Agile Supply Chain Software Implementation

A Continuous and Incremental Improvement Approach to Successful Supply Chain Software Deployments

By

Jim Piermarini & Mark Rockey
Executive Summary

For most supply chain and IT executives, achieving supply chain software implementation and enhancement success seems to require perfect alignment of the stars and planets. That is, the stars of business processes and the planets of software expertise. While this type of convergence seems to be a once-in-a-life time occurrence in traditional project approaches, Profit Point delivers successful projects for Fortune 500 companies time and again. Our track record of repeated success is largely due to the agile and iterative approach we take to software projects.

Traditional software projects employ a “waterfall” or “big bang” approach where development teams work in isolation for six to twelve months based on business users’ documented requirements. Then, the development team returns with a product that almost definitely misses the true objectives of the project.

Based on our experience, it is a costly mistake to work in this serial manner. Business and technical requirements need to be refined based on a deep understanding of the technical limitations of the existing software landscape, and frequent product testing by business users.

Traditional software projects that employ a waterfall approach can easily become drawn out and exceed budgets when the product is tested too late, and the wrong requirements become too intertwined within the deliverables. Many times, this results in costly change requests, and soured relationships between the software vendor and you.

Profit Point takes an iterative approach to supply chain software projects by leveraging the concepts of agile software development and continuous improvement to deliver projects that directly addresses the users’ objectives and more closely meets budget and time commitments.

We believe in the benefits of business users working alongside the development team, adjusting requirements as frequently as needed, and shifting the focus of projects when the original requirements cannot be accomplished.

For the last 20 years, we have successfully and satisfactorily implemented and enhanced supply chain software for numerous Fortune 500 companies by using an iterative and agile approach.
## Contents

Executive Summary.................................................................................................................................................... 1

1. Agile – a Prescription for Pain Avoidance ........................................................................................................... 3

2. “Big” Software Projects – Destined to Disappoint ............................................................................................... 4

3. The Profit Point Way: Deploy, Iterate and Avoid Pitfalls ....................................................................................... 6
   Determine Your Objectives – where are we going? .................................................................................................. 6
   The Small Bang – a check on Project Viability ....................................................................................................... 8
   Iteration – Small Surprises Not Big Shocks ............................................................................................................ 9
   Pivoting – Course Correction .................................................................................................................................. 12

4. Results are In – Iterative Software Projects Deliver Success .................................................................................. 14

5. Conclusion & Recommendations ............................................................................................................................ 15

6. About the Authors ..................................................................................................................................................... 16
1. Agile – a Prescription for Pain Avoidance

You don’t need to have experienced it to know that failed software implementation projects are painful. These failures are always difficult but in some companies, failure can cost you your job, especially if you are the project leader. This white paper will address the common sources of failure and provide you with an approach to increase your chances for success.

Why do projects fail? – And what can be done?

Buyers of business software often presume that the software product company is an expert who knows what their client wants. In reality, software that is not tailored to a company’s processes are starting points at best. At worst, they are harmful to business operations, bottom lines, and customer relationships. The smartest companies know that while vendors can be experts in software development they likely are not experts in their clients’ internal business processes. This gap between what the business wants and what the vendor understands is a common source of project failure.

Agile software development methodology was introduced to the technology world over a decade ago to address this gap. Today many companies have applied the Agile methodology to implement software as well as develop it, allowing software implementation projects to more closely meet their users’ needs in less time and for less money.

The iterative approach of Agile methodology contains many parallels to the continuous improvement approach in manufacturing and logistics. We at Profit Point have found through many years of working with fortune 500 companies that the iterative development cycles for implementing supply chain software delivers the best functionality, simplifies change management, and generates concrete cost savings.

In this paper, we hope to share our experiences with you and show you how our love of working closely with our customers and using the Agile methodology go hand-in-hand to achieve a low cost and high success rate approach to supply chain software deployment.
2. “Big” Software Projects – Destined to Disappoint

Why do so many software projects fail? After all, it is not that complicated, is it? The customer provides a list of required features, functions and processes. The vendor provides a price for the project. The two parties shake hands and meet again after six months. Voila - a perfectly functioning piece of software! Unfortunately, this rarely happens.

There are several major downsides to approaching supply chain software projects in this “Big Project” approach:

1. Requirements are not properly communicated (or even known!)
2. Requirements change over time
3. Users are not involved throughout the development of the project and
4. Projects tend to focus on deliverables and not the actual business outcome.

