International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Agile Methodologies in Salesforce Projects: Enhancing Delivery Timelines and Collaboration

Sai Vishnu Vardhan Machapatri, Sanjay Gorantla

Submitted: 15/11/2021 **Revised:** 12/12/2021 **Accepted:** 20/12/2021

Abstract: Agile methodologies are very popular in how organizations develop software. Agile is flexible, it enables teams to deliver faster and is oriented around collaborative workflows. Since Salesforce development projects are mostly agile due to the iterative and custom nature of the platform, dev teams that can embrace Agile flow feel its potential for faster delivery. This paper shows how Salesforce development teams can implement Agile methods, specifically focusing on reducing delivery timelines and improving team collaboration. It provides visuals such as diagrams and graphs to demonstrate how adopting Agile practices let teams reap benefits and provides code snippets that illustrate how readers can generate these visuals.

Keywords: *specifically, illustrate**,** visuals, adopting*

Introduction

The customer relationship management (CRM) platform Salesforce is highly popular and extremely effective because it is extremely flexible, scalable and able to fit into virtually any business process. With Salesforce projects increasingly being used to run customer relations at many companies, project managers often need to develop new methods to deliver projects fast and in a way that can respond to a changing CRM environment. Agile methodologies scored especially high in terms of flexibility, adaptability and speed Popular methods of delivering projects with Salesforce include traditional project management, agile approaches (such as Scrum and Kanban) and methods that combine traditional project management, Six Sigma (a methodology for process efficiency) and Scrum (emerging from the software development field, Scrum is a development process that delivers iterations over a defined interval). Both Salesforce administrators and IT managers wanted new project methodologies to suit the changing nature of working with Salesforce over time.

Whereas Waterfall-style approaches try to plan an entire system in advance and attempt to build the perfect system in a single implementation, agile development takes a different approach: we will break a project into a set of short development cycles (called a sprint) over which we will try to build some working increment of the system, which is delivered to the stakeholders regularly. The main promise of agile is that this approach will give us enough freedom to incorporate feedback and adapt the requirements if necessary. Combined with Salesforce's schematic nature (contracting complexity using declarative configuration, APEX, or LWC), agile becomes a logical way to manage complexity in Salesforce development.

saivishnusf@gmail.com
Gorantlasanjay483@gmail.com

Problem Statement

As the needs of organizations change, they face the ongoing challenge of managing Salesforce development projects with changing requirements and complex customizations in a timely way. Many traditional project management approaches, such as Waterfall, are poorly suited to the world of rapid and continuous change that characterizes the success of "modern" business development. Waterfall approaches tend to have lengthy feedback loops that result in slowed delivery, which creates significant challenges to demand-driven business needs. Collaboration across the spectrum, from developers to administrators to stakeholders, is typically inhibited depending upon how Salesforce is delivered and, unfortunately, miscommunications and misaligned priorities, especially between stakeholders and developers, are common outcomes. In this era of digital transformation, many businesses have come to rely on Salesforce for business-critical operations to deliver the latest innovations sought after by customers and business stakeholders. If we expect businesses to deliver innovations on time, in phase, and in budget, we need a more responsive and flexible approach to managing projects.

Solution Statement

So, how do we address these challenges while developing new requirements in Salesforce? Enter the famous Agile methodologies that help improve iteration and pivot during the development process. The major advantage of Agile is that it helps develop a Salesforce product in small chunks called sprints. By utilizing Agile delivery processes, you can release functional chunks of your Salesforce system more frequently, with the ability to review it more closely with your stakeholders who implement it and use it daily. This becomes more convenient for them as they can provide feedback and ensure timely delivery of what they need. Moreover, Agile helps you increase collaboration between developers, admins and stakeholders by encouraging communication through daily stand-ups and sprint reviews, thereby reducing miscommunication. They actively participate in the sprint review and allow more chances to give feedback and foster more collaboration and customer engagement. All in all, an Agile approach will help you improve your delivery timelines, manage changing requirements, foster collaboration, and achieve a quick turnaround in Salesforce development.

