commercetools
Next generation commerce

Headless Commerce Implementations

Project Management Best Practices

www.commercetools.com

Table of Contents

Executive Summary **3**

Introduction: Transform with a Plan **4**

A headless commerce implementation differs from (and is similar to) a monolithic implementation **5**

How to run a headless commerce implementation **7**

Strangler Pattern 7

Managing a Multi-Vendor Project Successfully **8**

Understanding the Role of a Project Manager 9

Responsibility Assignment Matrix Index 9

Aligning the Implementation Team Resources 10

Notes on Agile vs. Waterfall 11

Implementation of Agile Is Best for Headless 12

Initiate

Training

Execute

Monitor/Control

Close

Run

Implementation Case Studies **15**

Case Study 1 - Telco Migration
How to adopt Agile and change organizational mindset during a migration 15

Case Study 2 - Maintenance
Post-Launch Maintenance Project 19

Case Study 3 - Fashion Migration
The “Already Agile” Migration 22

Conclusion **25**

Executive Summary

Digital transformation requires more than a technical transformation to cloud-native solutions; it requires a shift in organizational mindset. Afterall, the purpose of transformation is to enable experimentation, adaptability and innovation across every discipline. This need for agility is especially critical in eCommerce and is driving a massive migration from older-generation commerce to a more modern approach.

Organizations that have moved from monolithic commerce applications, including Oracle, IBM, Magento and Salesforce, have realized that managing their migration project requires a new approach – think ecosystem vs. monolith and flexibility as opposed to rigid rules.

In this whitepaper, we define the best practices developed by certified PMs who have successfully implemented commercetools: a cloud-native, headless solution built on microservices. It includes examples from actual deployments and provides an easy-to-follow project management framework for success.

Introduction: Transform with a Plan

Digital transformation means architectural transformation, often starting with the move to cloud-based solutions which allow for rapid iteration with multiple deployments daily. To achieve this, companies need to bolster their tech teams with top talent – employees with a growth mindset and the ability to react quickly – in short, a "Yes" mentality. Solution architects and devOps engineers are especially valuable in these transforming companies. But one of the most critical areas to focus on is project management. Effective Project Managers (PM) can be a driving force in the success of a software migration project.

One type of migration getting a lot of attention in the modernizing enterprise is eCommerce. There are a few key drivers of the shift. Architecturally, twenty-plus year-old monolithic commerce platforms limit organizational maneuverability in a market that is all about speed. Financially, mountainous technical debt surrounds these older applications, including high licensing fees, maintenance fees, upgrades and outdated rev share models that cut deeply into retailers' profits. Professionally, tech teams are locked into proprietary code development that has no career value beyond their current role, lack options and become demotivated.

commercetools disrupted the eCommerce platform market by inventing and rewriting a modern commerce solution from scratch, starting with a vision of making a commerce platform like the one Amazon built: available to any organization – but at a fraction of the cost and time it would take to build it from scratch.

The commercetools API-based cloud commerce platform is based on a microservices architecture that increases overall agility and maintainability. It not only allows more experimentation, adaptability and innovation, but also allows for rapid iteration with multiple deployments daily. It enables businesses to hire based on talent and not platform lock-in.

The four pillars of the commercetools architecture – called **MACH** – are **M**icroservices, **A**PI-first, **C**loud-native and **H**eadless and let you deliver commerce functionality at **MACH** speed. Its modular approach allows companies to integrate easily with frontend and backend systems and add new channels in days. It is fundamentally different from a monolithic platform due to its modular design allowing a phased migration rather than a large-scale lift and shift. In addition, because it is headless – meaning any frontend UX can be connected via APIs – this implementation enables a best of breed approach.

For the Project Manager, this often means coordinating timing and resources from multiple vendors simultaneously. Multiple sets of Agile teams are required, but a structured project is still needed to deliver against a budget and receive predictable support from your vendors.

The realities of delivering a large-scale commerce implementation require a combined project management approach that takes advantage of the benefits of the innovative technologies and still delivers a cohesive integrated solution on time and on budget. Therefore, neither a purely Agile approach or an absolute Waterfall approach with its clearly defined deliverables are recommended. It's imperative that the project team has the flexibility to incorporate processes from both Agile and Waterfall to meet the defined deliverables.

The traditional project management (Waterfall) approach is linear, where all the stages of a process occur in an order. This approach assumes that time and cost are variables and requirements are fixed. The reason why it conflicts with Agile project management is an inflexibility that it is not meant for large projects and leaves no scope for changing the requirements once the project starts.

