EXIN Agile Scrum Foundation

Module 1 Study Guide
Agile Way of Thinking

Agile Scrum Foundation

Welcome to your Study Guide

This document is supplementary to the information available to you online, and should be used in conjunction with the videos, quizzes and exercises.

Study Guide Icons

	**Tip**	This will remind you of something you need to take note of or give you some exam guidance.
	**Definition**	Key concept or term that you need to understand and remember.
	**Role**	Job title or responsibility.

Agile Scrum Foundation

Contents

Agile Way of Thinking

	The objectives for this module were for you to: ▪ Explain why Agile methodologies are important and how they add value ▪ Describe the Agile Manifesto and Agile principles ▪ Recognize and explain the Agile framework

How Adaption to an Agile Framework Works

These steps, or delivery processes, are used for developing a piece of software:

- Analyze
- Design
- Construct
- Integrate
- Test

In this predictive form of a lifecycle, each process is completed before we proceed to the next:

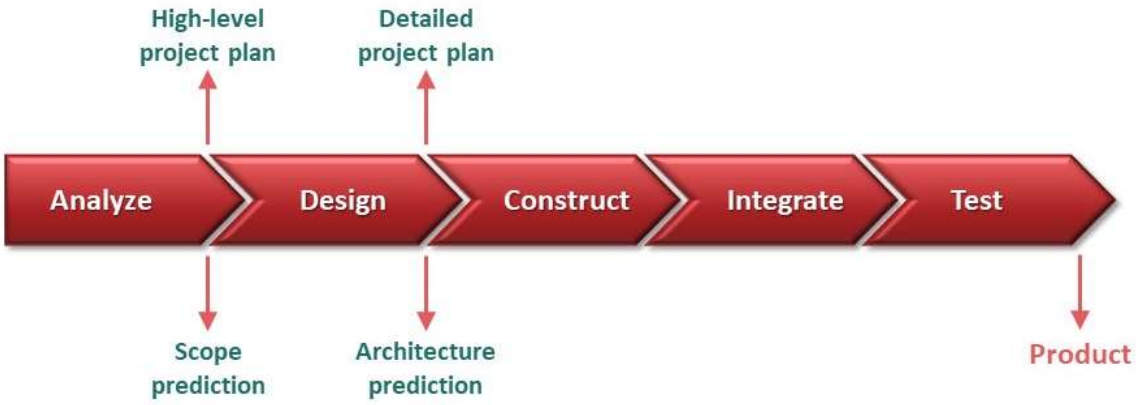

Figure 1.1 Generic Lifecycle

The steps can overlap, as can be seen below, but the main driver of the cycle is the sequence of a development process:

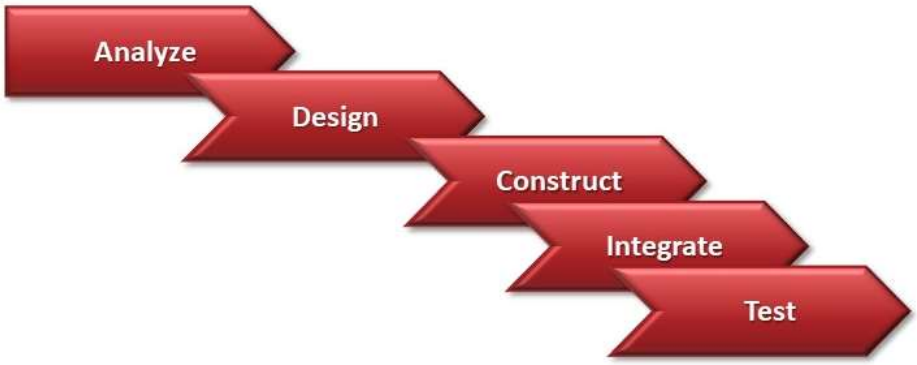

Figure 1.2 Step Lifecycle

The specification and design are planned first and the plan is then used to predict what is needed and how to do it.

The word waterfall is commonly used to refer to predictive lifecycles used in IT projects where progress is seen as flowing steadily downwards. The waterfall development model comes from the manufacturing and construction industries; physical environments where after-the-fact changes are costly, if not impossible. Since no formal software development methodologies existed at the time, this model was simply adapted for software development.

The first formal description of the waterfall model is often cited as a 1970 article by Winston W. Royce who presented it as an example of a flawed, non-working model. When writing about software development, waterfall is generally seen as a negative way to carry out software development, as the customer may not be happy with the end result and want changes.

	**Definition**	**Quote** "People don't know what they want until you show it to them." **Steve Jobs**

The customer often only sees the product towards the end of the project when the cost of change is the highest. To overcome this problem, the product can be created incrementally.

