

Chapter 1

Stages of an Implementation Overview

Welcome to the challenging world of implementing business application software. It's not for the faint of heart, and there aren't many guides that show you how to get there. That's the goal of this book—to provide an actionable guide to implement Microsoft's Dynamics systems successfully.

This book is designed for anyone involved in an Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM) implementation. The content is focused on Microsoft Dynamics, as that is what I have spent my career working on. This book can and should be read by anyone in an executive-level leadership position—the chief executive officer (CEO), chief information officer (CIO), and chief financial officer (CFO)—if you are considering implementing a new business application. Implementing a new business application is something that you will only do a few times during your working career, and there simply is a lack of reputable guides to support you.

AFTER YOU COMPLETE THIS CHAPTER, YOU WILL BE ABLE TO DO THE FOLLOWING:

- ◆ Define the Client Journey to complete a Microsoft Dynamics implementation
- ◆ Differentiate between three common implementation methodologies
- ◆ Recognize triple constraints and how they impact an implementation

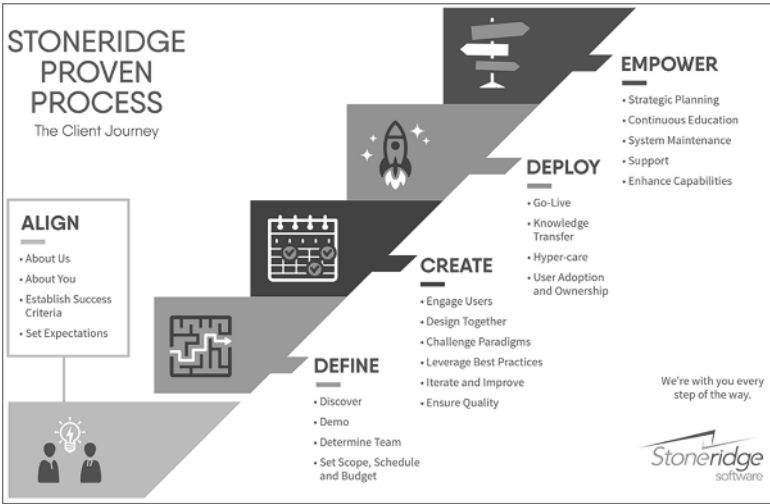
What Is Microsoft Dynamics?

Microsoft Dynamics is a set of business application software built by Microsoft and focused on Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM). Microsoft sells Dynamics in several different flavors that are ever-changing. The cloud-based solution is called *Microsoft Dynamics 365*, and it includes a very broad set of functionalities across Sales, Customer Service, Human Resources, Finance, Supply Chain, Commerce, Project Service, Marketing, and Customer Data Management. You can see additional information about the product on Microsoft's website at dynamics.microsoft.com.

The Client Journey

At Stoneridge Software, we developed what we call the *Client Journey*, which provides high-level stages to complete the implementation of Microsoft Dynamics. This takes you all the way from initial internal conversations, to meetings with potential vendors, to the enhancements you want to do after you’ve gone live on the software (see Figure 1.1).

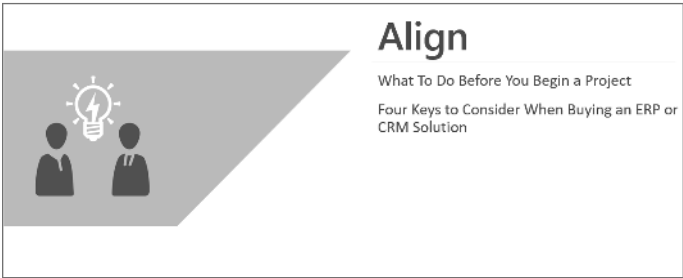
FIGURE 1.1
The Client Journey



We’ll break down the Client Journey in the chapters of this book. It all starts with gaining internal alignment and finding alignment with your chosen implementation partner. If you can do that, you’re going to be ahead of most companies out there. The *Align stage* is focused on what you want to accomplish with the project.

