|  |  |
| --- | --- |
| **Module Name:** | Software Architecture |
| **Module Ref:** | 5N2772 |
| **Assessment:** | Skills Demonstration 2 |
| **Author:** | YourName |
| **Date:** | ThisDate |

Table of Contents

[Table of Contents 2](#_Toc50710273)

[Declaration of Authenticity 3](#_Toc50710274)

[History of Agile 4](#_Toc50710275)

[Scrum 4](#_Toc50710276)

[Kanban 4](#_Toc50710277)

[DSDM 4](#_Toc50710278)

[XP 4](#_Toc50710279)

[Their use together 4](#_Toc50710280)

[DevOps 4](#_Toc50710281)

[Agile Manifesto 5](#_Toc50710282)

[Background 5](#_Toc50710283)

[Main Contributors 5](#_Toc50710284)

[Four Pillars 5](#_Toc50710285)

[Twelve Principles 5](#_Toc50710286)

[Agile over Waterfall? 5](#_Toc50710287)

[Bibliography 6](#_Toc50710288)

# Declaration of Authenticity

I confirm that the following report is my own work and does not include any work completed by anyone other than myself. I have completed the work in accordance with instructions given and within the time limits set by my centre. I understand that copying the work of others in whole or in part, either from people in the class or content from the Web will automatically result in exclusion from an award for this module.

Signed:

Date:

# History of Agile

*“Taiichi Ohno (1912-1990) was a prominent Japanese businessman. He is most famous for being the creator of the revolutionary Toyota Production System, which was later called Lean Manufacturing in the United States. He is widely regarded as one of the symbols of the manufacturing resurgence in Japan after the devastation that country suffered during World War II.”*

*Available at:* [*https://www.getvetter.com/posts/159-taiichi-ohno-an-intro-to-the-father-of-lean-manufacturing#:~:text=Taiichi%20Ohno%20(1912%2D1990),Manufacturing%20in%20the%20United%20States*](https://www.getvetter.com/posts/159-taiichi-ohno-an-intro-to-the-father-of-lean-manufacturing#:~:text=Taiichi%20Ohno%20(1912%2D1990),Manufacturing%20in%20the%20United%20States)*.*

Manufacturing industry is on the verge of a major paradigm shift. This shift will take us away from mass production, way beyond lean manufacturing, into a world of Agile Manufacturing.

It is a term applied to an organization that has created the processes, tools, and training to enable it to respond quickly to customer needs and market changes while still controlling costs and quality.

Agile is a tool box that has several things in it

Why do we need to be Agile?

Global competition is intensifying.

Mass markets are fragmenting into niche markets.

Cooperation among companies is becoming necessary, including companies who are in direct competition with each other.

In 1950s Structured programming improves: developing time, clarity and quality by extensive use for: subroutine, block structures, for and while loops

In 1960s waterfall become a successful methodology in some projects and we still using in our day’s waterfall is a sequential (non-iterative) process which is seen as steadily downwards through the phases.

In early 1970s ITERATIVE and INCREMENTAL, idea is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental), allowing to gain knowledge during development of earlier versions.

Early 1920s PROTOTYPING is the active of creating prototyping of software applications, incomplete versions for users to evaluate the design

Late 1980s SPIRAL is a risk-driven process model generator for software projects

Late 1890s V-MODEL is an extension of the waterfall model, the process steps are bent upwards after the coding phases from the typical V shape

In 1990s rapid application DEV put less emphasis on planning and more emphasis on process, adapting and the necessity of adjusting requirements

In 1990s agile methods rise, they implement adaptive planning evolutionary development early delivery, and continues improvement and (encourages) rapid and flexible suspense to change

In 1994s DYNAMIC SUSTEMS DEVELOPMENT METHODS core technics: Timeboxing, MoSCoW, Prototyping, Testing, Worksshop, Modeling, configuration management. Characteristics: iterative and incremental, architecture-centric, Risk-Focused

In 1995s SCRUM, enables teams to sell-organize by encouraging physical co-location or close online collaboration of agile team members, as well as daily face to face communication

In 1996s EXTREME PROGRAMMING or (XP) advocates frequent “releases” in short development cycles, introducing checkpoints for requirements adaptation and CRYSTAL is lightweight adaptable approach with specific tailored set policies, practices, and processes based on unique characteristics

In 1997s FEATURE-DRIVEN DEVELOPMENT or (FDD) purpose is to deliver tangible, working software repeatedly in a timely manner.