> ***"What we love about microservices is we can do lots of deployments. We don't have to take this huge project and take waterfall developments and run them through QA. Teams can be autonomous and deploy capabilities within ther scope of microservices, test them and get them out the door. You can do dozens and dozens of deployments on a regular basis without too much risk of changing too many things"***
>
> – Dirk Hoerig, CEO and Co-founder, commercetools

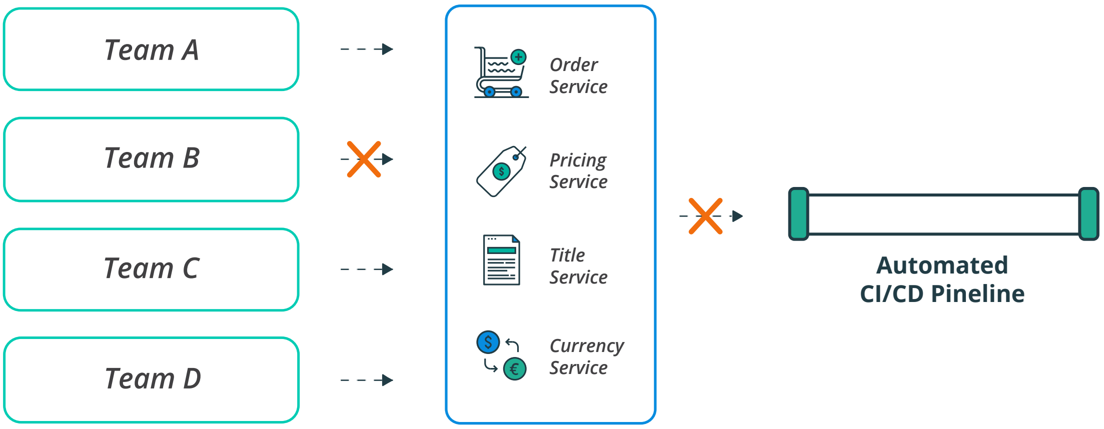
A headless commerce implementation differs from (and is similar to) a monolithic implementation

Just as no two organizations are the same, no two commerce implementations are the same. Over time – even when starting with the same software – code is added, use cases are supported and unique products and channels must be accounted for. Therefore, when starting a headless commerce migration project, it is important to understand the differences and similarities between a headless commerce implementation and a monolithic replatforming project.

What does each of the scenarios mean?
Headless = Headless separates the frontend (mobile app, CMS, etc.) and the backend (commerce) layer. Multiple teams focus on discrete areas of the implementation with agility to operate independently with limited dependencies. Project architecture oversight focuses on ensuring pieces tie together and deliver the desired customer experience.

Monolith = highly dependent teams build on each other's previous work (very Waterfall) and delays in one area delay the entire project. A single "big bang go live" after a long project brings large risk of failure and project overruns.
As eCommerce sales continue to grow, many businesses are leaving behind monolithic applications in favor of microservices, as this modern commerce architecture lets them seamlessly connect different touchpoints to the backend via APIs, giving them the flexibility to change and scale the customer-facing experience independent of the backend. With the advent of new technologies, organizations have found it difficult to continue delivering and staying competitive with their monolithic legacy applications. Monolithic architectures can't be easily updated and businesses can't move rapidly enough to keep up with customer expectations. As a result, they lose sales and their customer retention efforts in the process.

A study by Light Step[1] found that 92% of the 353 senior development stakeholders surveyed saw an increase in adoption of microservices in their organizations in the past year and they expected to see that trend continuing in the next year.



Consider a scenario where you have multiple teams working on a monolithic application. If Team B deploys bad code to a lower environment, it is going to block all the remaining teams from pushing any of their code changes until Team B pushes a code fix. Since you are dependent on a single pipeline to deploy changes from multiple teams, there is a high chance of one team blocking the other, causing delays in feature delivery to the business.

Continuing with this example, here are some ways headless and monolithic deployments differ:

Headless commerce allows brands to stay agile and constantly iterate on customer experience improvements. It also enables businesses to adjust rapidly to market changes and to implement new UX changes and add new functionality without having to transform the backend logic. A monolithic architecture, on the other hand, bonds the UX and the core creating slow and inflexible organizations. Companies get stuck because they can't keep up with the pace of change in the market.

[1] Global Microservices Trend Report, 2018, Lightstep

Headless commerce allows new functionality, third-parties or partners to come into the solution without impacting other teams' development pipelines. Common examples are enhancement to customer communications, third-party tagging and promotions, and more tactical integrations such as tax or payments.

With a monolithic architecture, to conduct UX experiments, both frontend and backend code are forced to simultaneously modify. In contrast, with headless commerce, you're free to experiment with the UX without affecting the backend operations as they are decoupled.

How to run a headless commerce implementation

One of the core tenets of modern commerce architecture is flexibility. The separation of the frontend and the backend commerce functionality (headless commerce) lets marketing "own" the experience layer and enables them to quickly alter frontend elements to run campaigns, promotions or create the most advanced "customer experience" on any frontend device (aka channel). Headless commerce gives companies the freedom to innovate when, where and how they want. From an implementation standpoint, it means the death of the "replatform" with a 12- to 18-month move to a new software solution. A headless commerce deployment uses a phased approach over time, strangling off monolithic components, and turning on new modules.