Two chapters of the book are focused on the Align stage, starting with what to do before you begin a project and then a session on four keys to consider when buying an ERP or CRM solution. The goal of these two chapters is to get your team internally aligned and then provide you with an idea of what to be looking for in the best product for your company (see Figure 1.2).

FIGURE 1.2
The Align stage in the Client Journey



Once you have alignment internally and know the right type of implementation partner for you, you start on the *Define stage*. This is when you answer the (literally) million-dollar questions like what software are we going to buy? Who's going to be on the project? What does our budget look like? The Define stage takes you through the sales process and the pre-kickoff process of the implementation (see Figure 1.3).

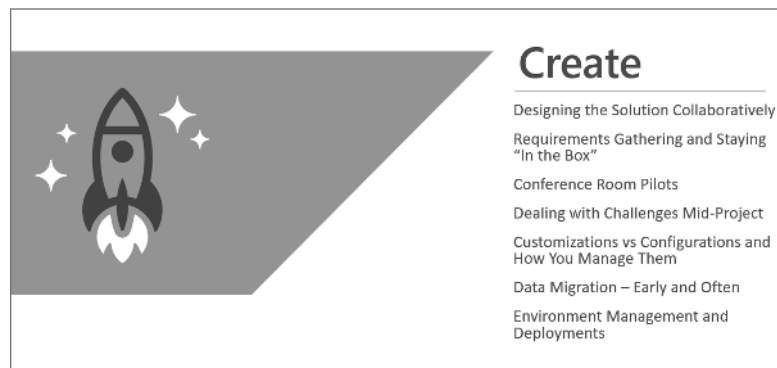
FIGURE 1.3
The Define stage in the
Client Journey



Seven chapters in the book focus on the Define stage, starting with the sales process, where we cover discovery, demos, and contract negotiations; then we move into the steps that you need to start the project out right. From there, we dive into the project preparations starting with how you organize your team, to moving to the cadence with which you run the project, and then to change management. We outline how to break down the work by process area, how to evaluate third-party solutions, and finish up with the kickoff meeting.

The *Create stage* is where most of the action happens (see Figure 1.4). This begins after the kickoff meeting, and it takes you through the point where code is complete and you're ready to test and get the software out the door.

FIGURE 1.4
The Create stage in the
Client Journey



Seven chapters in the book focus on the Create stage. There is a lot to get done during this part of the implementation. I outline how to sort through the processes to generate requirements

and how to make those changes to the software while driving toward a cut-off point that allows you to get ready to go live.

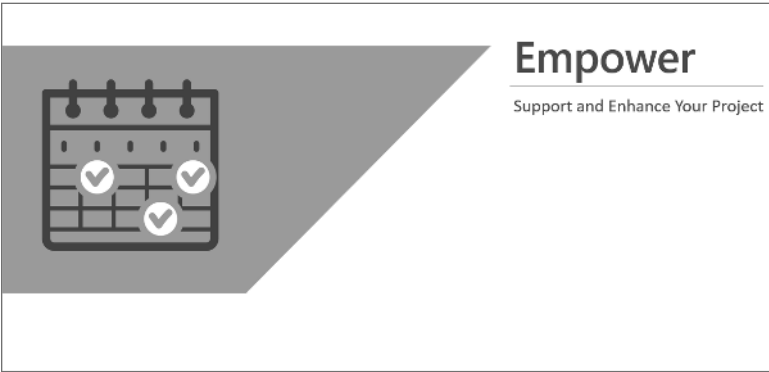
The *Deploy stage* is when you take the software and process from the project team and roll it out to everyone (see Figure 1.5). It takes you through the testing, training, and go-live phases. The chapters in the Deploy stage include testing, training, going live, and the post go-live activity we call “hypercare.”

FIGURE 1.5
The Deploy stage in the Client Journey



The *Empower stage* is where you find your rhythm for future advances to the solution and look for opportunities to continue receiving additional value from your solution. There is just one chapter in the book on this stage, and we discuss the various paths that companies pursue and the activities with which you want to be consistent after a successful go-live (see Figure 1.6).

