October 28, 2019

Taha Aflouk

5N2772

Skills Demonstration 2

Software Architecture

Table of Contents

[1 – History and Evolution of Agile 2](#_Toc27245535)

[1.1 – What is Agile? 2](#_Toc27245536)

[1.2 - Where did Agile come from? 3](#_Toc27245537)

[1.3 - Reason for its development 3](#_Toc27245538)

[1.4 – Other Models 5](#_Toc27245539)

[2.0 – Agile Manifesto 5](#_Toc27245540)

[2.1 – Contributors 5](#_Toc27245541)

[2.2 – Principals 5](#_Toc27245542)

[3.0 – Agile vs Waterfall 6](#_Toc27245543)

[3.1 – Advantages 6](#_Toc27245544)

[3.2 – Disadvantages 6](#_Toc27245545)

[Bibliography 7](#_Toc27245546)

# 1 – History and Evolution of Agile

First came the crisis

In the early 1990s, as PC computing began to proliferate in the enterprise, software development faced a crisis. At the time, it was widely referred to as "the application development crisis," or "application delivery lag." Industry experts estimated that the time between a validated business need and an actual application in production was about three years.

The problem was, businesses moved faster than that, even 25 years ago. Within the space of three years, requirements, systems, and even entire businesses were likely to change. That meant that many projects ended up being cancelled partway through, and many of those that were completed didn't meet all the business's current needs, even if the project's original objectives were met.

In certain industries, the lag was far greater than three years. In aerospace and defense, it could be 20 or more years before a complex system went into actual use. In an extreme but by no means unusual example, the Space Shuttle program, which operationally launched in 1982, used information and processing technologies from the 1960s. Highly complicated hardware and software systems were often designed, developed, and deployed in a time frame that spanned decades.

## 1.1 – What is Agile?



In 2001, a small group of people, tired of the traditional approach to managing software development projects, designed the agile manifesto. It is a more improved method for managing the progress of software projects.

## **1.2 - Where did Agile come from?**

In 2001, a small group of people, tired of the traditional approach to managing software development projects, designed the agile manifesto. It is a more improved method for managing the progress of software projects.

A close up of a logo

Description automatically generated

“Agile is a time boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end.”

## 1.3 - Reason for its development

Here are 10 good reasons to apply agile development principles and practices…

The iterative nature of agile development means features are delivered [incrementally](http://www.allaboutagile.com/2007/03/agile-principle-5-how-dyou-eat-elephant.html), enabling some benefits to be realized early as the product continues to develop.

Research suggests about 80% of all market leaders were first to market. As well as the higher revenue from incremental delivery, agile development philosophy also supports the notion of early and [regular releases](http://www.allaboutagile.com/2007/03/agile-principle-6-focus-on-frequent.html), and ‘perpetual beta’.

A key principle of agile development is that [testing is integrated throughout the lifecycle](http://www.allaboutagile.com/2007/04/agile-development-agile-testing-is-not.html), enabling regular inspection of the working product as it develops. This allows the product owner to make adjustments if necessary and gives the product team early sight of any quality issues.

Agile development principles encourage [active ‘user’ involvement](http://www.allaboutagile.com/2007/02/principle-1-active-user-involvement-is.html) throughout the product’s development and a very cooperative collaborative approach. This provides excellent [visibility](http://www.allaboutagile.com/2007/05/beauty-of-not-doing-agile-development.html) for key stakeholders, both of the project’s progress and of the product itself, which in turn helps to ensure that expectations are effectively managed.

Small incremental releases made visible to the product owner and product team through its development help to [identify any issues early](http://www.allaboutagile.com/2007/05/beauty-of-not-doing-agile-development.html) and make it easier to respond to change. The clear visibility in agile development helps to ensure that any necessary decisions can be taken at the earliest possible opportunity, while there’s still time to make a material difference to the outcome.

In traditional development projects, we write a big spec up-front and then tell business owners how expensive it is to change anything, particularly as the project goes on. In fear of scope creep and a never-ending project, we resist changes and put people through a change control committee to keep them to the essential minimum. Agile development principles are different. In agile development, change is accepted. In fact, it’s expected. Because the one thing that’s certain in life is change. Instead the [timescale is fixed](http://www.allaboutagile.com/2007/03/agile-principle-3-time-waits-for-no-man.html) and requirements emerge and evolve as the product is developed. Of course for this to work, it’s imperative to have an [actively involved](http://www.allaboutagile.com/2007/02/principle-1-active-user-involvement-is.html) stakeholder who understands this concept and makes the necessary trade-off decisions, trading existing scope for new.

The above approach of [fixed timescales and evolving requirements](http://www.allaboutagile.com/2007/03/agile-principle-3-time-waits-for-no-man.html) enables a fixed budget. The scope of the product and its features are variable, rather than the cost.