Strangler Pattern

For most headless commerce implementations, commercetools advocates for a Strangler Pattern. We see that most organizations are able to reduce risk, realize value and remove legacy application roadblocks quickly.

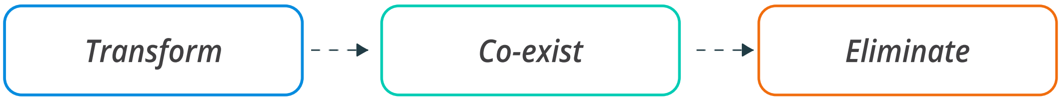
The Strangler Pattern is a popular design pattern to incrementally transform your monolithic application into microservices by replacing a particular functionality with a new service. Once the new functionality is ready, the old component is strangled, the new service is put into use and the old component is decommissioned altogether.
To implement the Strangler Pattern, you can follow three steps:

1. Transform
2. Coexist
3. Eliminate

Based on the diagram below, the steps can be summarized:

- Initially, all application traffic is routed to the legacy application.
- Once the new component is built, you can test your new functionality in parallel against the existing monolithic code.
- Both the monolith and the newly built component need to be functional for a period of time. Sometimes the transitional phase can last for an extended duration.
- When the new component has been incrementally developed and tested, you can eliminate the monolithic functionality.

Transform and Eliminate Pattern



Common blocks of functionality that are considered for the Strangler Pattern are the product catalog, which can include browse and search capabilities, the customer's My Account area or the cart and orders functionality. Within these areas, you can further breakdown the pages which are leveraging the new headless platform.

There are other approaches which can be equally successful, but existing systems may prevent you from achieving a strangler approach. Strongly consider a strangler approach and consider the possible outcomes during your planning phase.

Managing a Multi-Vendor Project Successfully

Headless commerce – as stated previously – is a "best of breed" approach, meaning that multiple vendors could be involved during a commerce platform migration. In some projects, the frontend is switched out first while in others, the frontend and backend development happen in parallel. In addition, the client may have internal resources or hire a system integrator to do the deployment.

In multi-vendor projects, it is important that all vendors understand the actual requirements set by the customer and acknowledge that a multi-vendor project is more complex due to vendor codependency. The impression of these differences in terms of understanding and insight of the project deliverables can be reduced if both the customer and vendors stay attentive.

Some of the key areas that are foundational to effectively manage multi-vendor projects are documented below.

Understanding the role of your Project Manager

- Responsibility Assignment Matrix
- Aligning on the correct team resources
- Notes on Agile vs. Waterfall

In Waterfall, the PM is responsible for successful implementation of a project through the five stages/processes of a project lifecycle:

- Initiating
- Planning
- Executing
- Monitoring and Controlling
- Closing the project

Included in these phases is identifying requirements, management of stakeholders and balancing the competing project constraints arising during the project. The project constraints include the: Scope, Quality, Schedule, Budget, Resources and Risk.
While a Project Manager in Agile works very closely with top management for strategic decision making, they still maintain the role of being the individual responsible for successful implementation of the quality defined product. There are different roles in a fully Agile environment including: Project Manager, Product Manager, Product Owner, Scrum Master and others. It is the PM that supports the entire team throughout the iterations and shields them from distractions.

It's absolutely important that you empower your Project Manager to set up a RACI framework.

Responsibility Assignment Matrix Index:

The RACI matrix is a responsibility assignment chart that maps out every task, milestone or key decision involved in completing a project and assigns which roles are Responsible for each action item, which personnel are Accountable, and, where appropriate, who needs to be Consulted or Informed. The acronym RACI stands for the four roles that stakeholders might play in any project.

- **Responsible** - The person assigned to perform the task or deliverable.
- **Accountable** - The person or role is responsible for the overall completion of the task or deliverable. They won't perform the task, but are responsible for making sure it's finalized.
- **Consulted** - This person, role or group will provide information required to complete the task.
- **Informed** - People or groups that will be kept up to date on the tasks/deliverables.

See sample RACI chart below.