FIGURE 1.6
The Empower stage in the Client Journey



Implementation Methodologies

Three common implementation methodologies are used across all different types of IT projects: Waterfall, Agile, and Scrum. Scrum is a variant of Agile, so we'll talk about those two collectively. The methodologies are discussed to provide general knowledge and background information, as they are commonly known in the IT world. What I recommend is a hybrid of these implementation methodologies, which we'll cover as we get further into the book.

Waterfall and Sure Step

Let's start with the oldest and most common methodology used in business application implementations—the *Waterfall methodology*. Born in the late 1950s, this was the first known implementation methodology and became more commonly known and implemented after 1985. The Waterfall methodology is a step-by-step methodology that tries to keep all of the activities in linear order from start to finish. You start with the Analysis phase, finish all of its elements, and then you move on to the Design phase. You then do all of the Design phase work and then move forward to the Development phase, and so on. In the Waterfall methodology, you are not supposed to work ahead and take on a future task outside of your current phase. The Waterfall methodology is a good fit for an ERP implementation because, in many ways, you do need to work through things in chronological order. You need to define your account structure, customer structure, your products, and other master data that you want to be able to set up early on in a project.

In the decade of the 2000s, Microsoft suggested that all Dynamics implementations follow its *Sure Step Methodology*, which was a Waterfall-based solution (see Figure 1.7).

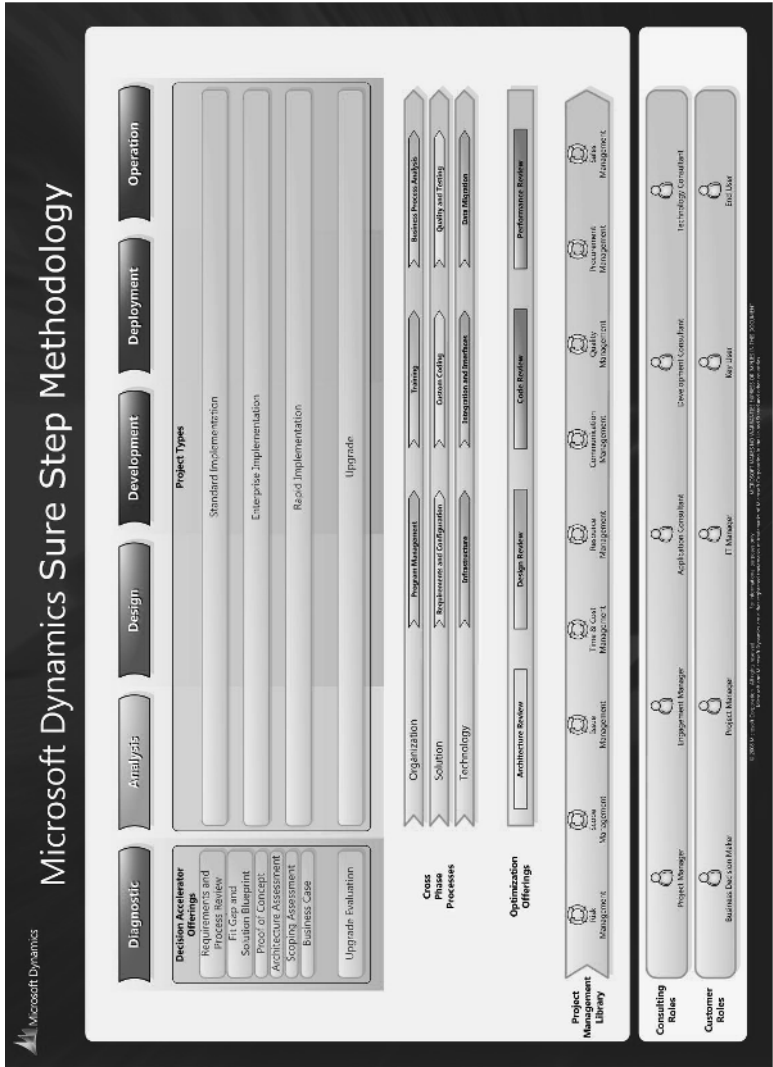
As you can see, it starts with the Diagnostic phase and then moves to the Analysis phase, the Design phase, the Development phase, the Deployment phase, and finally the Operation phase.

