live, NO WORKAROUND available
2) Level 2 – Must happen soon after go-live, temporary workaround available
3) Level 3 – Can happen anytime after the go-live event, workaround in place

Of all of the lesson’s learned for this approach, this factor was the most invaluable since it allowed the technical staff to push back on the business staff to challenge the need to call something a Level #1 when it is clear that it is a level #2 or #3 (Figure #7).

Given this model, it became clear that not all of the processes would be managed to the level that the business would have liked or wanted. It also allowed the business to understand that some things had a higher need than their own specific change and they should be prepared to live without the change for some time if possible. This approach then applies nicely to those last minute process discoveries that always appear at the last minute as projects are about to go-live. The Integration team will then have the chance to challenge these processes so that the business can make a good decision to attempt to integrate these new processes or not prior to the go-live event.

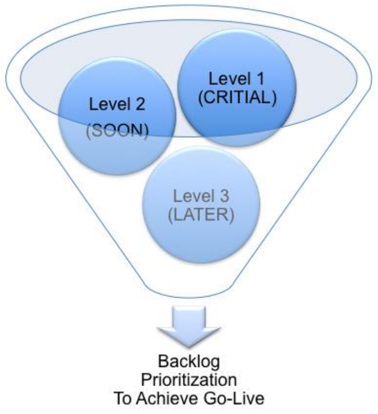

Figure #7: Backlog Prioritization to Chase Go-Live Within an Agile Framework

This brings us to the final challenge in this area that was faced. That is the need to document what the new process is and then to train the staff on those new processes and software usage. The change management technique that is mostly followed by most organizations is called the ADKAR method (Awareness, Desire, Knowledge, Action and Reinforcement). In the project that utilized this approach we found the staff to be well prepared to engage the development of the training materials and we found the staff continued mentoring the use of the new system well after the go-live event occurred.

For the project that was part of this effort, the team selected the training and documentation approach of writing just one set of documentation that would be used for training and as a desk reference. These word documents had all of the processes for each of the areas with the steps to perform the work. With each of the project releases revealing more and more of the application that could be used to define client processes (Figure #8). Where there was gaps in the software, since it was still being developed, work arounds had been put in place to ensure the go-live could still happen. Those workarounds were documented right into the training materials, with the future process also documented, but caveated with the message to not use until the change has taken effect. Once the change went into play and was verified, the documentation could be quickly updated to show the new process.

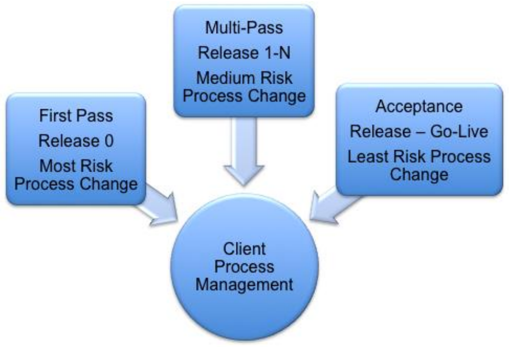

Figure #8: Process Risks Discovered and Challenged Early and Often

Since there is so much moving at the same time, every sprint cycle the client had the chance of re-organizing the product backlog and making changes to what they wanted the software to do and how they were going to manage their existing processes. In some cases deciding not to implement changes that they thought they needed since they decided to take a different approach with their processes. ***And, most importantly, in all of these cases we did not have to negotiate a change to the contract.***

3.3 Reporting and Data Integrity

Another important aspect of a system is the reporting that is generated since the reports give the management team the level of detail they need to make the right choices and ensure that the organization functions properly.

Many of the system changes that occurred at this level came from the re-organization of the required reports and in some cases the need to capture or look at the data in a different way. These changes, pushed through the product backlog allowed the team to make changes based on need.

As such the team was able to quickly focus on delivering the reports that were required for go-live. These changes reduced the total number of required reports to a more manageable level and the final reports required for go-live became a more meaningful number and in the end more useful.

In the implementation that utilized this approach in this paper, the Vendor had implemented a "day-later" data dump process. So every night, the database was delivered to the client's ftp folder and the client then pulled their database into their own environment. This allowed the client to customize specific reports that did not need real-time data. Additionally, many of the Vendor Reports could be generated into excel spreadsheets that allowed additional reports be generated.

Within the context of flexibility, the integration team was able to utilize an agile approach to prioritizing and building the necessary reports to ensure that the team could meet their go-live commitments. This approach additionally, gave the team options when the vendor would decline to update a report in a given manner due to other customer conflicts.

Finally, as part of the post-go-live effort, the team performed several assessments to understand what Data Integrity tools should be created to ensure that the data going into the system is not a problem as the system expands and people use it in ways that was intended and not intended. Due to the fact the system was a multi-user system the team found several cases that if the process was not followed and some of the screens where used improperly, some data integrity issues would appear. Instead of trying to get the vendor to change their interface (Which they most likely would not do) they could generate scripts that would run nightly or weekly to identify situations where these issues would appear and the staff then could correct.