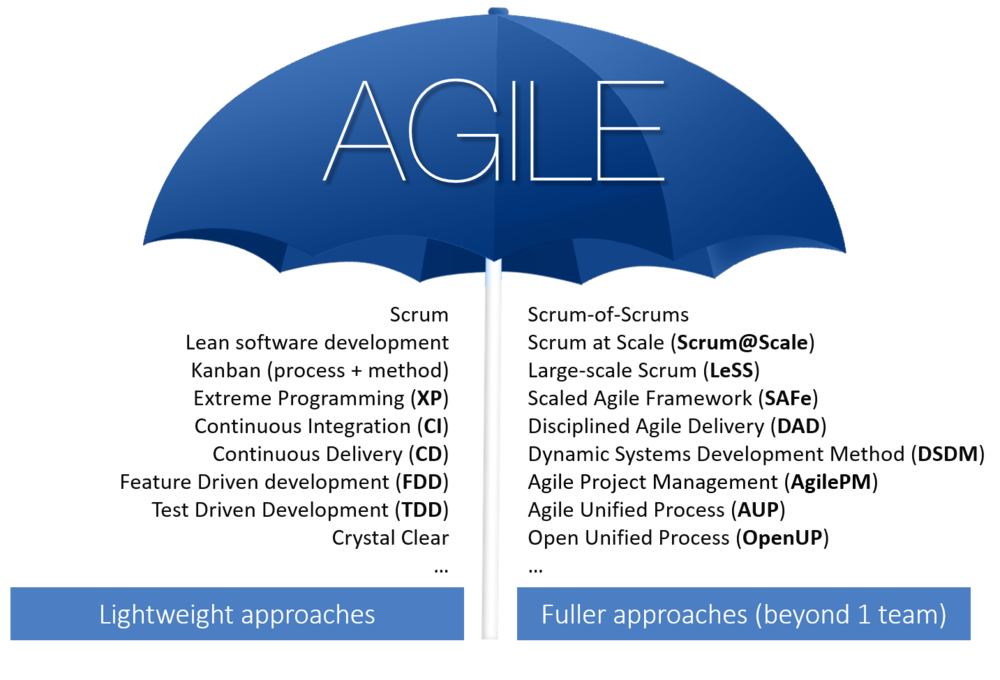
In 2001 MANIFESTO for Agile Software Development

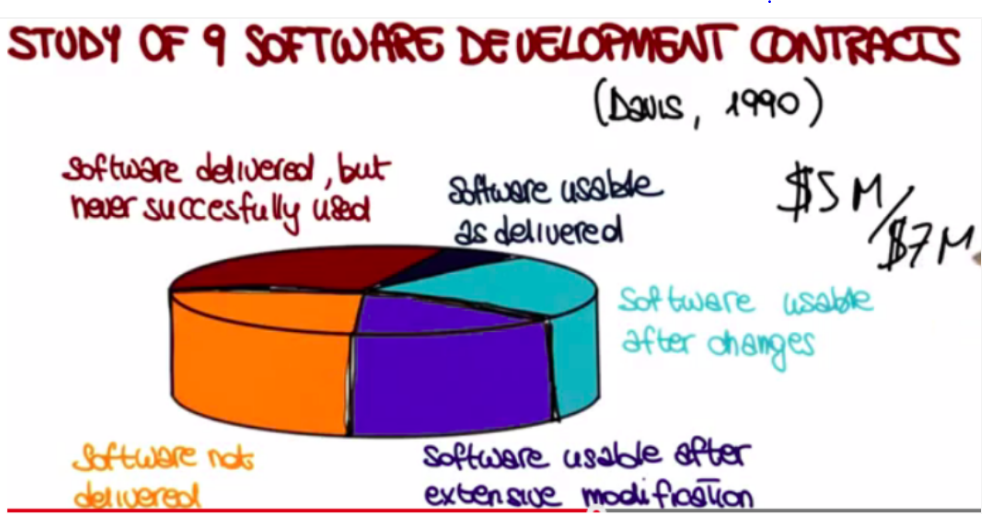
In 2000s AGILE UNFIED PROCESS applies agile techniques including test-driven development (TDD), agile modelling (AM), agile change management and database refactoring to improve productivity

