Contents

[Introduction to the Avanade Delivery Framework (ADF) 2](#_Toc62044819)

[Value of ADF to our Clients 2](#_Toc62044820)

[Key benefits of ADF 2](#_Toc62044821)

[ADF supports multiple Delivery Approaches 2](#_Toc62044822)

[Agile Delivery 2](#_Toc62044823)

[Milestones 4](#_Toc62044824)

[Risk vs. Value 4](#_Toc62044825)

[Traditional Delivery 5](#_Toc62044826)

[ADF is a *Structured* Approach 6](#_Toc62044827)

[Level 1 - Disciplines 6](#_Toc62044828)

[Level 2 – Activities and Work Products 7](#_Toc62044829)

[Level 3 – Samples/Templates, Guidance, Tool/Asset Recommendations 7](#_Toc62044830)

[ADF Disciplines 7](#_Toc62044831)

[Delivery Leadership 8](#_Toc62044832)

[Requirements Modeling 8](#_Toc62044833)

[Design Modeling 8](#_Toc62044834)

[Development 9](#_Toc62044835)

[Testing 9](#_Toc62044836)

[Release Readiness 9](#_Toc62044837)

[Environment and Configuration Management 10](#_Toc62044838)

[Secure Software Development Lifecycle (SDLC) 10](#_Toc62044839)

[Integration with other Avanade Assets 10](#_Toc62044840)

# Introduction to the Avanade Delivery Framework (ADF)

The Avanade Delivery Framework (ADF) provides the delivery methodology through which Avanade teams deliver our full range of solutions to our clients. ADF aggregates content across the entire delivery lifecycle from initial client conversations through to the transition to a managed service, and it serves as a vehicle for continuous improvement, evolving based on best practices from leading-edge implementation experience and ongoing research and development. ADF is underpinned by the Scrum Framework, an industry standard, optimized by Avanade to cater for our client engagement models and scaled to exploit our global network of Advanced Technology Centers.

# Value of ADF to our Clients

Avanade understands that our clients need us to deliver solutions quickly while staying responsive to change. On the other hand, clients also need us to deliver compelling solutions with globally distributed teams and predictable delivery standards – all of which are strengths of ADF’s value-driven, risk-based approach.

ADF bridges the gap between the need for speed and the necessity of quality. It leverages many of the principles and best practices of the Agile philosophy, while retaining the structure and project management oversight required for success.

# Key benefits of ADF

* Supports modern engineering approaches and behaviors
* Drives reliability and predictability across the entire project lifecycle, enabling speed to market
* Drives discipline and rigor, and enables accurate and thorough work completion, which reduces cost and risk to the project
* Fosters project standardization and customization to facilitate flexibility and repeatability
* Enhances cross-group coordination and skill set utilization and supports multi-location project delivery environments
* Promotes communication and collaboration, which in turn promotes the sharing of ideas, innovation, and development of reusable assets
* Reinforces quality

# ADF supports multiple Delivery Approaches

ADF is primarily geared towards Agile delivery, which is reflected in the Agile practices, principles, and terminology that are integral to the methodology; however, it also supports traditional iterative or waterfall delivery.

## Agile Delivery

Avanade Agile delivery is based on the Scrum framework as described in the Scrum/Nexus Framework. The Scrum framework structures software development in cycles of work called Sprints, time boxes of work which are typically 1-4 weeks in length and take place one after the other without pause. The Sprints end on a specific date whether the work has been completed or not and are never extended.

As shown in Figure 1 below, at the beginning of each Sprint, cross-functional Developers in the Scrum Team select items from an ordered list of Product Backlog Items (PBIs) and forecast what they can complete by the end of the Sprint. Each workday, the Developers gather briefly to inspect their progress, and adjust the next steps needed to complete the work remaining. At the end of the Sprint, the Scrum Team reviews the Increment with Stakeholders, and demonstrates what value was delivered during the Sprint. The Scrum Team obtains feedback that can be incorporated in future Sprints. Prior to the beginning of the next Sprint, the Scrum Team discusses what went well and what didn’t and creates a plan for improvements to be enacted during the next Sprint.