The [active involvement](http://www.allaboutagile.com/2007/02/principle-1-active-user-involvement-is.html) of a user representative and/or product owner, the [high visibility](http://www.allaboutagile.com/2007/05/beauty-of-not-doing-agile-development.html) of the product and progress, and the [flexibility to change](http://www.allaboutagile.com/2007/03/agile-principle-3-time-waits-for-no-man.html) when change is needed, create much better business engagement and customer satisfaction. This is an important benefit that can create much more positive and enduring working relationships.

Above all other points, the ability for [agile development requirements to emerge and evolve](http://www.allaboutagile.com/2007/03/agile-requirements-just-in-time-and.html), and the ability to embrace change (with the appropriate trade-offs), the team build the right product. It’s all too common in more traditional projects to deliver a “successful” project in IT terms and find that the product is not what was expected, needed or hoped for. In agile development, the emphasis is absolutely on building the right product.

The active involvement, cooperation and collaboration make agile development teams a much more enjoyable place for most people. Instead of big specs, we discuss requirements in workshops. Instead of lengthy status reports, we collaborate around a task-board discussing progress. Instead of long project plans and change management committees, we discuss what’s right for the product and project and the [team is empowered](http://www.allaboutagile.com/2007/03/agile-principle-2-agile-development.html) to make decisions. In my experience this makes it a much more rewarding approach for everyone. In turn this helps to create highly motivated, high performance teams that are [highly cooperative](http://www.allaboutagile.com/2007/04/agile-principle-10-no-place-for-snipers.html).

(Gonçalves, 2019)

## 1.4 – Other Models

## Scrum

**“Scrum** is an agile way to manage a project, usually software development. Agile software development with **Scrum** is often perceived as a **methodology**; but rather than viewing **Scrum** as **methodology**, think of it as a framework for managing a process.”

A picture containing screenshot

Description automatically generated

* Gathering Requirement
* Team Roles
* Release Planning
* Sprints
* Burndown chart

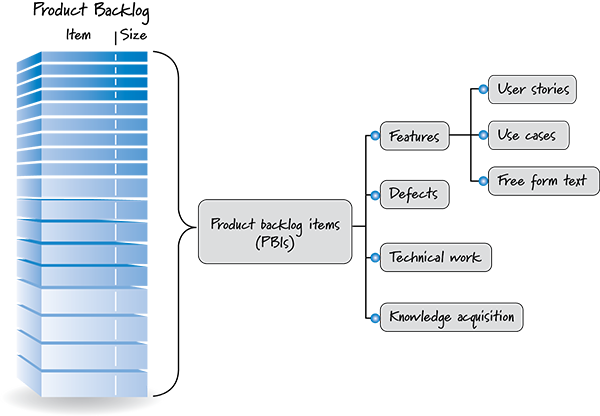
### Gathering Requirements-1

Get requirements from the customer and put them in this format.

This called a user story.

* + - As a [who]
    - I want [what]
    - Because [why]

### Gathering Requirements-1



### Team Roles

### Product owner :

Picks features

Represents the customer

### Scrum master :

Sets up meetings

Make sure everything is working smoothly

### Release planning

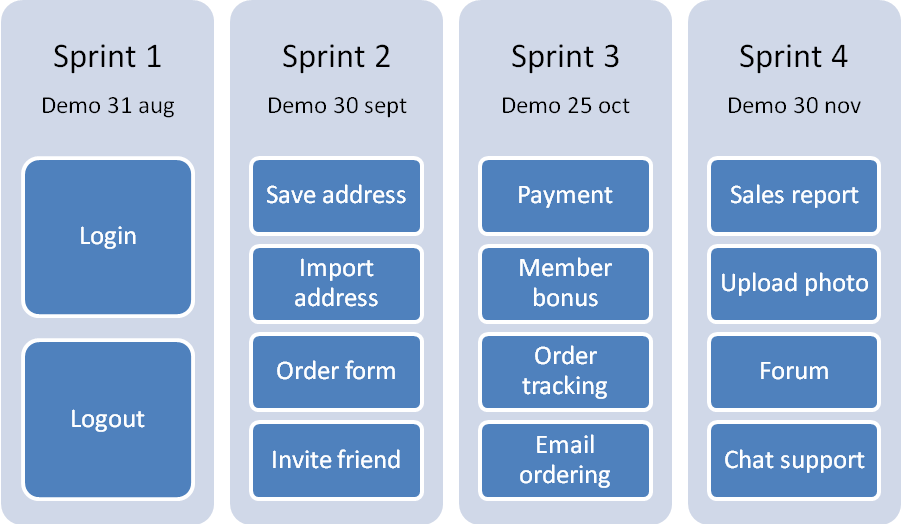
Pick user stories for next release

Prioritize

Estimate ( and possibly breakup

Now you have a release backlog and a time estimate of how work to do.