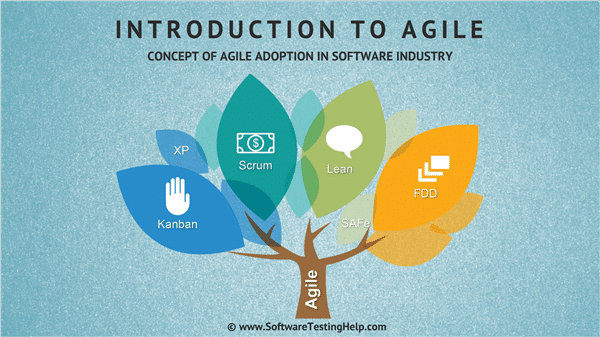
DISCIPLINED AGILE DELIVERY is a process decision framework that enables simplified process decisions around incremental and iterative solution delivery

In 2010 LARGE\_SCALE SCRUM (LeSS) is regular scrum applied to large-scale development for example, for one product group with 500 people, SCALED AGILE FRAMEWORK (SAFe) is a framework consisting of a knowledge-based of integrated patterns intended for enterprise-scale Lean-Agile development

This is Agile umbrella



 Projects went from hundreds of line of code to thousands, tens of thousands of lines of code… to millions of lines of code. That was because of the failures correlating with traditional methods that a new way of approaching software design software design and implementations was needed.



Agile is a set of values and principles has project management methodology, Framework and tools. Agile mind-set helps us deliver valuable software despite uncertainty and risk, this mind-set focuses on Value, Validation, Flexibility, Fast Delivery, Fast Learning, Team, and continuous improvement.

In the traditional methodology of software development first thing we need to do planning and then we do development for the whole project and in the testing in the end and release this method has big problems, the problem is the requirements is unknown from the start (not clear) and development can take very long time and then the release product will reach late and over the budget and sometimes does not do what we want it to do because the requirements has been changed so Agile software development methodology is made to solve these problems to make the development (iterative and incremental) and the planning (adaptive planning) and if something changed something in the requirements we can change that (flexible to change) because we divide up the project to smaller pieces and each piece implements as planning and development then release.

how does a team become Agile? they make their decisions based on Agile values and principles.

agile is really a collection of beliefs that teams can use for making decisions about how to do the work of developing software.

Agile does not make decision for us, instead it gives a foundation for teams to make decision that result in better software development.

When you have a team that is following Agile they will be making hundreds of decisions each week in the way

Agile is a tool box for developing software that box has tools we can 2 or more different things together in the same project and manage a project by braking it up the large portions into smaller functional portions or (stages) that can be developed quickly.

Agile is an idea and to implement that there is more than one way to do that each method has name and a way to implement it.

## Scrum

Complex project can lead to real headaches organization the team changes in scope roles that are not clear but we can change that with scrum and agile framework at its foundation scrum can be applied to any project or product development effort.

It works like that, a product owner creates the product backlog a prioritized wish list during spring planning the team pulls a chunk from the top of that list and decides how to complete it the team has a set timeframe the sprint to complete their work they meet in a daily scrum to keep the work moving forward along the way the scrum master keeps the team focused at the end of the sprint the work should be potentially shippable the team conducts a sprint review on the product and a retrospective on the process then they choose the next chunk of the backlog and the cycle repeats with scrum we can ensure the most valuable work has been completed and by the time the projects ends.

The main goal is to make the software development easier in the end.

Gathering requirements-1, we have to get the requirements from the customer and put them in format like this:

As a [who]

I want [what]

Because [why]

This called a User Story and each user story should be written on a paper and that paper represents a feature that feature will be implemented in the system