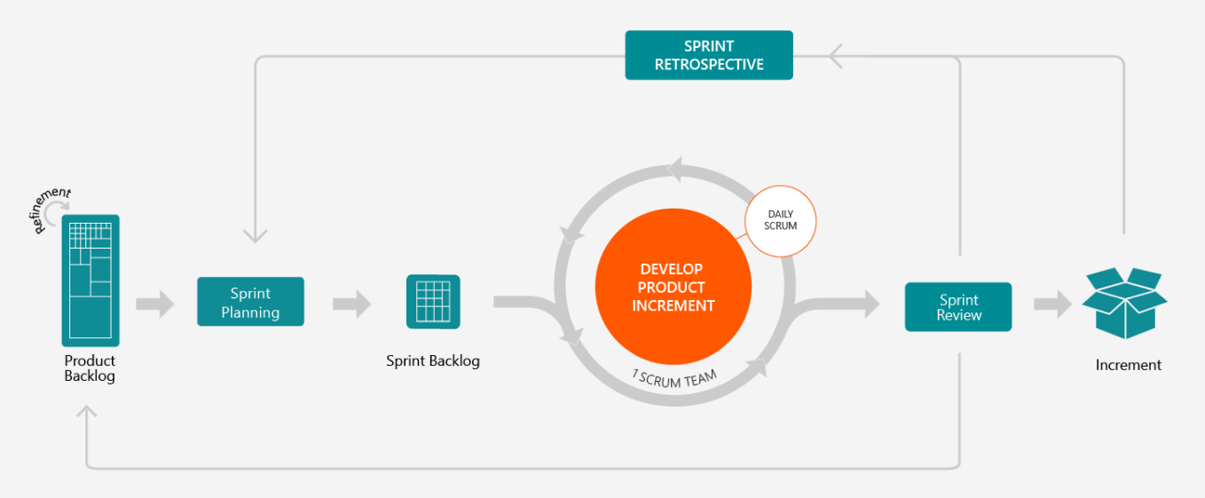


Figure 1: Sprint Lifecycle

As shown in Figure 2 below, the ADF takes the Scrum construction lifecycle and extends it to show the full delivery lifecycle from the beginning of the project to the release of the solution to production.