3.4 Portal Integration into Existing Website

Most systems now days have Portal integrations and existing websites that represent what they do as a company. These external systems will now have access to screens that can update the content that might otherwise require a phone call to someone to update the information directly into the system. During most projects you will find once the Agile SaaS Integration approach proves to be effective, the Web Integration team will seek out the same effective process and procedures to ensure their sites go-live in the same successful manner. Additionally, if there is a tie to the SaaS system to the customer's web site, there may still be changes that are needed in the SaaS system to meet their web site design needs. These web-based changes can also be defined and prioritized and feed to the SaaS vendor within this agile framework.

4 The Real World Example

4.1 Non-Profit Integration of Granting and Donor Management System

The project, where we used this framework for the first time, was a large non-profit which had selected a SaaS vendor to host and provide all of their non-profit system needs. We had to move all of their funds management and donor relations systems to the new product. The challenge with this migration was that they had about 50-60+ people that had to be on the new system all at one time. These people had been doing the same job in the same manner for years, so change management was going to be key the project's success. The framework that we have defined in this paper was utilized and was effective in not only identifying the necessary changes in the system, but also proved to be an effective manner in which to build project champions within the steering committee.

The use of the release cycles for migration allowed the core team to see the changes happening to the system and allowed them to peal the onion back on their processes as they explored and defined their "As-Is" and "To-Be" Processes. This effectively allowed the team to develop a level of comfort and confidence that was translated to the executive team and the end users that eventually had the responsibility of implementing the new system.

Overall the project was extremely successful in managing the organization to their new system due to the competence of the entire team. But the most effective skills that helped to perform the necessary risk management to ensure the project's success was as follows:

1) Project Managers – Seasoned and pro-active project managers for the Client and Vendor
2) Technical Resource – Effective senior technical resource is a must to package and deliver data
3) Business Analyst – Process Based – To help define the documentation & drive process change
4) Business Analyst – Report Based – To help manage the technology and report delivery

Finally, the full support of the Executive Team and their engagement once issues are raised for their attention cannot be missed. Any organization in the midst of change cannot succeed if their leadership is not an interested, motivated and engaged party in the effort. This Agile SaaS Integration Methodology allowed that to happen on this project with the ever-growing system and confidence of the people involved in the overall effort.

5 Conclusion

This resulting Agile SaaS Integration framework provides the basis to utilize one of the most revolutionary approaches to software development, the agile methodology, within an integration environment to almost guarantee success, or at the very least push the most risky components into the spotlight to either address them early or challenge the project and organization to consider delaying or cancelling the proposed project. The failures of projects are mostly rooted in the project's inability to reveal the risks that challenge the project and the methods in which those risks are evaluated. This framework provides some of the first documented steps in defining an Agile SaaS Integration process that can best guarantee an effort rooted in establishing and maintaining quality throughout the risk management effort where the resulting effort can put the least amount of stress on the organization while delivering the value that the project set out to achieve.

References

1. Williams, Alex. 2013. “Forrester: SaaS And Data-Driven “Smart” Apps Fueling Worldwide Software Growth.” techcrunch.com, January 03. http://techcrunch.com/2013/01/03/forrester-saas-and-data-driven-smart-apps-fueling-worldwide-software-growth/ (Accessed August 22, 2014).
2. Cunningham, Ward. 2001. “Manifesto for Agile Software Development,” http://agilemanifesto.org (Accessed August 24, 2014).
3. Project Management Institute. 2014. “Project Management Book of Knowledge Guide and Standards,” http://www.pmi.org/PMBOK-Guide-and-Standards.aspx (Accessed August 24, 2014).
4. PMI’s Pulse of the Profession In Depth Report. 2013. “The high cost of low performance: The essential role of communications,” http://www.pmi.org/~/media/PDF/Business-Solutions/The-High-Cost-Low-Performance-The-Essential-Role-of-Communications.ashx (Accessed August 24, 2014).
5. Hamilton, Byatt and Hodgkinson. 2011.“Risk Management and Project Management go hand in hand.” cio.com.au, May 03. http://www.cio.com.au/article/385084/risk_management_project_management_go_hand_hand/ (Accessed August 22, 2014).
6. ZDNet/Topics. 2013. “Cloud: How to do SaaS Right.” Zdnet.com, March. http://www.zdnet.com/topic-cloud-how-to-do-saas-right/ (Accessed May 03, 2014).
7. Gilbert, Jody. 2013. “Executive Guide to Best Practices in SaaS and the Cloud.” Zdnet.com, March 01. http://www.zdnet.com/executive-guide-to-best-practices-in-saas-and-the-cloud-free-ebook-7000012032/ (Accessed May 03, 2014).
8. King, Julia. 2012. “The rebirth of re-engineering.” computerworld.com, September 10. http://www.computerworld.com/s/article/9231002/The_rebirth_of_re_engineering (Accessed May 03, 2014).
9. Gartner/ IT Glossary. 2014. “Business Process Re-Engineering (BPR),” http://www.gartner.com/it-glossary/bpr-business-process-re-engineering/ (Accessed May 03, 2014).