In an adaptive lifecycle increments of the product are created in short periods of time. Then they are shown to the customer before a decision is made on what to do next based on their feedback:

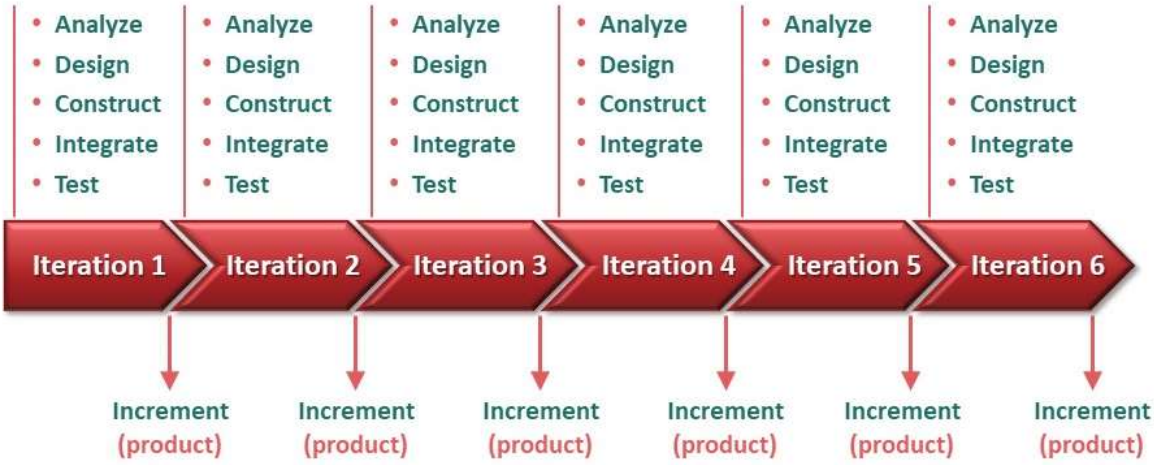

Figure 1.3 Adaptive Lifecycle

Each increment is created by running all the development processes during that period and then repeating the process in the next instance of time. This is sometimes called iterative development, and the periods of time within which iterative development takes place can be called iterations.

Agile is the popular term for systems that use adaptive lifecycles; research shows Agile projects deliver higher success rates when compared to the waterfall approach.

	**Agile Project**	"In 2002, Agile projects made up less than 2% of overall projects and less than 5% of new application development projects. Today, Agile projects account for almost 9% of all projects and 29% of new application development projects [...] The increase in project success rates can directly tie back to projects resolved through the Agile process." Source: The Standish Group: CHAOS Report

The term 'Agile project' can be a contentious one, as many practitioners feel that Agile development and traditional project management have fundamentally different aims and ways of working. When the expression was

used in this course, the term 'project' was applied in its broadest sense, for instance, as a container for work.

The Agile Manifesto and Agile Principles

The Agile Manifesto was developed in 2001 and documents the underlying concepts of Agile software development. The manifesto was developed by a group of 17 practitioners, from different programming methodologies.

It's important to look at the wording of the manifesto. It says that there is value in the items on the right but emphasizes the value of the items on the left more:

Manifesto for Agile Software Development

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions…over processes and tools

Working software over…comprehensive documentation

Customer collaboration…over contract negotiation

Responding to change…over following a plan

That is, while there is value in the items on the right, we value the items on the left more."

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

	These four bullets are also referred to as a 'statement of value.' You need to remember this term and revise the Agile Manifesto for your exam.

Statement of Value Explanation

Statement of Value	
Individuals and interactions…over processes and tools	Tools are sophisticated pieces of software that create many projects and jobs, but they won't solve problems caused by overlooking human aspects and they're not a magic wand. Tools can only facilitate a system; they don't replace the need for a system. Processes that try to ignore or replace human aspects are bad, and processes that address individuals and interactions and make them part of the system are good.
Working software over…comprehensive documentation	This applies adaptive thinking and favors adaptive systems that create pieces of software or increments, and using them to adapt, rather than planning what will happen through documentation.
Customer collaboration…over contract negotiation	In an Agile system collaboration is important to specify new requirements, check increments and gain feedback, meaning that parties can work together towards creating a product that adds value. It is important to remember that customers still often want fixed scope, fixed price contracts which conflict with Agile methods.
Responding to change…over following a plan	This means that instead of a predictive, upfront plan we will adapt or 'change' as the project evolves.