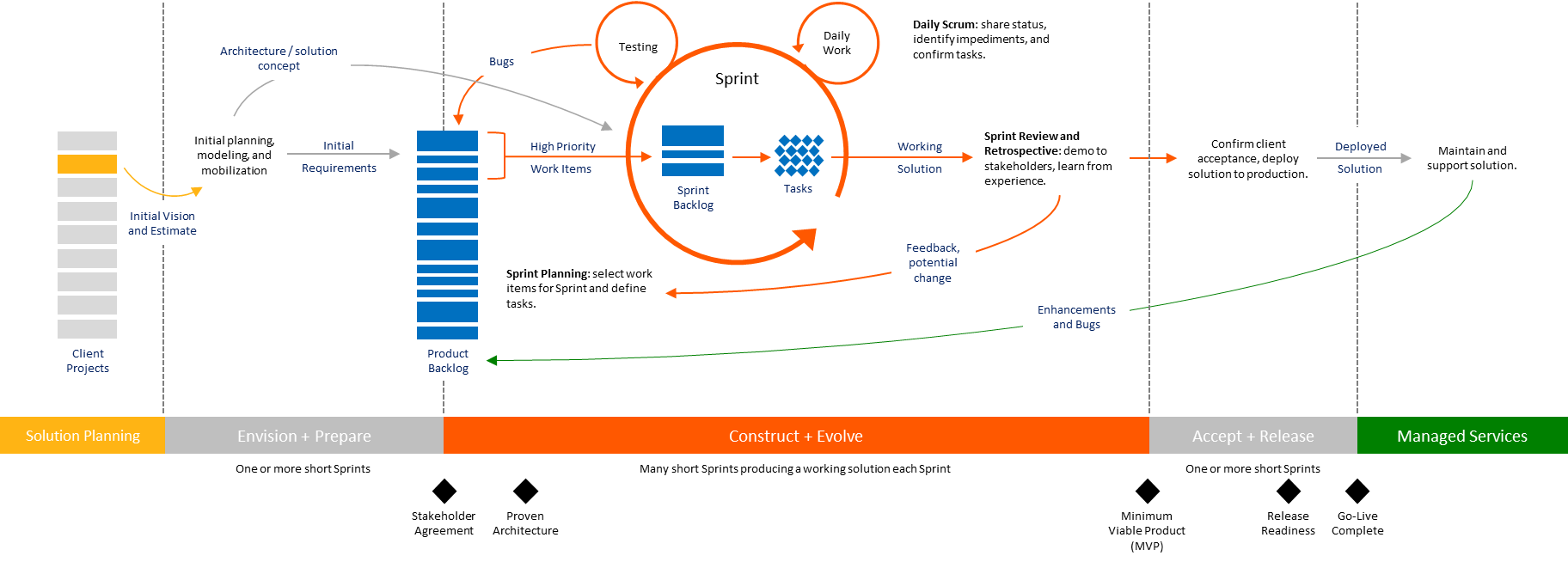


Figure 2: Agile Delivery Lifecycle

In a typical Avanade project, there are multiple releases to production, and for each release there are three categories of activities that are performed - Envision and Prepare, Construct and Evolve, and Accept and Release.

The important thing to keep in mind is that these are groups, or categories, of activities that have a common goal; these are not formal lifecycle stages or phases, as can be seen in Figure 3 below, which shows a hypothetical Agile project with two releases of six Sprints each. The boxes in the diagram represent the activities that the Scrum Team performs during each Sprint. Note that the Envision and Prepare activities at the beginning of the project (Release 1) look like a stage/phase because these activities generally need to be performed first to mobilize the project team, establish the project environments, and so on before the Scrum Team is ready to begin developing the solution; however, there is no formal stage containment.

Construct and Evolve and Accept and Release activities can be performed together throughout the release to build and test the solution functionality and prepare to roll out the solution to the business and end users. In less mature teams you may see something that looks like a Construct and Evolve phase followed by an Accept and Release phase, but for highly mature teams these activities might all be performed together in the same Sprint. The diagram below shows a typical approach where Accept and Release activities are sprinkled throughout the Sprints along with the Construct and Evolve activities, but there is a final Sprint for Systems Integration Testing (SIT) and User Acceptance Testing (UAT) prior to the release.

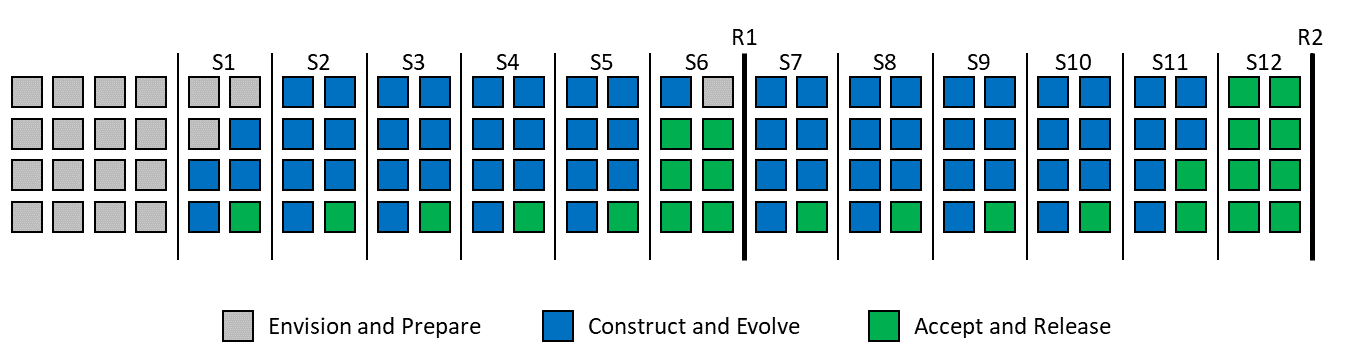


Figure 3: Hypothetical project with two Releases

### Milestones

From a development perspective, each Sprint provides an increment of functionality to the product. The end of each Sprint corresponds to a checkpoint where the Scrum Team demonstrates to Stakeholders that the objectives for the Sprint are met.