Requirements are Not Properly Communicated – the Knowledge Gap is no one’s Fault
While most companies are very familiar with their internal supply chain processes, they find it hard to explain them to external parties. Most supply chain professionals do not mean to sound complex or confusing. Your hardworking team members know how to efficiently explain what they do to the people they most commonly work with. Therefore, there is typically a body of knowledge and understanding that they assume the people who they are talking to already know. The knowledge gap here is not between folks within the company, rather it is between the team members and the software company. We think that ultimately the source of the knowledge gap is the difference between the two companies… one is a manufacturing company, the other is a software company. They simply have different world views and cultures. The knowledge gap is not anyone’s fault. It is a natural outcome of having a highly specialized focus on your business – to the exclusion of their business.

For example, if Company A indicates that they want to create automated raw materials staging to reach a 90% service level, they may assume that their software development Vendor X understands they mean the service level at their distribution centers – because that is how they think of it. However, since Vendor X typically measures service level at their final customer, the Vendor’s solution would deliver rather different results.

Requirements Change Over Time – Scope Creep: Friend or Foe?
In the fast-moving world of manufacturing and logistics, requirements change quickly when business shifts with customer demands and supply sources. Over a period of six months or a year in a large (non-Agile) software development project, requirements will likely change. Major business changes will add pressure for change orders that will be in tension with project management’s desire to hold features constant to the original scope. Every change order that is declined makes the delivered solution more obsolete and more susceptible to being labeled a failure by a business user who
requested the change. Every change order that is accepted puts the original timeline and budget at risk. The label of failure lurks behind the delayed rollout or budget overrun too. It may seem you are doomed either way.

For example, Company A clearly specifies that the logistics routing optimizer must consider all the logistics costs within the U.S. at the beginning of a six-month project. However, after the third month of the project, a sourcing agent found raw materials from Malaysia and therefore import duties must be included in the optimization. This represents a major change in the requirements and will extend the project timeline and possibly cost or risk having the tool be branded as incomplete or useless.

**Users are Not Engaged During the Development Process – The Voice of the User is Key**

Traditional software projects depend on project managers or product owners to communicate between the customer and the vendor. While this may appear to be an efficient model due to the centralization of communication, actual users provide major insights to the development process. Users are the ultimate consumers of the software product and will work with the software in their daily routines. Without users’ frequent involvement and feedback, vendors may still deliver a functioning piece of software but in a way that the user had never intended to use.

For example, Company A’s initial requirements may specify that an automated plant scheduler considers staff work schedules, raw materials availability, and machine set up time. However, at the end of the project during user testing, the project manager discovers that users need to manually adjust staff work schedules for each day of the week based on some additional information that does not exist in any system of record. Incorporating this additional information may represent a very large change to the scope of the project and extend it by months. This is neither a miscommunicated or changed requirement because users simply could not verbalize this requirement without seeing and interacting with the software integrated into their daily work.

**Projects are Focused on Deliverables and Not Outcomes – which may completely miss the point**

Deliverables can be defined as a certain functionality or a work unit (e.g. person-month). Large software projects are typically deliverable-focused because it is easy to measure and gives vendors a very definite way to meet their contractual obligations. However, the deliverables-based approach can often leave the client with something they derive very little value from. Without a mechanism to find and resolve the ‘gray areas’ of the requirements, we often encounter red flags in the project. It is natural that the path that is simplest for the vendor will be taken.

For example, Vendor X may deliver an automated optimizer for scheduling outbound trains that is optimized on total cost and derives its cost savings from running it daily to closely match daily demand variability. This is exactly what was in the requirements document. However; what is delivered may provide Company A with no additional savings because what is not mentioned in the requirements is that rail schedules are communicated to the railroad company only once a week, and daily changes are ignored.
3. The Profit Point Way: Deploy, Iterate and Avoid Pitfalls

Profit Point’s approach to deploying, developing, and enhancing supply chain software applies the principles of Agile Software Development. Agile software development is an iterative approach to developing software. Wikipedia provides a straightforward and encompassing definition of Agile software development:

*Agile software development describes a set of principles for software development under which requirements and solutions evolve through the collaborative effort of self-organizing cross-functional teams. It advocates adaptive planning, evolutionary development, early delivery and continuous improvement and it encourages rapid and flexible response to change.*

We combine our expertise in manufacturing, scheduling and logistics with our capabilities in software development, rapid prototyping and flexible contracts to deliver projects that squarely address our customers’ goals at lower costs.

This is how we do it:

**Determine Your Objectives – where are we going?**

The very first step of a software deployment project should be a clear understanding of your objectives. Objectives are not to be confused with requirements. Objectives are a statement of the business goals to be achieved. Requirements reflect the business' current understanding of the steps needed to meet the objectives. Very often, customers will write requirements in isolation before they meet with a vendor and transmit these requirements as part of the RFP process. And, very often, vendors are happy to unquestioningly accept the write up of requirements. Vendors can then assume that meeting the requirements based on their own interpretation will also meet your objectives. This is simply not true.