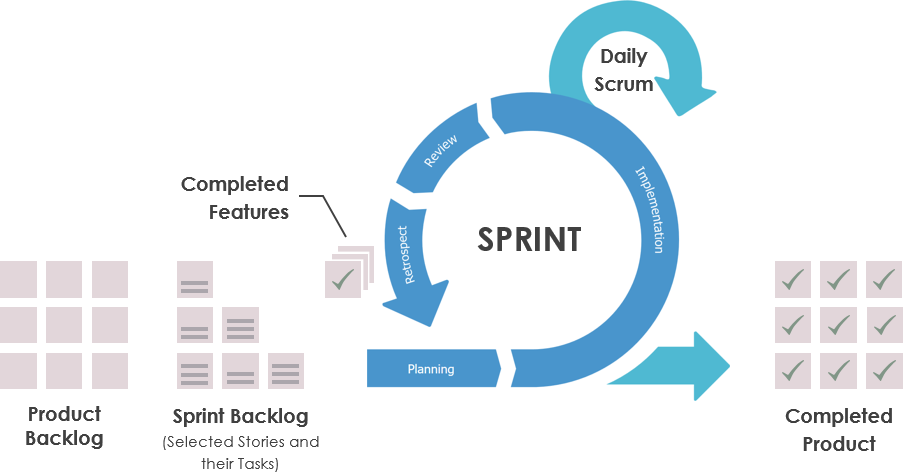
Gathering repuirements-2, When we get gather all the user stories in one place called product backlog and all of these stories in the backlog or we can call that wish list that we wish to implement them in the project.

Team Roles- After we get the requirements we have to divide up the team and there are 2 important position in scrum team first thing is the product owner who represent the interest of the client in the project and that who picks the features and also makes sure everything is going in the right way.

And there is the scrum master who sets up the meetings and who makes sure that everyone is working ok and smoothly with all the team.

So after that we will have to think about the release planning so we pick user stories for next release then we prioritize it as the important thing then the less important and then we give time estimate to each of them (and possible breakup) if that takes long time to smaller user stories.

So we have that done then we will have a Release Backlog and a Time Estimate of how much work to do.



Sprint-1 sprint are short duration milestones from 2 days to 30 days depends on the release cycle, all the team works on a part of the user stories and finish it in the sprint

Sprint-2 when the team finish the first sprint they work on some user stories and get them to a ship-ready state 100% complete, this includes doing all planning, development, and testing needed for the user stories.

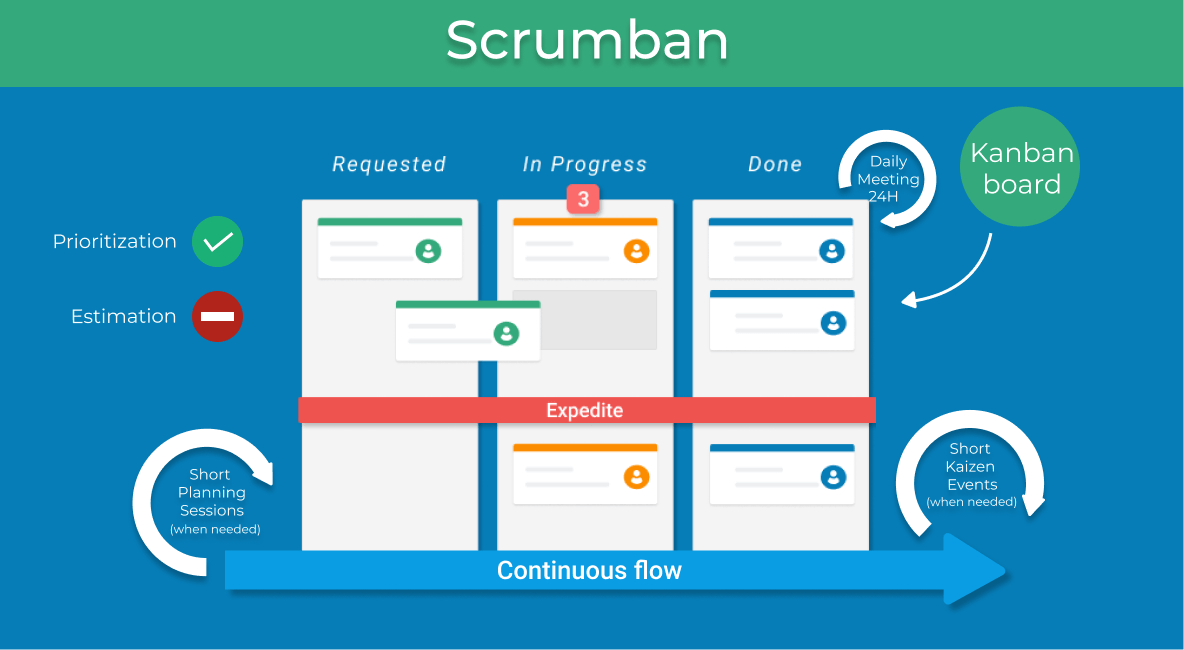
Sprint-3 in the sprint there is daily Scrum meeting, the Scrum master do quick 15-minutes (standing up to be more quickly) meeting everyday: to see what did they do since last meeting and if they are facing any problems that meeting to let all the team know what is happening and what is the other members doing and let the team to help each other solving the problems if anyone has the solution.