From a management perspective, each release is tracked through the completion of several milestones. These milestones provide evaluation criteria that aligns to the objectives of each set of the Envision and Prepare, Construct and Evolve, and Accept and Release activities. Achieving a milestone represents an objective criterion used to measure progress. The key milestones are:

* Stakeholder Agreement
* Proven Architecture
* Minimum Viable Product (MVP)
* Release Readiness
* Go-Live Complete

### Risk vs. Value

In the Agile Delivery Lifecycle, a risk-value approach provides Stakeholders with visibility on two main drivers:

1. Risks need to be driven down
2. Value needs to be driven up

The ADF addresses this in two ways. First, it prescribes specific Envision and Prepare activities - upfront analysis and design activities - needed to produce an initial indicative Release Plan that the Scrum Team is confident they can deliver. These upfront analysis and design activities are necessary to uncover risks and complexities early in the project lifecycle.

The second way the ADF addresses the risk-value tradeoff is by focusing the Delivery Team on risk reduction while tracking value creation. Agile Elaboration is a potential reordering of a few work items to push the technically risky ones to the top of the stack so that the Scrum Team considers both the technical risks and the value creation. This is made transparent through the Product Backlog.

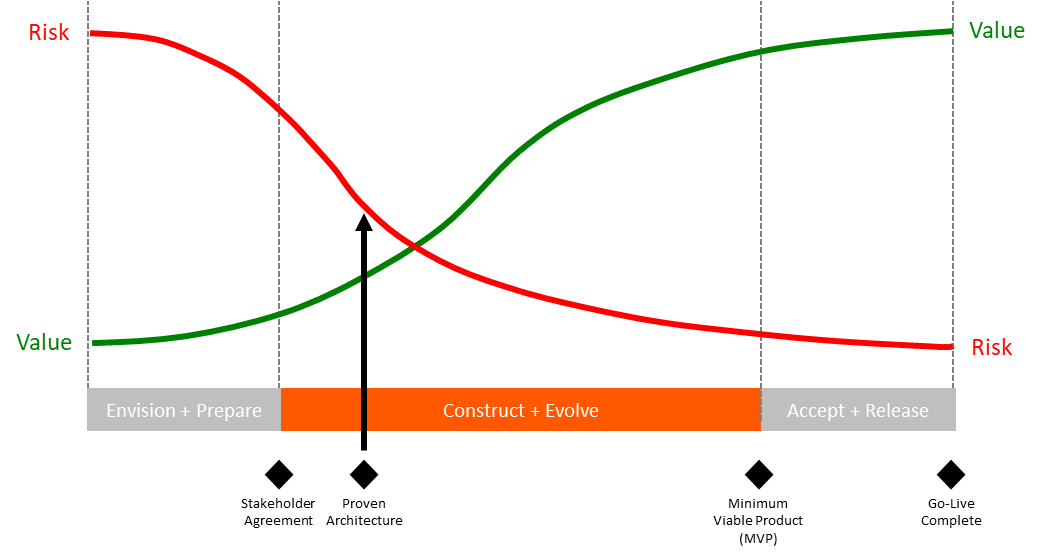


Figure 4. Risk reduction and value creation during the project lifecycle

Risk is a manifestation of the likelihood of unexpected things happening to the project that stand in the way of value creation. Risk is directly proportional to uncertainty in estimates and Stakeholders typically want to know early on what value the project can deliver in the stipulated time. In many cases, you reduce risk when you create value by implementing and testing the most critical capabilities. However, there are situations where risk reduction and immediate value creation are at odds with each other, requiring careful balancing of these competing priorities to maximize stakeholder value.

## Traditional Delivery

Avanade is seeing more and more of our clients embracing Agile delivery; however, some clients still prefer, or the situation may warrant, a traditional waterfall or iterative delivery approach, which is based on the principle of stage containment. Stage containment is an approach to minimize the number of problems discovered after work has been completed within one stage and passed on to a subsequent stage. This is accomplished by dividing the project lifecycle into stages and reviewing, testing, and approving the work products produced in each stage before proceeding to the subsequent stage. Stage containment is critical because problems become increasingly more expensive and difficult to fix as they are found later in the development lifecycle.