Agile Features

Agile Feature	Description
Focus on people	Agile is focused on professional people and how to use their professionalism optimally to produce high quality products and services.
Working software	Agile aims to deliver high quality working (functioning with needed functionality) software products.
Flexibility	Flexibility is the ability to adjust to changing conditions, an important element of being Agile.
Customer involvement	Customers or user representatives are frequently involved and asked to give their opinion and help to prioritize.
Multidisciplinary, cooperating teams	Different professionals with different specialties will be part of a Scrum team, together delivering the requested products.
Trust	This is the basic ingredient to be able to deliver the requested quality.
Iterative	Start with a minimum of required functionalities to create the increment, in the next sprints these are gradually expanded on, or we iterate.
Incremental	First a skeleton or core functionality is produced, the 'flesh and skin' are added later, as the overall architecture emerges.
Time boxes	Agile teams work in time boxes; a time box has a maximum length but usually not a minimum! Having a fixed duration continuously pushes the team to focus on the most valuable things first.

In Agile principles the maximum duration of a time box is 2 months, but in Scrum it is 1.

Scrum

Scrum is the most commonly adopted Agile framework for project management. Scrum represents a significant shift in terms of how projects are managed. The level of change should not be under-estimated by an organization seeking to adopt Scrum.

	**Sprint**	Sprint is another term for iteration, it is a small, timeboxed period of time.

Agile ways of working are popular and considered to be the best for software development projects, but it is worth noting that Agile projects can be difficult, especially if a development team has to answer to shareholders rather than customer demands. Successful teams are given the freedom to respond to change and use evidence-based decisions to create value for the customer.

	You can read more about agility in this article entitled 'Why does every project have to be Agile these days?' https://mailchi.mp/gothelf/services1185141?e=af0a1f1618

Agile Principles

There are 12 Agile guiding principles:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale
- Business people and developers must work together daily throughout the project
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation
- Working software is the primary measure of progress
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely

- Continuous attention to technical excellence and good design enhances agility
- Simplicity--the art of maximizing the amount of work not done--is essential
- The best architectures, requirements, and designs emerge from self-organizing teams
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

Agile Frameworks

Agile is a term that includes many different iterative and incremental software development approaches. An Agile framework is an approach based on the philosophy of the Agile Manifesto. A project that uses an Agile framework or methodology will always involve continuous planning, testing, integration and development.

Agile has developed into many different 'flavors', most organizations adopt a custom hybrid approach based on a mixture of these:

- Extreme Programming (XP)
- Test Driven Development (TDD)
- Feature Driven Development (FDD)
- Crystal Clear
- Dynamic Systems Development Model (DSDM)

Extreme Programming

XP is a very powerful and practical Agile method of working.

	**Extreme Programming**	'Extreme programming is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements.' Source: https://en.wikipedia.org/wiki/Extreme_programming

One XP practice is called pair-programming, meaning developers work in pairs, with one person coding and the other observing and commenting until they switch at a time they have specified. Even though the pair of programmers are responsible for the code, everyone in the team is accountable for it, this is called 'collective code ownership'.

Pair-programming might seem like a waste of resources, but the benefits are:

- Work quality is increased which prevents rework that will cost resources and time
- The expertise of the developers is increased
- It increases the bus factor of the team
- It creates a continual team-building activity

	**Bus Factor**	'The bus factor is a measurement of the risk resulting from information and capabilities not being shared among team members.' Source: Wikipedia

Test-Driven Development

Another XP practice is Test-Driven Development (TDD). TDD is also known as test first development. In this approach, the test is created first and then as much code as needed is written to pass the test. Creating the test first rather than starting straight away can make a project seem slow, but this approach means everyone is focused on solving the problem and often results in a higher quality, faster, maintainable solution.

The advantages of TDD include:

- Always having a complete set of tests for the whole system. Every time you add a new feature, you can easily run the tests, ensuring the integrated system works and nothing is broken in the old code
- Everyone is focused on the problem they are going to solve, instead of the solution

	It is worth noting that TDD requires discipline and a long-term viewpoint of the project!

Regardless of the framework, and even with thorough testing, escaped defects might be found by the customer when the version is released, instead of the

quality assurance team. It is important to limit the number of escaped defects and monitor trends in those so they can be responded to.

For example, if the escaped defects are increasing a little after each iteration, the aim is to find the cause and then revise the definition of done. This metric for Agile teams helps with tracking and sends the correct message to the developers:

functionality that is released, should be of good quality

	**Definition of Done**	The Definition of Done is prepared at the start of the project and shows what has to be done for each item to make it potentially releasable or done.

Crystal Clear

Crystal is a set of Agile methods focused around communication, people, skills and interactions, with the different types of crystal indicating the size of the project, team and approach. Crystal Clear is used for the smallest team to improve productivity and involves co-locating teams to the same room and having osmotic communication.

	**Osmotic Communication**	Indirect information transfer through overhearing conversations or simply noticing things happening around you.

Osmotic communication can still be achieved with teams that are distributed elsewhere through things like group email and applications such as Slack.