Sprint-4 in the end of each sprint the team members get together to do a meeting that called Sprint Retrospective to evaluate their performance in the sprint what was good? And what was bad? And how to improve?

Burn-down Chart that chart represent the time and that let us see every day how much time left for the project, using the Time Estimate, we know the amount of work in the sprint (or release). Update this value every day and plot it on the chart, it helps us to know if we are track or late.



We can use Trello Kanban Board to follow the work and see what done and what is in progressing and what we have to do.



2020. [online] Available at: <https://www.youtube.com/watch?v=TRcReyRYIMg> [Accessed 13 November 2020].

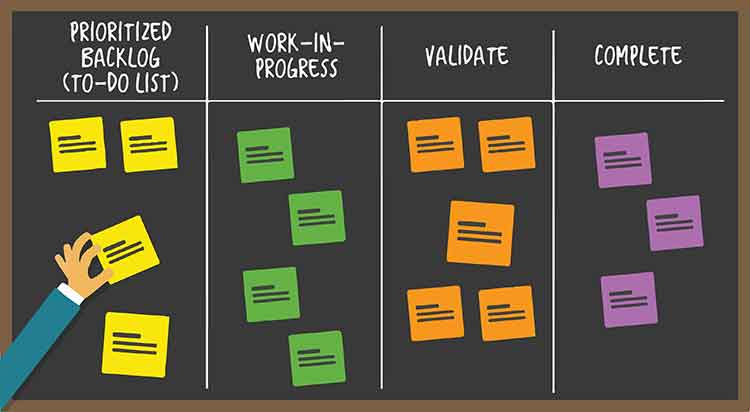
## Kanban

Kanban started out as part of Toyotas just-in production system that system is now known as Lean and word Kanban refers to cards that graphically represent the flow of components and finished products through the production system, it has risen in popularity massively over recent years with the growth and growth of agile project management.

Kanban on its own is a powerful project management process and combined with other agile methodologies like SCRUM can be even more useful in a wider of circumstances

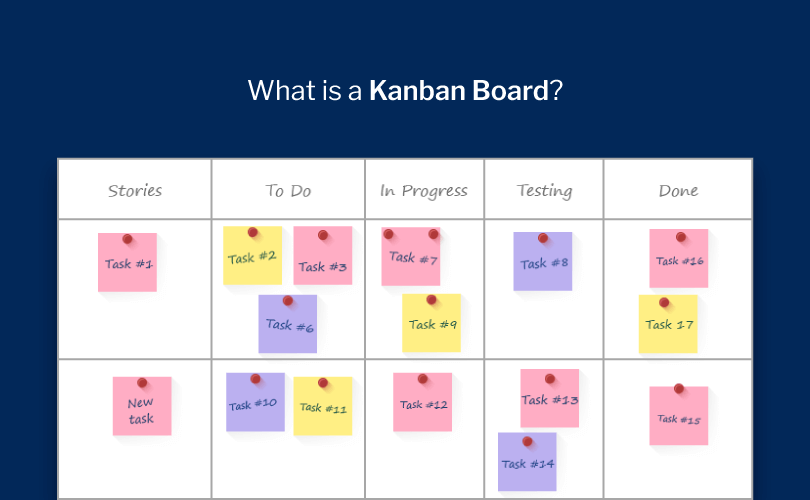
That method was built for continuous delivery, Kanban turns out to be really important in order to use it as a project management technique, rather than just a visualize your work technique.

Kanban was obviously designed in a manufacturing environment, so if we think about manufacturing raw materials go into a pipeline a series of processes take place and a finished deliverable product comes out the end, Kanban does make the position of each requirements in the pipeline very clear and if some are getting stuck so there’s a bottleneck the project management can easily spot it and then they can deploy resources to address the issue, Kanban is a visual system for tracking the progress of work through the project lifecycle it starts with a list of stage which shown as columns on a board these columns are sometimes called (swimlanes) and each unit of work is then represented as a card the team moves these cards from one lane to the next as their progresses team members draw work items into the first active zone as capacity allows.