Task	Project team	Client
Initiate	A	C
Plan	A	C
Execute - Development	A	I
Execute - Internal Testing	A	I
Execute - UAT	I	A
Execute - Deployment	C	A
Monitor & Control	A	A
Close	A	A

Aligning the Implementation Team Resources

The project success hinges on having an experienced Project Manager who is able to also focus on staffing the project team with the proper resources. Informed and guided by the already established RACI, the Project Manager should insist on the following as minimum requirements:

1. Business processes are documented as a part of requirement preparation by the appropriate team members
2. Subject Matter Experts based on the existing systems and business processes are assigned to the project team
3. Business Owner/Decision Makers for business process changes needed as a part of the project team are in each meeting
4. Business Owner/Decision Makers have “pre-meetings” with the Technical Team to ensure internal priorities are aligned
5. Stakeholders throughout the business from all key areas are engaged to ensure you have all of the supporting pieces needed to be successful
6. Scrums are held to ensure cross-team collaboration
7. If training is needed, it is identified at the beginning of the project and proper training resources are secured for customer team members

These items are recommended to provide a foundation for, but not a guarantee of, success:

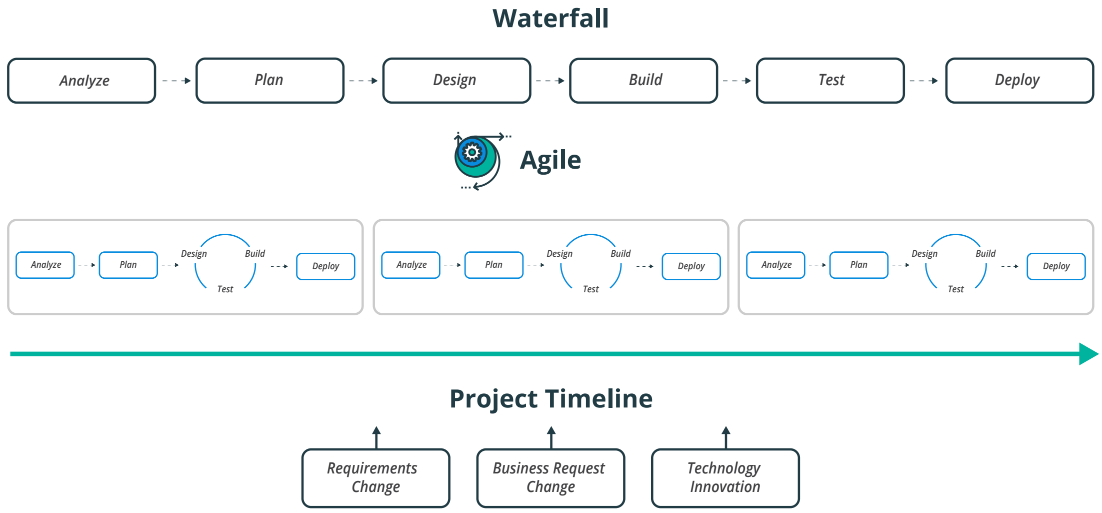
- Architecture Review Board
- A Scrum of Scrums to support the multiple internal teams as well as system integrators and other vendors involved
- Steering committee meetings held on a regular basis to provide guidance and input to the project team

Notes on Agile vs. Waterfall

Selecting the right method for managing the tasks can either make or break the success of your projects. Agile is a philosophy that emerged to address the shortcomings of the Waterfall approach. The main difference between the two practices is flexibility. Many Agilists (practitioners of Agile methods) refer to anything that is not "Agile" as "Waterfall". And many of them use it as a pejorative term. So, we get this false and destructive syllogism:

> *Waterfall = no Agile*
> *Agile = Good*
> *Waterfall = Bad*

But the reality is that each method has its merits depending on the project. Implementation projects for example, can be different from a run project. Implementation has more structure as you deploy, change the business processes, and then move past launch into a more Agile facility. Choosing Agile to start an implementation without the associated changes is fraught with risk. A Waterfall process is better for large teams and doesn't require heavy synchronization across teams due to the structure put in place. Agile works best when it's executed inside small to medium-sized teams that have established a high level of cross-integration.



Implementation of Agile Is Best for Headless

commercetools has found - over numerous implementations - that a modified Agile approach has been successful in large and small efforts.

While we won’t prescribe a specific framework and practice that make up a modified Agile process, we see a reduced Waterfall-like discovery as a key initiation step for the project.

The following illuminates how the modified Agile framework is made up of processes that are focused on improving the quality of the “code” and gather feedback from users, framed by an iterative mapping, just like pure Agile, but also borrows from the Waterfall framework, which is more structured and often inflexible. The practices below, in the Initiate, Execute, Monitor/Control sections and the following case study help to underscore this point.

The primary phases of Agile development are described below.

Initiate

Define the project at a broad level, gather stakeholders, create a project charter or project initiation documentation and decide which process model helps to focus on the business priorities.

Standard Agile Process:

- Project planning
- Product roadmap creation
- Release planning
- Sprint planning
- Daily meetings
- Sprint review and retrospective

The streamlined process is open to change and focused on continuous improvement. It is gradual and iterative. It starts with a simple design, which will be updated as the project progresses.

The work area is divided into modules and performed in sprints. Each sprint lasts a couple of weeks and consists of the following stages:

The plan:

- To separate the concept into small pieces.
- Requirements analysis and numerous meetings with stakeholders to gather detailed information.