Using stage containment involves:

* Planning for and using techniques to identify problems as early as possible
* Adopting organizational approaches to reduce hand-offs and provide ownership for the deliverables being created
* Using techniques to monitor and measure containment

By concentrating on containment, a project can decrease the cost of fixing problems and minimize the number of residual problems in the finished solution. Containment also provides an impetus for fixing the process that originally caused the errors, or the process built to prevent them.

In the ADF, the project lifecycle is broken into the following six stages:

1. Plan - review and confirm the work products created in Solution Planning.
2. Analyze - develop the complete requirements pack describing the complete solution functionality.
3. Design - create detailed designs for all solution components.
4. Build - build and unit test the solution components.
5. Test - test the solution components and the end-to-end solution.
6. Deploy - deploy the solution to the business and end users.

In waterfall delivery, each stage is performed sequentially, one at a time, as illustrated in Figure 5 below.

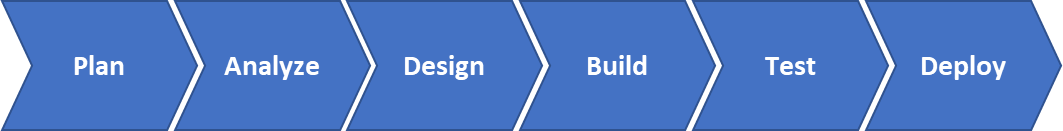


Figure 5. Waterfall Development Approach

In iterative delivery, the work is organized into a series of iterations where each iteration encompasses a series of development activities (e.g., design, build, and test) that results in a working product that is progressively more complete with each iteration, as illustrated in Figure 6 below.

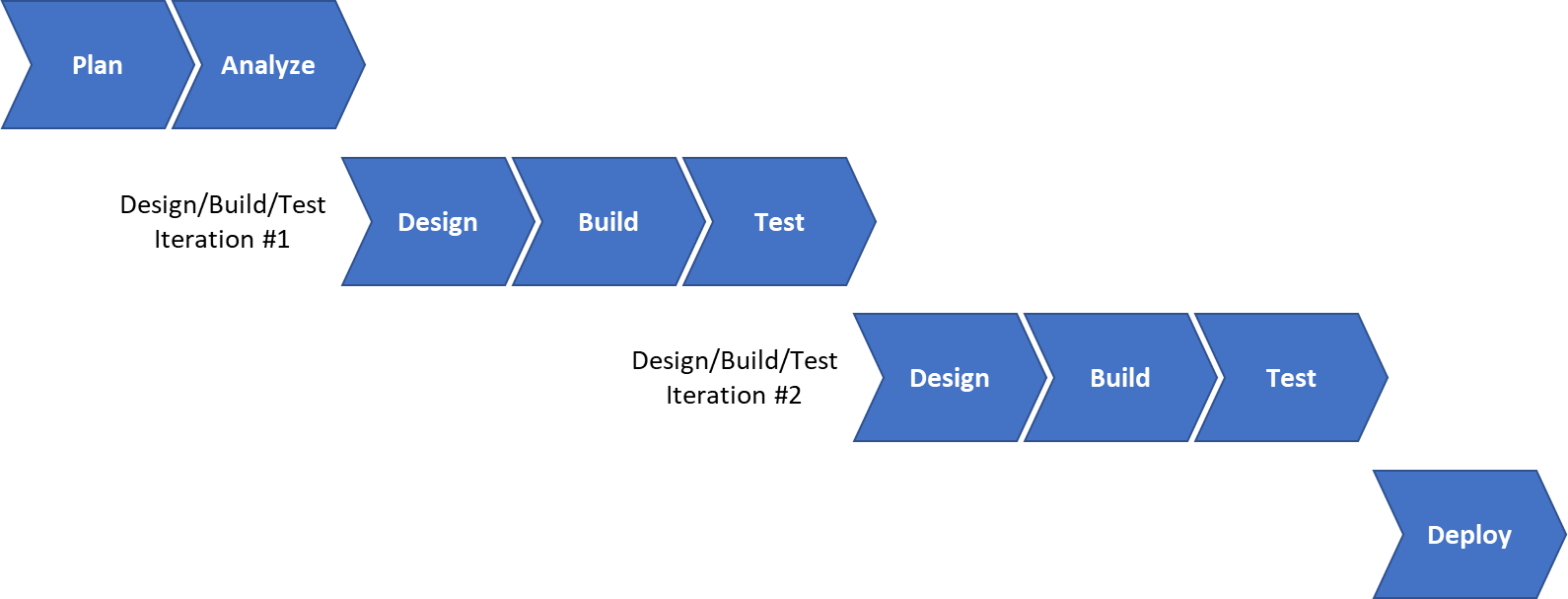


Figure 6. Iterative Development Approach

# ADF is a *Structured* Approach

ADF is a structured approach, which consists of three levels of detail.