In my days at Microsoft, I learned and used this methodology quite a bit, and it contributed to many of the documents in the Optimization section, as my team was focused on project governance, health checks, and workshops. Partners and Microsoft Consulting Services were directed (but not mandated) to use this methodology, so you saw a lot of adoption, especially in the 2007–2011 time frame.

During that time, a shift started to occur. It began with the Microsoft Consulting organization, as they realized that they needed an implementation that was more flexible than the rigid Waterfall approach. Microsoft started implementing some of the Agile concepts into the classic Waterfall approach, and that is when I realized that there was a better way. Microsoft stopped actively recommending Sure Step and stopped any further development on the methodology. The latest version dates back to 2010, so you can certainly review that information with the knowledge that the content is quite dated.

Customers and partners can find additional resources on Microsoft's CustomerSource site (mbs.microsoft.com/customersource/northamerica). Tons of document templates are available that can prove valuable on projects. You can certainly mine for treasure in the database without following it step-by-step.

FIGURE 1.7
Microsoft Dynamics
Sure Step Methodology



Agile and Scrum

The *Agile methodology* was born from a meeting in 2001 between 17 IT professionals who felt that there was a better way than the stagnant Waterfall methodology. They wrote the *Agile Manifesto*, which includes 12 principles for developing software efficiently and effectively. The manifesto has four parts (see Figure 1.8):

- ◆ Individuals and interactions over processes and tools
- ◆ Working software over comprehensive documentation
- ◆ Customer collaboration over contract negotiation
- ◆ Responding to change over following a plan

FIGURE 1.8
Four parts of the
Agile Manifesto

Individuals & Interactions	Working Software	Customer Collaboration	Responding to Change
• Communication is a key reason for project's success • People build software products • Teams self-organize • Motivate individuals and foster interactions among team members, customers, stakeholders. High-context interactions • Focus on people – they are the ultimate source of energy	• Primary goal is to create and deliver value. Satisfy/delight customer • Deliver frequently to give business advantage to the customer • Create documentation that adds value • Customer value/priority focus	• Be flexible and cooperative on customer's Dynamics needs • Work with the customer to deliver the intent of the contract • Closely collaborate & follow customer's product vision • Have flexible contracting models. Maintain relationships	• Change is a reality, reality is the worst enemy of a plan • Changes in customer's business environment impact our plans • Embrace change by adaptive planning. Reflect & improve

The Agile methodology is the opposite of the Waterfall methodology. It's designed to get software designed, developed, tested, and released quickly. This is great for web development or incremental development on top of a core platform, but it doesn't perfectly fit an ERP or CRM implementation because some components need to be done in a certain order.

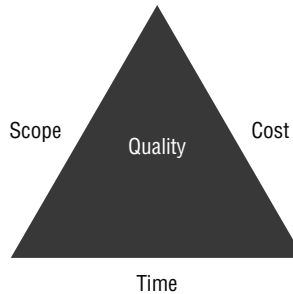
The *Scrum methodology* takes the Agile Manifesto and puts it into action. Many great concepts are included in Scrum that help any type of implementation. We'll get into greater detail in Chapter 6, "Sprints and Tools Needed to Run Your Project," but a *sprint* is a Scrum tenet that breaks activity into a two- or four-week cycle that you use to plan out your work. This allows you to manage a product backlog, which is everything you need to do on a project. From this product backlog, you assign activities to each sprint and that becomes your *sprint backlog*. The backlog provides a good idea of how much should be accomplished during a two-week period, and it creates urgency and accountability on a regular interval instead of everyone staying up all night for two weeks straight to get ready for a go-live.

I really like how sprint planning provides the ability to break the project into pieces that can be understood and managed. When you are estimating software development time, you never want someone to say 800 hours because this means that they have no idea how long it will take. If you say "there are several different components of the project, some I estimate at 8 hours, others I estimate at 40 hours, and collectively it comes to 800 hours," I will believe you because you've taken the time to break it into great detail. In an ERP or CRM implementation that is going to take a year, you have to get a better breakdown of the work to know if your projected timeline is anywhere close to being accurate.