During this phase the scope of the project is defined, project plan is developed, costs are identified, availability of resources is checked, and roles and responsibilities are defined clearly. The simplest and most effective approach to define and document project roles and responsibilities is the RACI model described above.

As outlined in the Project Case Study example one on page 15, Agile is a better approach when any of the following conditions occur:

- There is no formal documentation
- No project plan has been created
- No formal timeline has been developed or agreed to
- Project Manager role is not defined
- Structure was very "loose" – due to a small team (can be both a benefit and a curse)
- Conflict between defined deliverables and Agile work methods

Training

Account for business process changes that require training and decision making if the team is not well aligned to meet the customer expectations:

Andrew Miller had this to say about adapting a project in midstream to accommodate needed training: ***"To mitigate this issue, additional resources can be added to the team, but this only partially helps."*** A better approach per Miller is to conduct and implement the training plan to get the required skill set. ***"It eliminates the need for added development staff and bolsters the development team's skillset."***

Execute

In the execution phase, project deliverables are developed, tested and deployed. This can be broken down into iterative cycles and is where a Strangler Pattern would be implemented to start to realize the value/benefits of the headless solution faster. There are short, specific time goals for the delivery of the sprint and regular and monitored accountability called Timeboxing.

- **Design**
- **Build**
- **Test**
  - Frontend testing is done by the external vendor
  - Backend testing is done by the development team
  - Functional testing is done by the project team
  - Final testing and sign off is done by the client

- **Deploy**
  - Deployments are generally performed by the client team supported by the Systems Integration Team
  - Client needs to provide the right resources (DevOps) to manage the deployment for long term

Monitor/Control

The Monitor/Control phase happens concurrently with the Execute phase. The PM measures project progression and performance and reports on status. Monitoring is done 24/7 using dashboards and automated alerts. Off-hours support is done by the operations team. Scrum methodology is a best practice in Agile to focus on business opportunities.

According to Thorsten Bayer, Scrum was a best fit for a commercetools project which was ongoing. “No standard reporting is done during the sprints. The client is notified at the start and end of each sprint and the client has access to JIRA and monitors that system for updates,” said Bayer.

Close

In a proper Close, the team is released and a final recap meeting is conducted with all major stakeholders. Lessons learned will be thoroughly reviewed and all project documentation should be archived. Both Monitor and Control and Close will need to be performed by both the client and commercetools unless one PM is retained throughout the entire lifecycle.

Run

Convert the team to an Agile team responsible for ongoing monitoring, and iteration of the solution. A good practice for the project management team and the vendors is to hold regular meetings on the project so that inputs can be obtained and every key team member is on the same page with regards to the project deliverables.

In every phase of the project, stakeholders throughout the business should be kept informed to ensure you have all of the supporting pieces needed to be successful.

Implementation Case Studies

In the following pages, we take a look at three very different types of implementations and provide best practice call-outs along the way. Find the project that most closely matches your situation and look for ways to continually improve your own project management practices.

Case Study 1 - Telco Migration

How to adopt Agile and change organizational mindset during a migration

Migration from Oracle Monolith

One of the world's largest telecommunications companies had legacy systems with many legacy integrations. They wanted to consolidate into one modern, API-forward solution that would encompass all their pricing and promotional needs. The mindset of working with legacy commerce platforms had prevented the IT organization from truly adopting Agile methodology as they were in a continual break/fix and maintenance mode. Each system was its own monolith preventing a sprint-based approach. It is possible to do both: migrate to a MACH commerce architecture while adopting Agile. ***Your job as Project Manager is to identify project danger signs at the outset and gain agreement on a clearly defined framework.***

Ensure success with a strong kickoff

⚠ *Legacy Project Warning Signs: (Kickoff and Planning)*

- legacy systems (could also be brands & channels)
- Inexperience with Agile methodology
- Ill-defined project requirements
- Project management not considered during vendor selection phase
- Data models of legacy systems not well defined
- Stories not written for functionality or for the developers

✓ *Framework for Success: (Kickoff and Planning)*

For success during kickoff and planning, commercetools recommends following Standard Agile Process:

- Create project charter
- Identify stakeholders
- Select project management tools
- Project planning
- Product roadmap creation
- Release planning
- Sprint planning
- Daily meetings
- Sprint review and retrospective

Staff the project correctly from the beginning

An area to watch out for in a project heavily burdened by legacy systems and legacy mindsets is resourcing and staffing. As Project Manager, it will be your job to closely monitor the resources assigned, allocated time, talent and communications protocols.