## Level 1 - Disciplines

Level 1 is the high-level structure and grouping of the activities in the methodology. Activities are grouped into disciplines each with its own Activity Diagram showing the process flow for the activities in the discipline. This level is intended to communicate the process flow for each of the disciplines to the Senior Management team, and from Delivery Leads/Project Managers to project teams. For example, Figure 7 below shows the Activity Diagram for the Testing discipline.

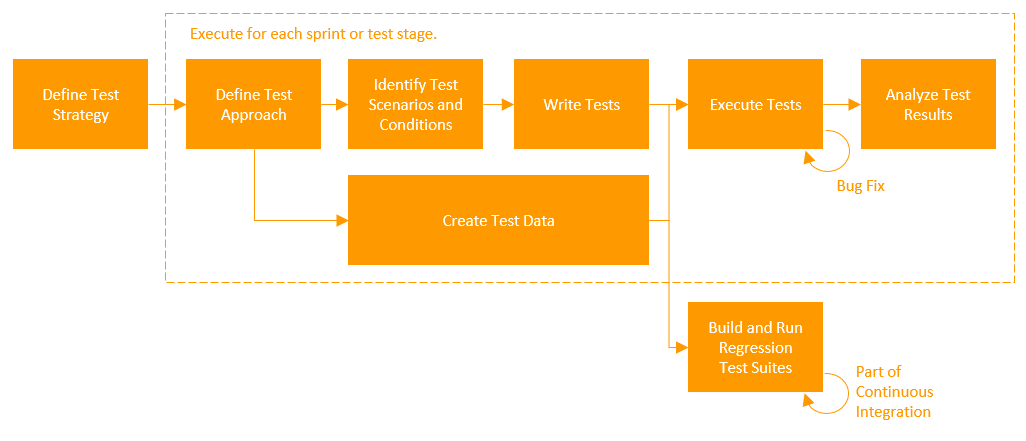


Figure 7. Delivery Leadership Activity Diagram

## Level 2 – Activities and Work Products

Level 2 is the activity and work product descriptions, which describe what to do and what to produce to deliver the solution. This level is intended for Project Managers and Delivery Leads to plan, execute and monitor the project’s work. For example, Figure 8 below shows the tasks that are performed in the Define Test Strategy activity in the Testing discipline.

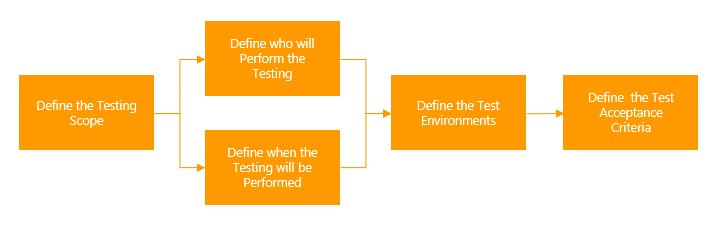


Figure 8. Define Test Strategy Activity

## Level 3 – Samples/Templates, Guidance, Tool/Asset Recommendations

Level 3 is the samples and templates, guidance, and tool/asset recommendations that enable our consultants to perform the work described in the activities or create the work products as required. This level is intended for project team members using the method to complete their work.

# ADF Disciplines

ADF is composed of seven disciplines described in the following sections. In additional to the seven core disciplines, the Secure Software Development Lifecycle (SDLC) provides cross-cutting guidance on delivering secure software solutions for our clients.

Each discipline is owned by a group of subject matter experts from across Avanade who meet at least yearly to discuss additions or enhancements to the content to support areas of increased or newly identified risk as well as suggestions from the field.

ADF has a structured feedback process so that our consultants in the field have easy access to the content owner groups for each discipline to ask questions or provide feedback.