Triple Constraints

One final concept that I want to establish early is this idea of *triple constraints*. To understand this concept, you need to think about a triangle where the three sides are cost, scope, and time. Quality is in the middle, and it is impacted if you shortcut any of the other sides (see Figure 1.9).

FIGURE 1.9
Triple constraints concept



The way the triple constraints concept works is that you want to find an equal balance between scope, cost, and time. At that equilibrium point, the project quality will be good. If you decide to add to the scope of the project, you have to add cost and time to the project as well to maintain quality. If you decide to decrease time (you want the project done faster), then you have to decrease scope (and cost) to achieve your objective without compromising quality. If one side of the triangle expands, at least one of the other two sides must also expand, but not necessarily both. Another way to look at it is once two sides of the triangle are set, the other side is locked in. Once you have settled on scope and budget, your timeline is something that could be calculated as seen in Figure 1.10.

FIGURE 1.10
Triple constraints with decreased cost

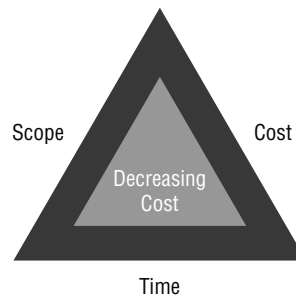
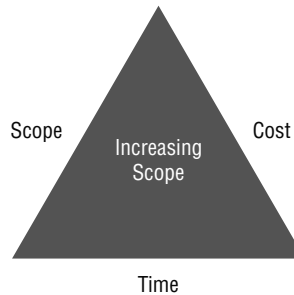


Figure 1.10 shows how the triangle starts in equilibrium. You have a reasonable scope for budget (cost) and time to implement the project. Hopefully, all projects start that way. In Figure 1.10, you decrease cost, and, in that case, you have to decrease time (because consultants will continually bill during the time) and then you're going to sacrifice either scope or quality. If you decrease scope, you can preserve quality, but the amount of benefit you get from the project

is less. If you maintain scope, you are basically saying that you're not going to test the solution thoroughly, or you're not going to train people and you're going to hope for the best once it goes live. You are increasing the odds that the project would result in a disaster.

Figure 1.11 shows a larger triangle. This is the scenario where you increase scope and therefore increase time and cost. Again, if you try to increase scope without increasing time and cost, you're diminishing training and testing and increasing the odds that the go-live could be a disaster.

FIGURE 1.11
Triple constraints with
increased scope



The Bottom Line

Organizing your project in sprints One of the key elements of the Agile and Scrum implementation methodologies is the concept of using a sprint to create interim deadlines within a project to make sure that you can deliver the full project on time.

Master It What is the first thing you do during a new sprint?

1. Run a retrospective
2. Do a planning session
3. Do a review session
4. Outline the following sprints

Waterfall methodology The Waterfall methodology is the classic way to execute on an ERP or CRM implementation, but I recommend that you use a hybrid of the Waterfall and Agile methodologies for your project.

Master It What is a part of the Waterfall methodology?

1. The implementation is done in a step-by-step process
2. Sprints are used
3. New requirements are placed in the backlog
4. The Create phase

Sure Step Microsoft put together a methodology in the early 2000s called “Sure Step,” which was widely adopted for Dynamics implementations during the 2000–2015 period.

Master It Microsoft Sure Step follows which methodology?

1. Agile
2. Scrum
3. Client Journey
4. Waterfall

Challenging projects ERP and CRM implementations are challenging projects that often go over time and over budget. The goal of this book is to prepare you better for a project so that you can complete it on time and within the budget.

Master It What percentage of implementation projects go over budget?

1. 100 percent
2. 60 percent
3. 40 percent
4. 20 percent

Triple constraints The triple constraints diagram is a triangle that represents a common concept in project management which shows how a change in one of the sides of the triangle affects the other two sides.

Master It Which of the following is not part of the triple constraints of project management?

1. Cost
2. Scope
3. Sprint
4. Time