⚠ *Legacy Project Warning Signs: (Staffing)*

- Different expectations of onsite vs. remote team support
- Assumption of “on the fly” deliverables and schedules
- Misalignment on role of in-house vs. external teams
- Under-staffing, lack of Project Manager
- Team misalignment to customer expectations

✓ *Framework for Success: (Staffing Plan)*

- Create project plan
- Develop resource plan
- Define performance measures
- Communicate roles and responsibilities to team and stakeholders
- Identify risks and create contingency plans

In addition to planning resources, team roles also need to be clearly documented and agreed upon:

✓ *Framework for Success: (Team Roles & Comms)*

Standard Process:

- Determine team members
- Assign a product owner
- Define roles and tasks assigned to those roles
- Create contact information
- Communicate to team

Plan for learning curves when introducing new tools

Legacy software often means legacy tools. Access to the legacy platform and design of the system has quite likely for years meant the commerce team was working in proprietary and outdated tools. In a migration project, be aware that new tools may need to be created to extract data from legacy systems and prepare it for import into the new system. If this is the case, time needs to be allotted in the overall project plan.

⚠ *Legacy Project Warning Signs: (Tools)*

- Unfamiliar with project tool sets like GitHub and JIRA
- Access rights to tools limited; need to be granted
- New tools onboarding and training across team not part of project plan

✓ *Framework for Success: (Project Tools)*

- Determine current software and functionality
- Identify current software SMEs
- Define functionality of new software
- Plan for data migration

Be mindful of team hand-offs during development, testing and deployment

Depending on the organization, there may be one or several parties involved in development and test phases of the platform migration. Whether clearly outlined upfront or modified during the implementation, the PM needs to ensure proper hand-off throughout this critical phase. In addition, reports must be generated on both sides and presented to key stakeholders via the PM.

⚠ *Legacy Project Warning Signs: (Development, Test and Deployment)*

- Multiple parties handling different phases of the project
- Gap in PM resources (PM required at client)

✓ *Framework for Success: (Development, Test & Deployment)*

- Have a kickoff meeting with development team
- Monitor project performance
- Perform risk management tasks
- Conduct status meetings
- Report to status to team and stakeholders
- Update project schedule
- Modify project plan
- Execute change control process if necessary

Close strong

Everyone likes being part of a win. Growth minded team members also like learning from project resets along the way. A strong close, with measurement of key deliverables, provides a sense of completion, acknowledgement of the move to organizational mindset change, and of course, a reason to celebrate. For a project like this with legacy software, legacy tools and the need to look differently at platform migration, our PM team recommends the following approach.

✓ *Framework for Success: (Project Close)*

- Verify all deliverables
- Hand-off the tasks to the team handling the new/updated systems
- Conduct lessons learned meeting
- Document open items or follow-up tasks
- Organize all project documentation
- Communication project success to team and stakeholders

Case Study 2 - Maintenance

Post-Launch Maintenance Project

Using Scrum to deliver business priorities

In a project that has already launched and is in Maintenance/Enhancement mode, the PM team speaks with the customer to get new requirements. This leads to the creation of a JIRA ticket and hand-off of the request to the development team.

✓ *Framework for Success: (Standard Agile Process)*

- Project planning
- Product roadmap creation
- Release planning
- Sprint planning
- Daily meetings
- Sprint review and retrospective

Since the project is in maintenance mode, the focus is on accurate requirements gathering, handoff and communication. The team works through a dedicated customer contact. The contact works with the internal resources and business teams (i.e. system users) within the client company to gather new requirements. In this specific example, the customer does not have an internal development staff so all development work is done by an external vendor. This methodology could work equally well if the client has its own internal development staff.

✓ *Framework for Success: (Maintenance)*

Standard Process

- Create project charter
- Identify stakeholders
- Select project management tools

For this maintenance mode project, Scrum methodology is currently used. Prior to this, the team used a Waterfall methodology. The switch to Scrum helps to focus on the business priorities. Using Scrum, the customer is aligned with the timing of sprints and works to determine the requirements of the development ***work to be done for future sprints.***

During the planning phase, the customer contact works with the internal clients to plan upcoming development requests. The external vendors perform the frontend development and another vendor does the backend development.

✓ *Framework for Success: (Using Scrum)*

- Create project plan
- Develop resource plan
- Define performance measures
- Communicate roles and responsibilities to team and stakeholders
- Identify risks and create contingency plans

As described above, this project has a dedicated client contact, a project lead from an external vendor and a second external vendor. If in-house developers are used, there would be a dedicated contact within that development team. From commercetools, the team consists of JAVA developers. In this case, there is no functional owner and the software is externally hosted by commercetools.

✓ *Framework for Success: (Team Structure)*

- Determine team members
- Define roles and tasks assigned to those roles
- Create contact information
- Communicate to team
- Determine product owner
- Define tasks
- Communicate to team

One of the major considerations of any commerce migration project – whether it's a first-time implementation or a maintenance project – is the use of legacy tools, software and the skills of the development team. In this case study, the customer had an ERP system which was out of date when the project started. commercetools had to import the data, and then the client upgraded the ERP system to the latest version. Now commercetools exports the data back to the client system. Data is exchanged via CSV files through SFTP via cron jobs.