	**Slack**	Slack is an American cloud-based set of proprietary team collaboration tools and services, founded by Stewart Butterfield. Source: Wikipedia

Dynamic Systems Development Method

Dynamic Systems Development Method (DSDM) is an Agile method that has a project manager and supports multiple teams and large projects. In a DSDM project the scope is dynamic, but the time, cost and quality of a project is fixed, meaning the project is timeboxed.

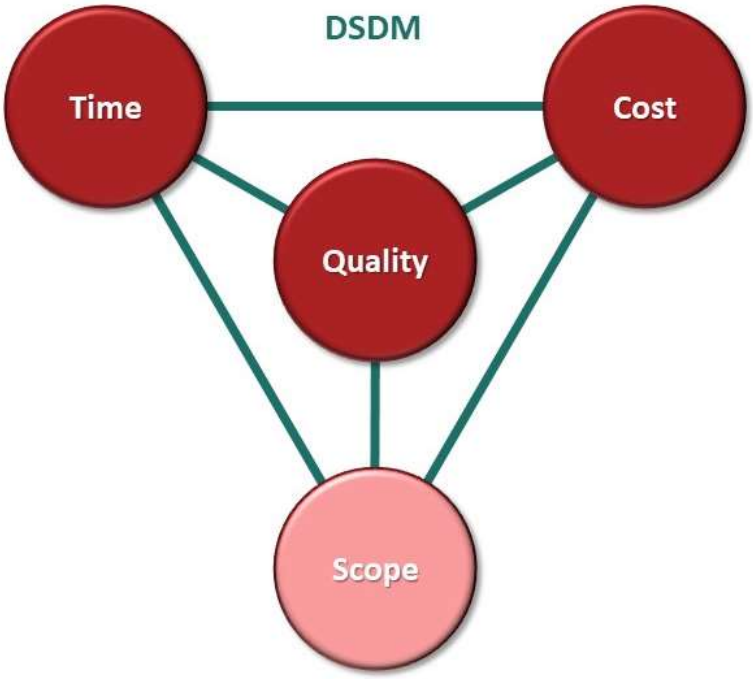

Figure 1.4 DSDM Project Constraints Triangle

The timeboxed project is delivered on time with the aim of delivering as much as possible, it also gives the opportunity to pause to review the product. If necessary, another timeboxed DSDM project can be started to add more features.

Agile allows up front planning to work out the duration of the project and cost, but detailed plans should be avoided as they hinder adaptation. DSDM uses a high-level upfront plan as a basis for setting the duration and cost of a project, and then applies a technique called MoSCoW® prioritization to manage scope. The MoSCoW menomic combines the first letters of: Must-Have, Should-Have, Could-Have, and Won't-Have-This-Time. The minimum for an acceptable product would be all the M features and the ideal would have all the M, S and

C user features. The duration for developing these features would be the timeboxed duration of the project.

The Scrum Framework

Agile is a high-level concept and needs a practical approach to run projects based on it such as methodologies and frameworks. Scrum is a lightweight process framework designed to manage software and complex product development. It promotes developing products of the highest possible value in an iterative and incremental way. Scrum is not a process or a technique for building products, it is a framework which can be used to employ various processes and techniques. Processes and techniques based on Scrum produce potentially shippable sets of functionality at the end of every iteration.

Scrum Framework/Practices		
Roles	**Rituals**	**Artifacts**
Product Owner	Planning	Backlogs
Scrum Master	Daily stand-up/Scrum	Burndown chart
Development Team	Review	Definition of done
	Retrospective	Potentially shippable product increment

	**Burndown Chart**	Used to track progress on a Scrum project through a downward slopping chart

Each Scrum project is done in a number of sprints. Sprint is the Scrum term for iteration. A product backlog is used to define the remaining scope of the product.

	**Scrum Product Backlog**	The Scrum product backlog is simply a list of all things that needs to be done within the project Source: https://www.scrum-institute.org/

A number of items are picked from the top of the product backlog and added them to the sprint backlog, which then becomes the plan for the upcoming sprint. Sprints are run as many times as required until either:

- The project is finished because all the items in the product backlog are done, or the customer has realized that the latest increment is enough, and there is no justification to spend more time and money adding more features
- The project is terminated for some reason, for example, no longer justifiable

The events inside the sprint are all timeboxed, and run as follows:

- **Sprint Planning:** a short timebox for selecting the user stories from the top of the product backlog and creating the sprint backlog
- **Daily Scrum (Daily stand-up):** a 15-minute timebox to collaborate and coordinate on a daily basis
- **Sprint Review:** for demonstrating the increment and communicating progress to the customer and receiving feedback
- **Sprint Retrospective:** for reviewing the way of working and planning for improvements in the next sprint

	**User Story**	A user story is a description of one or more features of a software system, usually written from the perspective of an end user or user of a system, like this: As a '*role*', I want to do '*something*', in order to '*explain purpose*', for example: *As a user, I want to reset my password to regain access to my account.*