In general, there are two types of objectives: Business Value and Process Improvement.

**Objectives are not to be confused with requirements.**

**Business Value** objectives relate to a desired and measurable improvement in your company’s targets. For example, your company may be looking to improve customer satisfaction or enter a different market. Sometimes your business targets may be more quantifiable. For example, your company may be looking to reduce overall costs by 30% or increase service level by 10%. Tying your project to a business objective not only makes it clearer to your vendor what is at stake but it also allows you to objectively assess the value of the software deliverables.

**Process Improvement** objectives are more tactical but crucial to ensure the success of the project. Almost all software projects aim to improve the throughput of work and reduce human error. For example, you may be looking to automate scheduling of raw materials so you can reduce the amount
of waste or increase the number of shipments you can book monthly. Understanding process improvement objectives can help your vendor think about how to more effectively implement your requirements. We also recommend providing data on your processes so the vendor understands how the outcome will be measured.

## Iterative Approach to Supply Chain Software Projects

### Saves Time and Money

<table>
<thead>
<tr>
<th>PROJECT INITIATION</th>
<th>TRADITIONAL “BIG BANG” PROJECTS</th>
<th>PROFIT POINT ITERATIVE APPROACH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer identified need to pass New Activity Comments to the ERP</td>
<td>Long list of requirements is defined up front, Strict Change Request Policies are in place</td>
<td>Objectives are defined, Short list of user stories are defined, Change Requests are handled during sprints</td>
</tr>
<tr>
<td>TECHNICAL SCOPE CHANGE</td>
<td>3 (3) months into project, Change request is written up, Customer is surprised</td>
<td>1st sprint, one (1) month into project, Developers explain issue, Customer approves additional requirements for next sprint</td>
</tr>
<tr>
<td>Development Team Discovers Additional Work will be Needed to Pass Data to ERP</td>
<td>End of development, 6 months into project, Change request is written up, Project is paused until decision is made</td>
<td>3rd sprint, 1.5 month into project, Developers explains issue, Small incremental change request is approved</td>
</tr>
<tr>
<td>SHIFTING COURSE (PIVOTING)</td>
<td>End of project, 9 months, Change request is written up, Most of existing work has to be redone</td>
<td>2nd Sprint, 1.5 months into project, Business users explain need, Customer approves additional requirements for next sprint</td>
</tr>
<tr>
<td>ETL tool needs to be updated to facilitate the data move to ERP</td>
<td>More costly and results in extensive delays</td>
<td>Minimal delays and significantly more cost effective</td>
</tr>
<tr>
<td>FUNCTIONAL SCOPE CHANGE</td>
<td>After testing, business users realized need to edit Existing Activity Comments as well.</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 1.1 – Side-by-Side Comparison of the “Big Bang” vs. “Iterative” Approach to Supply Software Projects**
Typically, your objectives may already be outlined in your business case when you requested funds for your project. In that case, your task is easy. However, if your internal objectives are not already documented, it is good practice to get the right people in a room and put them down on paper.

To properly address your objectives, your vendor must first understand your objectives, your supply chain, and software deployment. Then, along with the Agile spirit, the team, comprised of your supply chain team, IT team, and the vendor, should collectively document your objectives in terms that are clear to everyone. This approach requires more interaction than many companies are used to, but in our experience, is critical to success.

The Small Bang – a check on Project Viability

Whether you are deploying packaged software (e.g. aspenONE) or developing customized software from scratch, the first step is the “Small Bang”. In the context of packaged software, a small bang will include the initial deployment of the solution out of the box with user access properly configured as well as the main data source integrated. For completely customized software, an initial working concept with elementary interfaces and sample data might be used. For an enhancement of an existing software product, the same concept applies.

<table>
<thead>
<tr>
<th>Packaged Software Deployment</th>
<th>New Customized Software</th>
<th>Enhancement of Existing Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Install software on intended infrastructure</td>
<td>• Develop core functionality in test environment</td>
<td>• Simple implementation of intended functionality</td>
</tr>
<tr>
<td>• Configure user access</td>
<td>• Integrate with sample data sources</td>
<td>• Integration with test data</td>
</tr>
<tr>
<td>• Integrate with main data sources</td>
<td>• Basic user interface</td>
<td>• Essential user interface only</td>
</tr>
<tr>
<td>• Map one or two major business processes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1.2 – Typical Objectives of the Small Bang by Project Type

Regardless of what is included in the deliverable, the end result must enable the team to evaluate:

1. Whether the main functionality addresses both your business and process improvement objectives
2. Whether the functionality delivered can be enhanced or modified to create the intended product
3. Whether the functionality is close to how the users intended it to work and
4. Whether the data that the product will interact with can be represented and manipulated accurately with the produced functionality.
Answering “no” to any of the above questions should prompt a deeper discussion on whether the requirements have changed and should be recognized, the right team is in place to manage this project, or the project should be ended if the desired outcome is not achievable.