## Delivery Leadership

The Delivery Leadership discipline focuses on directing the activities that take place on the project. This includes managing risks, directing people (onboarding Delivery Team members, tracking progress, etc.), and coordinating with people and systems outside the scope of the project to be sure that it is delivered on time and within budget.

The Delivery Leadership discipline begins with the planning and mobilization activities required to begin the delivery phase of the project and facilitate long-term successful delivery. Effective planning and mobilization ensures we are leveraging our best practices for tools, methods and processes, ensures that the project team is immediately productive, and puts controls in place for ongoing leadership of the project.

It also includes the key leadership practices, such as client expectations management, deliverables management, quality management, and risk and issue leadership needed to monitor and control the project.

Finally, it includes the leadership activities to close the project.

## Requirements Modeling

The focus of the Requirements Modeling discipline is on understanding the client and their actual needs while exploring in scope areas/domains to understand and document business requirements.

Some initial requirements modeling is performed in the Envision and Prepare phase to understand the Stakeholder goals and expectations, business requirements, and solution features/scope; to define the overall project vision, and to create the initial Product Backlog.

More detailed requirements modeling is performed just-in-time (JIT) during the Sprints in the Construct and Evolve phase to elaborate user stories in the Product Backlog to the point that they are ready for the Developers to develop them. This detailed requirements modeling is mainly performed during the Backlog Refinement sessions. These sessions are usually held towards the end of each Sprint to prepare the user stories that will be developed in the next Sprint.

## Design Modeling

In the Design Modeling discipline, the Developers on the Scrum Team model the solution at the right level of detail so that it can be developed. Design modeling consists of high levels and low levels of abstraction. The high-level view of the solution is commonly referred to as the Architecture, and the low-level view of the solution is referred to as Design (or Technical Design or Detailed Design). Keep in mind that for an Agile project, both high-level and low-level designs are only documented when needed and at the level needed to provide the guidance to develop, support, and maintain the solution.

In the Envision and Prepare phase, the Developers will confirm the Solution Blueprint defined during Solution Planning and reconfirm stakeholder agreement. They will define the modeling strategy to be used to capture the Architecture and Design and will perform additional architecture modeling if needed. User interface (UI) design is also generally begun in Envision and Prepare where the overall structure and navigation are defined.

In addition to describing the architecture, the Developers will establish the standards, rules, and guidelines they will follow to deliver the solution, including design, programming, and documentation standards for component designs, code, and/or architecture components and services.

The Developers will perform more detailed design modeling during each Sprint in the Construct and Evolve phase. Design modeling is performed just in time (JIT) with just enough modeling to think through the design of a single backlog item, or portion thereof, before implementing the backlog item. Consideration as to how much design is required 'up front' and how much can be tethered to the PBIs is part of defining the modeling strategy.

The first couple of Sprints in the Construct and Evolve phase focus on proving that the architecture works by developing an end-to-end architectural prototype for the solution, thereby mitigating much of the technical risk on the project. Technical prototypes such as this are production quality code that forms the foundation, or skeleton, of the solution. Additional technical prototyping, or spikes, can be performed at any time as needed.

## Development

A disciplined development model is essential to successful delivery. The additional effort required by the developer to follow a disciplined development process reduces overall project effort by eliminating bugs at the source. The four main steps in a disciplined approach are understanding, preparation, execution and verification.

1. Developers should strive to fully understand the context of the functionality they are developing in order to make better decisions and remove ambiguity early in the lifecycle.
2. Whether it is formal class level design documents or a simple sketch, developers should perform some preparation prior to starting development.
3. Development execution should remain focused and unit testing should be followed by code analysis and refactoring where required.
4. Finally, before functionality can be said to be complete, the developer must verify that the functionality works in a production-like integration environment that includes all other solution components.

## Testing

The Testing discipline covers the higher-level testing activities beyond unit testing, which is included in the Development discipline, and includes the planning, preparation and execution of the tests.

Regardless of the project's delivery approach - Agile delivery or traditional waterfall/iterative delivery - the first, and most crucial activity, is to define the strategy to be taken for fully testing the solution. The Test Strategy should define the scope of the testing activities, who will perform the testing, when the testing will be performed, the tools and environments required for testing, and the test acceptance criteria. For each Sprint or test stage, the team will perform more detailed test planning to define the test approach for that Sprint or test stage.