### Sprints-1



### Sprints-2

You work on some stories and get them to a ship-ready state (100% complete).

This includes doing all planning, development and testing needed for the user stories.

### Sprints-3

Daily scrum meeting

Quick 15-minute (standing) meeting everyday:

What did you do since last meeting?

Are you facing any problems?

### Sprints-4

Sprint retrospective

A meeting after a sprint is done:

What was good?

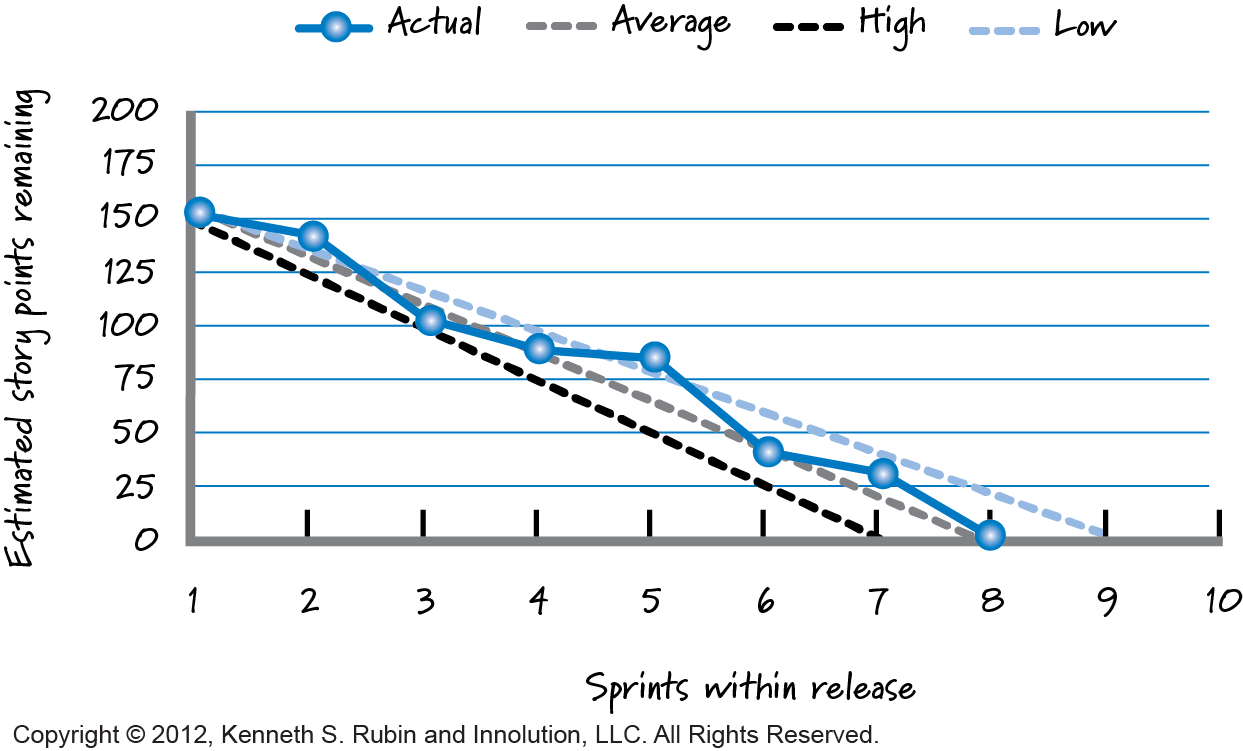
What was bad?

How to improve?

### Burndown chart

Using the time estimate, we know the amount of work in the sprint (or release).

Update this value everyday , and plot it on the chart. It helps you know if you are on track or late.



### Tools

What tools can I use to watch the scrum:

Trello is simple and free to use



<https://trello.com>

## Kanban

Kanban is a visual system for managing work as it moves through a process. Kanban visualizes both the process (the workflow) and the actual work passing through that process. The goal of Kanban is to identify potential bottlenecks in your process and fix them so work can flow through it cost-effectively at an optimal speed or throughput.

A picture containing screenshot

Description automatically generated

“An online Kanban board is a tool that helps visualize work and workflow, as well as optimize the way work gets done. Then name comes from the Japanese word Kanban, meaning “visual signal” or “card,” and also references the process improvement approach known as the Kanban Method.”

### How an Online Kanban Board Works

Like a physical Kanban board, a Kanban tool uses a series of vertical and horizontal lanes to represent workflow or process (i.e. the steps that work takes to advance from start to finish). Kanban cards, representing tasks and work items, are [moved through the lanes to reflect progress](https://leankit.com/learn/kanban/kanban-board/), using drag-and-drop functionality.