To be working Kanban first thing for we need to draw the table in order (To Do) => (Doing) => (Done) each one in a column then we get all the requirements on cards so we break them up to smaller pieces then the team start doing the work and move the (product) or cards to be done but the they can start with the easy stuff it will be fine but prefer to start with the most important and the hard stuff. The team give a specific time to have the work done and they see how they do it

*“the idea is that all the “to-do” is mapped out and a board can track what is to*

*do, in progress, being tested and finalised”*

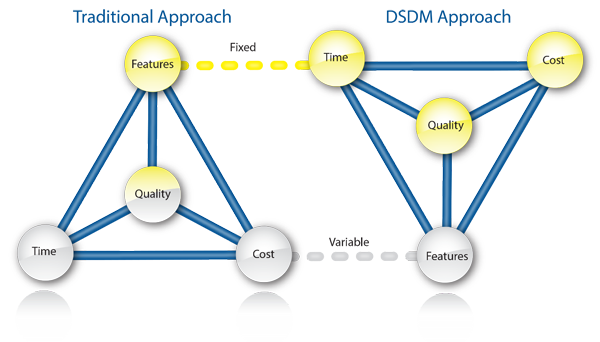
2020. [online] Available at: <https://www.youtube.com/watch?v=CD0y-aU1sXo&t=996s> [Accessed 13 November 2020].

## DSDM

There is increasing pressure on organization to deliver working solutions to business problems and opportunities in ever shorter time scales without compromising quality in 1994 business leaders were frustrated by costly rigid or unreliable software development methods they wanted to adopt a new rapid yet robust method that could instantly transform delivery making projects more responsive while bringing them in on time and on budget as a result the non-profit Dynamic System Development method or (driven strategy delivering more) consortium was formed by a number of organizations from both public and private sectors with the aim to define an industry standard for a new method. In 2001 DSDM helps shape the manifesto for agile software development following which the agile alliance was formed, the agile approach the customer involvement at its heart and has transformed the way teams deliver projects put in quality efficiency and regular early delivery of business values first, the SDM is the longest established fulled project agile approach tried, tested and used extensively for business systems projects and also more widely in non It environments

The DSDM philosophy is the best business value emerges when projects are aligned to clear business goals deliver frequently and involve the collaboration of motivated and empowered people the DSDM agile project framework provides the structure and guidance needed to bring this philosophy to life from understanding the business need through establishing a firm high-level foundation for the project and collaboratively evolving the solution that the business needs to deploying it into live operational use the SDM is clear definition of roles and responsibilities for everybody involved along with proven techniques to help them collaborate effectively underpins the capability of a project team to deliver what the business needs? when it needs it? Predictably and reliably evert time business people are proactively engaged in their project from day one and throughout guiding development of their solution prioritizing the work of the team and ensuring that what is delivered is the best that could be achieved within the budget and the time scale allowed

DSDM differs to the other methods in that is it’s seen as very much an agile project management method area, and some of the other agile concepts like scrum and extreme programming and lean are not really seen as true sort of project management methods that is the main differentiator for the DSDM



The eight principles of DSDM are very important it is really a set of behaviours it sets the culture it sets the tone of the project and the important thing about the eight principles, that the project management has to manage, for example number 3 in the principles which is all about (collaboration) if collaboration breaking down on the project this is what the project management needs to step in and try and stop because it is opening a risk on the project which is going to cause the project damage:

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

Time boxing and Moscow prior are fundamental to SDM they’re very important as well, this is what is going to enable us to get in on time and enable us to stay in control on tow week for week eight-week three-month basis and we use Moscow the way of prioritizing the work as what must be done, should be done and could be done but equally importantly what won’t be done this time and its these two in tandem that are the control mechanism for the DSDM.

2020. [online] Available at: <https://www.youtube.com/watch?v=rIlcaPEGpyg> [Accessed 13 November 2020].

2020. [online] Available at: <https://www.youtube.com/watch?v=cDiGt-l2n2U&t=129s> [Accessed 13 November 2020].