While the details of the testing in each Sprint or each test stage differ, the process behind them is generally the same:

1. Define and agree the testing approach for the Sprint or test stage
2. Design and prepare the tests
3. Execute the tests
4. Stabilize the solution through bug fixing and retesting

## Release Readiness

The Release Readiness discipline covers activities for deploying the solution to the production environments and releasing it to the end users. This includes preparing the production and operating environments for solution rollout to the end users, enabling end users and support personnel to use or support the new solution, rolling out the new solution to the deployed unit, and transferring the responsibility for operating and maintaining the solution to the operating group.

Enabling end users and support personnel to use or support the new solution includes the creation of appropriate documentation, such as user manuals, online help or online quick reference materials, user procedures, job aids, etc. It also includes end user and support training if in scope for the project.

## Environment and Configuration Management

The Environment and Configuration Management discipline deals with the technology services, tools, techniques, and standards for managing the entire software development lifecycle of client solutions, including operations.

The discipline consists of these three major components.

* Development Architecture covers the configuration and deployment of the infrastructure and software required to run the project, such as source code control servers and release management systems.
* Operations Architecture covers the configuration and deployment of the infrastructure and software required to operate the solution and the solution infrastructure once deployed to an environment.
* Environment Management is the management of all infrastructure (both physical and virtual) required to test the solution and to operate and support the solution in production.

The interaction between Development and Operations around the management of the solution and the environments is commonly referred to as DevOps.

## Secure Software Development Lifecycle (SDLC)

The Avanade Secure Software Development Lifecycle (SDLC) defines Avanade’s approach to software security assurance, and the secure application development methodologies used by Avanade to develop software and applications for clients, or tools and technologies that can access information assets, critical infrastructure, or any key infrastructure. As a Microsoft Gold Partner, Avanade aligns its software security assurance methodology to Accenture’s Secure Software Development Lifecycle (SDLC) which includes key elements from the Microsoft Security Development Lifecycle (SDL).

The Microsoft SDL is a software security assurance process that is utilized by Microsoft, its partners and customers to help build more secure software and address security compliance requirements early in the application development lifecycle. The Microsoft SDL helps reduce software vulnerabilities and exploits during application development. The Microsoft SDL is also utilized when moving apps to the Microsoft Azure public cloud, along with the Microsoft Operational Security Assurance (OSA) process for cloud operational security. The Microsoft SDL is based on three core concepts—education, continuous process improvement, and accountability.

The Avanade SDLC provides detailed guidelines to support Avanade’s security policy, and application development teams and Avanade executives who approve the release of software developed for clients must ensure that the practices outlined in this SDLC are followed. These practices include establishing security requirements, incorporating the security requirements into the application design, and conducting a Final Security Review (FSR).

# Integration with other Avanade Assets

By integrating ADF with delivery tools and accelerators, Avanade can better manage software development and IT initiatives by automating much of the process from end to end.

Avanade project teams use the Avanade Scrum+ Process Template for Azure DevOps Services to support the project lifecycle. The Scrum+ Process Template provides a set of customizations and best practices for Avanade projects looking to leverage Azure DevOps, including the following:

1. Custom work items for risks, issues, decisions/assumptions, change management, questions, milestones, and dependencies.
2. A set of shared queries that support the management of scope changes, dashboards, day to day Sprint management activities, and work item maintenance.
3. A set of dashboards to support the Scrum Team and provide information to Stakeholders and Program Management.
4. A set of PowerBI Reports that support Sprint and Release Management

As part of Avanade’s delivery model and collaborative approach, project teams also use Microsoft Teams and SharePoint® technology to connect with clients through a common digital workspace.

By integrating our proven methods and tools our clients can achieve compelling results:

* Increased collaboration between project team and IT—better alignment of the solution with the business.
* Increased traceability across the lifecycle, which promotes quality.
* Improved project management, through a unified view of project reporting and tracking.
* High quality so the final solution meets the business requirements and meets quality of service.
* Quicker time-to-market due to increased efficiency and productivity.
* Increased flexibility to build and adapt the solution to support potential changing business requirements.