✓ *Framework for Success: (Software Tools and Capabilities)*

- Determine current software and functionality
- Identify current software SMEs
- Define functionality of new software
- Plan for data migration

CSV With the system currently up and running, monitoring is done 24/7 by commercetools using dashboards and automated alerts.

No standard reporting is done during the sprints. The client is notified at the start and end of each sprint. The client has access to JIRA and monitors that system for updates.

✓ *Framework for Success: (Project Monitoring and Reporting)*

- Monitor project performance
- Perform risk management tasks
- Conduct status meetings
- Report to status to team and stakeholders
- Update project schedule
- Modify project plan
- Execute change control process if necessary

This project is on-going in an enhancement mode. The hand-off to the customer is ongoing as each sprint is completed. An official project "close" moved this project into its known and manageable phase.

✓ *Framework for Success: (Project Close)*

- Verify all deliverables
- Hand-off the tasks to the team handling the new/updated systems
- Conduct lessons learned meeting
- Document open items or follow-up tasks
- Organize all project documentation
- Communication project success to team and stakeholders

Use of the commercetools headless commerce platform allows quicker determination about whether functionality exists. With the previous software, the client had to read through the software documentation to determine if the feature they needed/wanted was already part of the software. With commercetools, if the functionality does not already exist it can often be built. commercetools is a set of tools which allows new features/functionality to be built. The previous software – SAP Hybris – either had the feature/functionality or it didn't. There was not the ability to create something new "on-the-fly".

Case Study 3 - Fashion Migration

The “Already Agile” Migration

Mature Agile team works in sprints and storyboarding

An American-based fashion retailer was running Oracle (ATG) commerce. After the decision was made to migrate to commercetools, a multi-phase project was kicked off using both commercetools and internal IT development resources. The client was already using an Agile methodology and had teams set up to support this. Two-week sprints were used along with story boarding. Development and QA was done during each sprint. The team did extensive requirements gathering at the start of the project.

✓ *Framework for Success: (Project Planning)*

Use Standard Agile Process

- Project planning
- Product roadmap creation
- Release planning
- Sprint planning
- Daily meetings
- Sprint review and retrospective

Requirements were driven by making sure the commercetools software provided parity with the existing software which was being replaced.

A formal kickoff meeting with the stakeholders was conducted along with a separate kickoff with the project team. Two months prior to starting development, the BAs began gathering requirements.

✓ *Framework for Success: (Project Kickoff)*

- Create project charter
- Identify stakeholders
- Select project management tools

In this project, a strong team of resources was assigned to ensure project success including, eCommerce engineers, BAs and client business owners. All resources were secured prior to project kick-off. The team was well-suited to the project. Team members moved to different pods during the development. Flexibility among the team was beneficial.

✓ *Framework for Success: (Project Team and Resources)*

- Create project plan
- Develop resource plan
- Define performance measures
- Communicate roles and responsibilities to team and stakeholders
- Identify risks and create contingency plans
- Determine team members
- Define roles and tasks assigned to those roles
- Create contact information
- Communicate to team

In addition to the team members mentioned above, this project also included a product owner who was part of the client leadership team. Furthermore, each area had a functional owner. This allowed the developers to focus on coding and creating solutions. This approach is strongly recommended in all implementation projects.

✓ *Framework for Success: (Project Owner)*

- Determine product owner
- Define tasks
- Communicate to team

As far as tools and legacy software, this project had the benefit of more modern tools: a sign that there will be limited impact with learning curves and onboarding new software. The one area that this project included, however, was a move of the environment from on prem to the cloud to allow maximum flexibility. Project members with institutional knowledge of the previous system provided valuable insight, which allowed easy data transfer between the old and new systems.

✓ *Framework for Success: (Software and Tools)*

- Determine current software and functionality
- Identify current software SMEs
- Define functionality of new software
- Plan for data migration

Development during this project was a 50/50 split between the project team and external vendors. In-house resources were used for QA and contractors were used for eCommerce testing. Dev ops did the deployment.

Framework for Success: (Establish Team Roles & Responsibilities)

- Kickoff meeting with development team
- Manage resources

Reporting was done during monthly senior stakeholder meetings and in monthly steering committee meetings. Weekly status was rolled into existing meetings, which the client already had.

Framework for Success: (Project Reporting)

- Monitor project performance
- Perform risk management tasks
- Conduct status meetings
- Report to status to team and stakeholders
- Update project schedule
- Modify project plan
- Execute change control process if necessary

Once the migration was complete on this project, it followed standard best practices for a successful closure. Success was measured by the amount of conversion being completed during the pilot and roll-out (speed, sales volume and website issues are rolled into the conversion measurement). The tasks listed below were performed during the close-down phase.