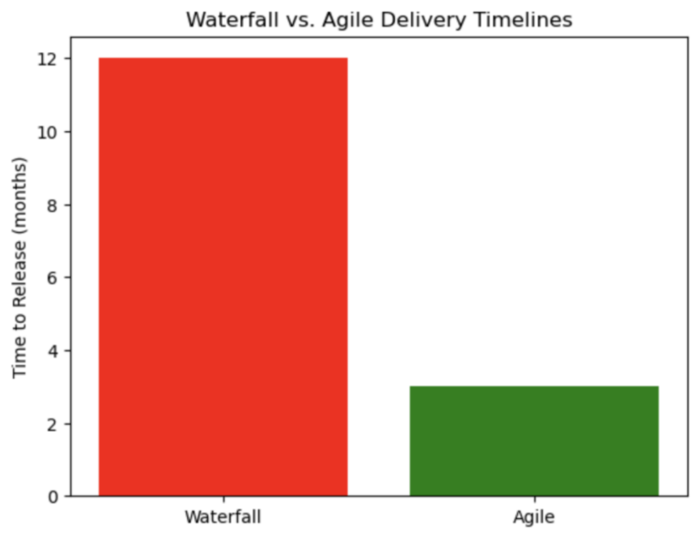
Agile Methodologies in Salesforce Development

In Agile Salesforce development, each sprint is geared towards delivering a working piece of software and getting feedback from the stakeholders at the end so that we could continue to work on the next sprint. What the Agile implies is that while stakeholders should get the deliverable by the end of the project (as in the Waterfall models), they get exactly what is developed within each sprint rather than a finished product. This is a particular advantage in Salesforce projects as they're released with a high rate of changes, and the business processes are constantly evolving.

In an Agile sprint at the same Salesforce project, the team might implement four custom objects, three workflows to automate processing in Salesforce, and four custom APEX triggers. Stakeholders provide feedback as the sprints evolve the system to match their company's changing needs. Salesforce's declarative tools (such as Process Builder and Flow) make Agile automatic for admins and developers, allowing them to produce automations that can continue to evolve over time without heavy rework.

Impact on Delivery Timelines

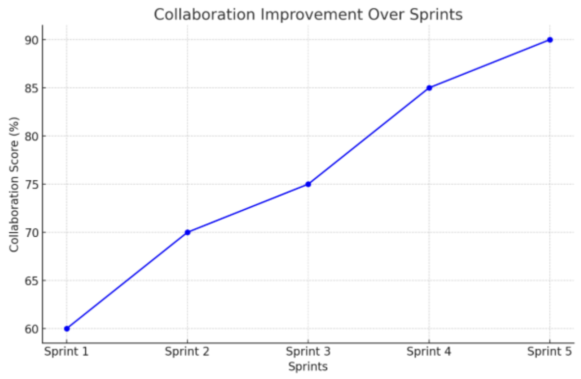
The biggest visibility benefit that Agile brings into Salesforce development is the drastically shortened delivery timeline. Among traditional project management methodologies, the Waterfall project methodology involves phases such as planning, design, development, and testing. Projects are executed in a linear fashion: once one phase is completed, another phase begins. Typically, this results in delayed delivery and feedback. In contrast, based on the same project timeline, Agile breaks the project into small work intervals called sprints, each of which is concluded with the delivery of a working product increment.



Improved Collaboration and Communication

This focus on transparency and collaboration makes agile methodologies a natural fit for Salesforce – development projects often involve teams of Salesforce admins, developers, and (most importantly) business users, and agile's daily stand-ups, sprint reviews and retrospectives are crucial to keep all team members in the loop about what's being worked on, what's the status of the project, what roadblocks exist, and what to work on next.

Since the value of Salesforce comes from coordination between many different teams who may be working on different aspects of the same platform at the same time, developers may be building APEX logic while business stakeholders are reviewing and approving adjunct reports or dashboard functionality. The iterative, open-channel nature of Agile encourages providing feedback at each step so that these moving parts can be dialed into each other.