**Iteration – Small Surprises Not Big Shocks**

The key to a successful supply chain deployment project is to be able to assess the current progress and adjust the execution frequently and quickly. To achieve this high level of agility, the vendor needs to deliver small and incremental functionality every few weeks (referred to as a sprint), require actual users to test and provide feedback, and adjust the product before repeating the cycle again.

If the project is proceeding after the initial Small Bang, a set of objectives are outlined for the next sprint. In this rapid iteration approach, customers must be prepared to make resources available to the technical team at the beginning and the end of each sprint. Typically, the team is composed of the actual supply chain professionals who will use the software, the IT project leader, and the vendor developers and project manager.

While the Small Bang may have had different objectives depending on the type of project, sprints simply inject enough functionality so that the business users can test it. Business users may outline several different requirements (often called stories) during sprint planning. Working with the developers, the team will be able to prioritize what stories can be addressed within the sprint period, and what the appropriate testing approach should be. At Profit Point, we use the online tool Pivotal Tracker to collaboratively manage the list of stories in our scrum meetings.

For example, in a scheduling software implementation project, the users were looking for a way to send some new information to their enterprise resource platform (ERP) that was not part of the existing interface. This is an enhancement to the current interface, and it was broken down into these stories:

**Requirement 1** - User needs to be able to add new Activity Comment in scheduling tool, and the information needs to be passed to the ERP system
Requirement 2 - User needs to see any errors that might have occurred as a result of passing the new Activity Comment information to the ERP system

While the stories listed above are well formulated and seem simple enough on the surface, there may be some underlying technical groundwork unknown to the business users. For example, while exploring how the data flows through the systems, the developers added a few related stories to deepen the scope of the enhancement. These were:

Requirement 3 - Scheduling System needs to be able to write new Activity Comment information to the ETL database of ERP system

Requirement 4 - ETL system needs to be able to pass the new Activity Comment information to the SAP ABAP code

Requirement 5 - ERP System (SAP) need to be able to put the new Activity Comment information away in the appropriate place (via ABAP Code) and return error codes

Requirement 6 - ETL System needs to receive error codes and write them to the database

Requirement 7 - Scheduling system needs to read the ETL database for error codes and display them to the user

Each story will be tested independently, albeit with test data, and each story has its own test plan.

In Figure 3.1, the stories are added to Pivotal Tracker, a story management tool, where users can prioritize them and track their progress during development.

Based on the development team’s understanding of the technical work and the users’ understanding of the importance of each story, the team may collectively decide to start with the first three (3) stories with a stretch goal of completing stories 4 - 6 within the two-week sprint. The discussion and negotiation that take place during sprint planning is expected, encouraged, and creates clear expectations of the expected outcome.

With the team aligned to what stories will be covered in the upcoming sprint, the rest are left in the “icebox”, a list of stories that will be addressed later in the project if deemed necessary by the team.

In the fast-moving world of manufacturing and logistics, requirements change quickly when business shifts with customer demands and supply sources.
At the end of each sprint as stories are completed, the business users will have the opportunity to test the deliverables. Testing is crucial because candid user feedback helps the entire team understand whether the functionality was delivered as expected, better than expected, or failed to meet expectations. If tests are not favorable, the users will explain why to the development team.

Continuing with our example above, let’s say the development team completed story points 1 and 2, and the software tested positively for its ability to allow the user to enter the new Activity Comment information in the scheduling system. Further, the system can write the new Activity Comment information to the ETL database.

However, it was seen that the scheduling tool would only save the new Activity Comment information to the ETL database for new scheduled activities, but not existing activities. The tests had all been established for adding a new activity, and new Activity Comments. Naturally, the users asked about changing existing activities’ Activity Comments.

In a traditional software project, all requirements would have been completed before this issue was identified, and it would come as a big shock to the whole team. With many downstream processes depending on this Activity Comment information, the cost to adjust dependent functionality could be tremendous and cause friction between the customer and the development team.