A screenshot of a social media post

Description automatically generated

## DSDM

Dynamic Systems Development Method (DSDM) is an organized, common-sense process focused on delivering business solutions quickly and efficiently. It is similar in many ways to SCRUM and XP, but it has its best uses where the time requirement is fixed.

DSDM focuses on delivery of the business solution, rather than just team activity. It makes steps to ensure the feasibility and business sense of a project before it is created. It stresses cooperation and collaboration between all interested parties. DSDM makes heavy use of prototyping to make sure interested parties have a clear picture of all aspects of the system.

A screenshot of a cell phone

Description automatically generated

### DSDM in more detail

DSDM has been developed to address common problems faced by projects such as late delivery, cost overruns or the final deliverable not being completely fit for purpose.

DSDM addresses these problems by creating an agile environment which is collaborative and flexible, yet remaining focused on hitting deadlines and maintaining the appropriate level of quality and rigour.

DSDM involves all stakeholders such as the business representatives throughout an iterative and incremental lifecycle.

All personnel involved in a project are given clear roles and responsibilities and work together in timeboxes to ensure the project is kept on schedule.

DSDM strikes the balance between performing a lot of ‘up-front’ design and performing none. DSDM believes in doing ‘enough design up-front’ in order to reduce risk yet still allow for the inevitability of change.

DSDM is particularly good at being used with other approaches. Two prominent examples of this are using DSDM with PRINCE2 which enables PRINCE2 to be easily run with an agile setting and running DSDM with Scrum which enables Scrum to be scaled up to run on more than just product development.

### The Eight Principles of DSDM

DSDM has eight principles. They represent an ethos, a culture, a way of working. The principles are actively managed at all times , because if a principle becomes compromised it represents a risk to the successful execution and completion of a project.

The eight Principles of DSDM are as follows:

1. Focus on the business need
2. Deliver on time
3. Collaborate
4. Never compromise quality
5. Build incrementally from firm foundations
6. Develop iteratively
7. Communicate continuously and clearly
8. Demonstrate control

### Extreme programing (XP)

|  |  |
| --- | --- |
| [Agile flow chart](http://www.agile-process.org/) |  |

The first Extreme Programming project was started March 6, 1996. Extreme Programming is one of several popular [Agile Processes](http://www.agile-process.org/). It has already been proven to be very successful at many companies of all different sizes and industries world wide.  
Extreme Programming is successful because it stresses customer satisfaction. Instead of delivering everything you could possibly want on some date far in the future this process delivers the software you need as you need it. Extreme Programming empowers your developers to confidently respond to changing customer requirements, even late in the life cycle.  
Extreme Programming emphasizes teamwork. Managers, customers, and developers are all equal partners in a collaborative team. Extreme Programming implements a simple, yet effective environment enabling teams to become highly productive. The team self-organizes around the problem to solve it as efficiently as possible.  
Extreme Programming improves a software project in five essential ways; communication, simplicity, feedback, respect, and courage. Extreme Programmers constantly communicate with their customers and fellow programmers. They keep their design simple and clean. They get feedback by testing their software starting on day one. They deliver the system to the customers as early as possible and implement changes as suggested. Every small success deepens their respect for the unique contributions of each and every team member. With this foundation Extreme Programmers are able to courageously respond to changing requirements and technology.  
The most surprising aspect of Extreme Programming is its [simple rules](http://www.extremeprogramming.org/rules.html). Extreme Programming is a lot like a jig saw puzzle. There are many small pieces. Individually the pieces

make no sense, but when combined together a complete picture can be seen. The rules may seem awkward and perhaps even naive at first, but are based on sound [values](http://www.extremeprogramming.org/values.html) and principles.  
Our rules set expectations between team members but are not the end goal themselves. You will come to realize these rules define an environment that promotes team collaboration and empowerment, that is your goal. Once achieved productive teamwork will continue even as rules are changed to fit your company's specific needs.  
This [flow chart](http://www.extremeprogramming.org/map/project.html) shows how Extreme Programming's rules work together. Customers enjoy being partners in the software process, developers actively contribute regardless of experience level, and managers concentrate on communication and relationships. Unproductive activities have been trimmed to reduce costs and frustration of everyone involved.  
Take a [guided tour](http://www.agile-process.org/) of Extreme Programming by following the trail of little buttons, starting here.

[What is an Agile process?](http://www.agile-process.org/)

# 2.0 – Agile Manifesto

## 2.1 – Contributors

## 2.2 – Principals

# 3.0 – Agile vs Waterfall

## 3.1 – Advantages

## 3.2 – Disadvantages

# Bibliography

Gonçalves, L. (2019). What Is Agile Methodology. [online] luis-goncalves. Available at: https://luis-goncalves.com/what-is-agile-methodology/ [Accessed 14 Dec. 2019].