Framework for Success: (Project Close)

- Verify all deliverables
- Hand-off the tasks to the team handling the new/updated systems
- Conduct lessons learned meeting
- Document open items or follow-up tasks
- Organize all project documentation
- Communication project success to team and stakeholders

Conclusion

Being good at handling projects is generally a matter of following project management best practices which are derived from project management methodologies, international standards, industry conventions and the organization's own guidelines from past projects.

The following are best practices for effective project management implemented by commercetools teams:

- Ensure that all stakeholders understand the requirements
- Develop and formalize project management roles
- Develop leadership capabilities along with technical capabilities
- Conduct regular Client Status meetings

Both Waterfall and Agile project management styles have professional skill sets that require study, practice and development over time. Before starting a project, it is always worthwhile to clearly define the process, procedures and roles of team members in the early stages of the project to avoid any conflict or miscommunications. Agile will not be right for every circumstance as it supports changes at any step of the project that may change the requirements and final look of the work product at any phase of the project. Even if an organization chooses to make the change, shifting to Agile is a great investment in time, effort and money. In predictive projects, we plan and budget the full project at the beginning, so with them the traditional or Waterfall approach goes well. In Agile projects, we make incremental decisions as the scope grows iteratively.

For further assistance planning a commercetools implementation or understanding any of the concepts put forth in this guide, please contact us.

Project Manager Bios

Andrew Miller

Co-Founder & Managing Director RIBA-Aydept

Andrew formed Aydept in 2015 to bring disciplined project management to mid-sized retailers. Before Aydept, Andrew was President of Signal Creek Technology where he managed enterprise projects for government agencies and large commercial firms in the public and private sectors. Andrew wanted to return to retail because of his love gained earlier in his career as IT Director of Vail Resorts Retail.

Mike Charow

Senior Project Manager

Mike has over 20 years of experience in project management. After working for large companies like Ford, Delphi and General Dynamics, he brings to retail a profound understanding of supply chains processes. Mike is accomplished managing the nitty gritty of projects such as scheduling, allocating resources, budgeting and change.

Raymond Ussery

Consulting Practice Manager

Raymond has led and consulted on technology, digital marketing and commerce projects for more than 12 years. He leads and adapts the development process as needed during the software development life cycle so that deadlines are met. Guided by his experience from consumer packaged goods, and automotive and manufacturing projects, he views program and operations best practices.

Jake Mays

Co-founder, Aries Solutions

Jake Mays is the co-founder of Aries Solutions; a North American company implementing outstanding and innovative commerce solutions for customers looking to improve their cloud-based eCommerce. After graduating from Ohio State University with a specialization in Operations Management, Jake began his career at Deloitte's Strategy & Operations Consulting practice. His other previous titles include CFO at Systems Integration Specialists and Program Manager in the retail sector with a focus on eCommerce and Omni-channel transformations.

Thorsten Bayer

Product Owner Solution

Thorsten began his career as a software developer, working 10 years with a major publishing house and went on to support several companies as a Senior PHP Developer in a freelance role. Thorsten joined commercetools in August 2016 as Technical Support / Project Coordinator. Within a year he moved into his Product Owner Solution role, which allowed him to pivot his career path towards project management and a more customer facing capacity. His focus is on enabling and supporting our customers in having a successful ecommerce solution, allowing them to maximize the most out of our commercetools platform. Thorsten's current role allows him to still be close to the tech side of software development, which is of particular interest to him.

About RIBA-Aydept

The power of project management and robust data integration. Our team of experts is focused on providing the best project management and data integration in the industry – the two key areas where IT projects often get into trouble.

When RIBA-AYDEPT manages IT projects, it never loses sight of the client's value aspiration. We adapt to your preferred PM methodology whether agile, waterfall, or blended. Our focus is on timeliness, reducing costs, and achieving the highest quality.

About commercetools

commercetools is the world's leading platform for next-generation B2C and B2B commerce. To break the market out of being restrained by legacy suites, commercetools invented a headless, API-first, multi-tenant SaaS commerce platform that is cloud native and uses flexible microservices. This enables customers to deliver the best commerce experiences across all touchpoints.

Founded in Germany in 2006, commercetools has worldwide offices spanning the US, Europe and Asia Pacific, with a customer base of Fortune Global 500 companies across industries.

Contact Us

Europe - HQ
commercetools GmbH
Adams-Lehmann-Str. 44
80797 Munich, Germany
Tel. +49 (89) 99 82 996-0
info@commercetools.com

Americas
commercetools, Inc.
324 Blackwell, Suite 120
Durham, NC 27701
Tel. +1 212-220-3809
mail@commercetools.com

www.commercetools.com
Munich - Berlin - Jena - Amsterdam - London - Durham NC - Singapore - Melbourne