In this project team, as part of the rejection process for the story, the user explained that they needed to be able to change existing activities’ Activity Comments as well as create new ones. Thus, the story requirements evolved into something more suitable for the user, and the project adjusted to the new requirement immediately.
By identifying inconsistencies with actual user expectations regularly during the entire project and adjusting for them in an iterative manner, we ensure that users are fully committed to the project (because it’s based on their exact requirements), and that there are no surprises for either the users or the development team at the end of the project. Further, the iterative approach gives the team flexibility to control what is in scope and deliver the project within budget.

Pivoting – Course Correction

The previous example examined how a successful sprint provides momentum and renewed energy into the project even though some requirements may need to be adjusted. However, what happens if the project results are not even close to expectations? In this case, teams have three options. They can:

i. Stop the entire project and release the budget back to the customer’s company
ii. Use their remaining resources to further enhance an aspect of the project that more closely meets business and process improvement outcomes or
iii. Redefine objectives with approval by the project’s executive sponsors and continue the project.

Regardless of the decision, the team will have prevented the waste of resources by deciding to change course. This is called pivoting which is a major advantage to operating supply chain software projects in an Agile mode.

Pivoting is essential to Agile supply chain software deployment and the way Profit Point does business with our customers.
Profit Point firmly believes that concepts and theories need to be tested. If the test does not provide business benefit or process improvement, then the project should be ceased or shelved immediately for the benefit of the customer.

Returning to our example above, during the attempt to make the ETL system talk to the ABAP code, a version incompatibility was discovered. With the existing SAP version and the existing ETL tool version there was no way to make this story work properly. This raised the pivot point. Do we upgrade the ETL system or is the whole enhancement not worth the effort?

Based on the estimate of cost and time to upgrade the ETL tool, the team decided that the best path is to upgrade the ETL tool, and continue with both the enhancement and the larger scheduling tool implementation project. The point here is that this incompatibility was found quickly, and it might have been enough to derail the entire project were the costs to upgrade different.
4. Results are In – Iterative Software Projects Deliver Success

The iterative approach to supply chain software deployment and enhancement results in increased project success rate, higher financial returns, and stronger team cohesion.

**Increased Success Ratio** – Traditional software projects have a relatively high failure rate. Iterative projects that engage the business users, frequently adjust requirements, and focuses more on objectives rather than requirements are much more likely to succeed. Sponsors and leads of the iterative projects usually enjoy higher recognition from management teams due to the higher success ratio and are more likely to be approved when future projects are proposed.

**Higher Financial Returns** – Iterative supply chain software projects will generate superior returns, because requirements can be realigned to objects as the project team discovers new business and technical nuances. Traditional projects usually set requirements in stone until the end with even small tweaks resulting in lengthy and costly change request projects. This results in a high amount of wasted effort, especially when developers are not validating implementation of features with the end user often. Profit Point’s iterative project approach is not only receptive to changes, our simple engagement model facilitates changes with ease and with no financial downside. See the comparison chart below.

**Stronger Team Cohesion** – Due to the level of engagement required from the business users to deliver an iterative project, the level of buy-in from the end users will naturally be higher than traditional projects. Further, the development team will also be much more motivated knowing that they have access to and are ultimately accountable to the actual users of the final product. Lastly, because of the higher success rates of iterative projects, the team members are usually much more motivated and engaged.
5. Conclusion & Recommendations

Supply chain software projects are complicated because internal processes must be properly communicated, requirements can change over time, business users need to be involved, and business objectives are not always clear. Profit Point’s Agile deployment methodology addresses these concerns.

Profit Point approaches supply chain software deployment in an iterative manner which delivers functionality that closely fits the intended business objectives, reduces risk of overspending on failing projects, and creates a stronger understanding between business users and the development vendor.

Many fortune 500 companies trust Profit Point to use this Agile methodology to ensure that their software implementations are successful and get them the business benefit that they expect.

Now that you are armed with the knowledge of how Agile methodology works with supply chain software projects, ask your vendor about it before you embark on your next supply chain software project. We also recommend that you ask them about how they work in an Agile mode and be sure it fits with your needs.
6. About the Authors

Jim Piermarini  |  jpiermarini@profitpt.com
Co-Founder & CEO

Jim is one of the founders of Profit Point Inc, and has been helping companies improve their supply chains for more than 30 years.

With a BS in Chemical Engineering, Jim enjoys turning the theoretical into the practical, and in bridging the gap between the business and the technology.

Mark Rockey  |  mrockey@profitpt.com
Production Scheduling and Supply Chain Planning Practice Leader

Mark has been implementing supply chain decision support systems for over 28 years. Mark stands out in his ability to combine his technical strength with his organizational skills to manage projects and deliver results.

He holds BS and MEng degrees in Operations Research and Industrial Engineering from Cornell University.