Increased Flexibility for Changing Requirements

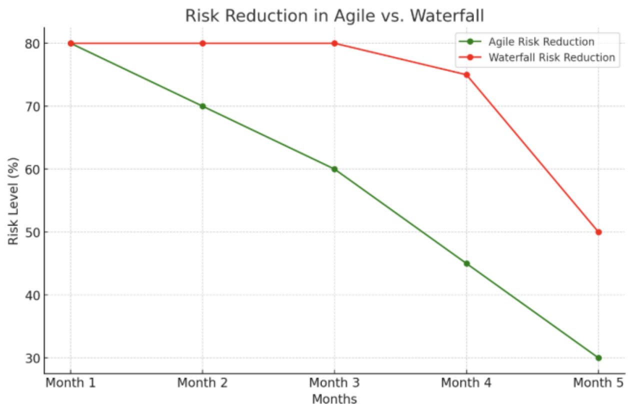
The business environment might change, and so the requirements for a Salesforce project might change as well. Agile approaches dealing with changing requirements quite seamlessly. With Agile, if stakeholders determine that items with higher priority will have to be ready first, they can adjust the Sprint priorities accordingly.

For example, a Salesforce implementation that begins by giving attention to lead management might need to shift attention to opportunity tracking or customer support later as the context changes. Agile allows the developers to pivot without major disruption.

Risk Mitigation Through Agile Practices

In terms of risk mitigation, Agile works very well. With the close feedback loops in Agile projects, problems can be identified early and dealt with before they can snowball into big problems. On the flipside, in Waterfall projects, these risks might not be revealed until close to the end of the project, when changes are often more expensive to implement.

Risk items in a Salesforce development environment can include integration risks, performance issues, or user adoption risks. By handling risks 'at the right time' – a core concept of Agile methodology – addressing risks during each sprint allows for iterative changes that prevent project failure.



Challenges of Implementing Agile in Salesforce Projects

Regardless of any benefits it provides, employing Agile methodology isn't without challenges when applied to Salesforce projects. First – Agile requires engagement and participation from key stakeholders who are willing to sacrifice a sprinkling of decision making in favor of regular, constructive feedback – and some larger enterprises simply don't know whether they can count on consistent input from all parties by the end of each sprint. Second – Agile challenges teams to adopt a different cultural mindset, fostering a sociocracy that's antithetical with changeless, top-down engineering, wherein decisions are subject to a handful of autocratic final choices.

A third challenge is balancing Agile's quick iterations with the lengthy commitments required for thorough testing and deployment. As with most custom Salesforce applications, their dependency on integrations and data sharing means that each sprint involves multiple systems, which necessitates strong release management practices and robust DevOps (e.g., Copado or Gearset) tools for automating test releases and deployments.

Conclusion

Salesforce development projects have benefited from agile methodologies. In larger projects, teams have been able to deliver faster, meet goals more accurately, communicate more effectively, reassess features and scope as the business need evolves, and work closely with stakeholders who can offer useful feedback and catch mistakes before they become major. Team collaboration, customer feedback, limited risks, and increasing productivity were results of adopting Agile frameworks in development projects. Through agile principles and values, development teams were able to break down projects into short iterative 'sprints' whereby stakeholders could provide feedback in smaller, yet frequent, increments, where the number of features in each sprint were prioritized by business value, and where teams could deliver usable software faster than in waterfall projects.

While Agile can be challenging (stakeholder engagement, anyone, and who has the mental endurance for constant iteration?), these challenges are outweighed by the benefits: Agile methodologies provide structure while maintaining flexibility for an organization looking to derive as much value as possible from their Salesforce projects. Agile methodologies follow projects through their iterations, ensuring that a Salesforce implementation is as responsive to change as it is helpful to its users.

References

[1] Schwaber, Ken. *Agile Project Management with Scrum*. Microsoft Press, 2004.

[2] Cockburn, Alistair. *Agile Software Development: The Cooperative Game*. Pearson, 2006.

[3] Cohn, Mike. *Succeeding with Agile: Software Development Using Scrum*. Addison-Wesley, 2010.

[4] Dingsøyr, Torgeir, et al. *Agile Software Development: Current Research and Future Directions*. Springer, 2010.

[5] Beck, Kent, and Cynthia Andres. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 2004