## XP

“*XP is a lightweight methodology for small to medium sized teams developing software in the face of vague or rapidly changing requirements”*

Kent Beck

Extreme programming is a software development methodology to improve software quality it is a short development lifecycle used to increment and improve productivity for the customer requirements, Extreme programming consists of

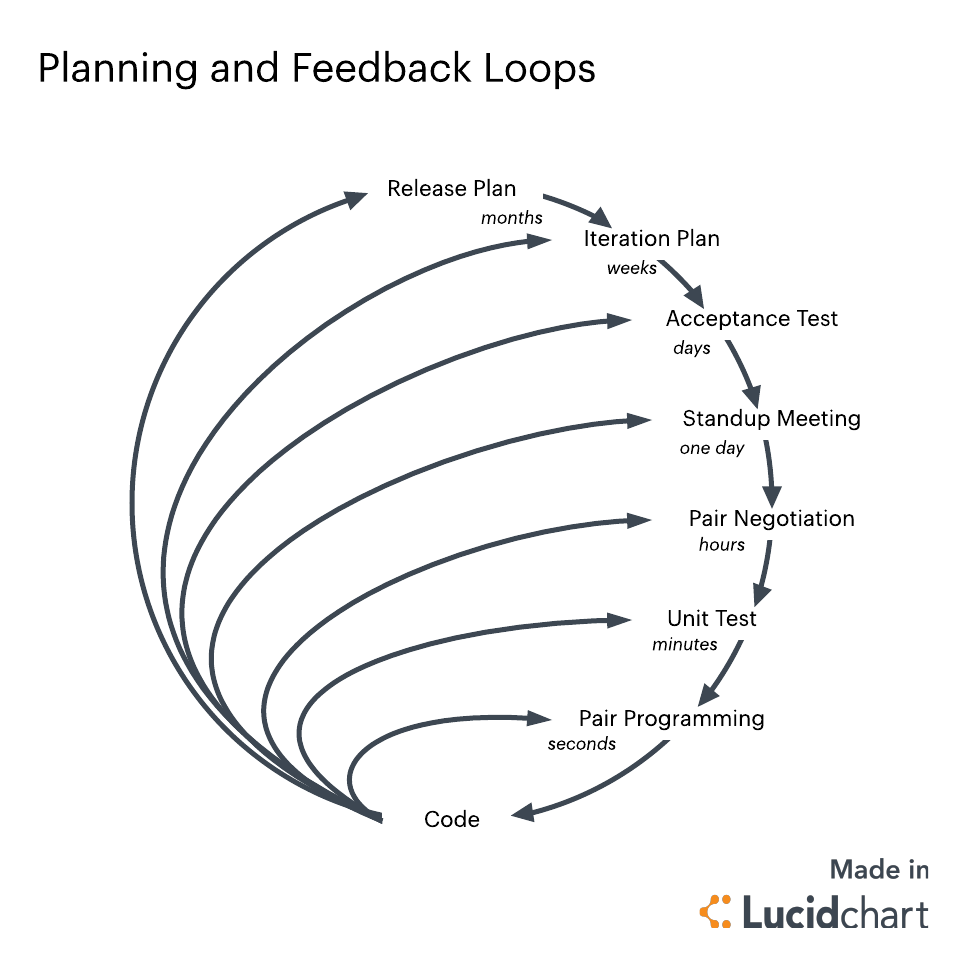
Pair programming: this is a method where two people are working using one computer sitting side by side and have two rows the navigator and the observer, pair programming if done well means less mistake are made and should cost the business less one partner learns from another in the rows reverse, new ideas are encouraging and so better code is produced, it is much less daunting than code review and there is no review team analysing the code

Code review: this is a continuous process of improvement used in creating more efficient and clean code giving the programmer a continuous feedback the developer is to quit, the code is usually well documented which makes it easier for others to carry on the work without slowing down business operation, code review usually includes a review team every so often coming in and analysing the program is cold some argue this is a positive as it means the program is more aware of the bad habits he has and writes more efficient

Unit testing: is a testing method for testing each unit as a smallest testable part it helps to identify failure in the logic of the code produced, the developer will be writing code that is easy to test making the program have a higher number of small focus function which is always a benefit

Integration testing: this is the phase in software testing where individual modules are combined and tested as a group it works as such one part of the software is complete and then tested the second part is complete and both the first and the second are tested and so on this leaves no room for mistakes it is the same as making a movie a new scene is added and you play the movie from the start to see if it is working and expected significantly lowers the risk of failure and make sure zero defect of present

Courage: this comes into play when important decisions to be make and are able to change the direction of the project by discarding incorrect decision the team will usually have to take responsibility and throw away some parts of the code that are not useful.



Continuous process

Design improvement

Comprehensive and insightful explanation of what is meant by XP. and its' main points. Include a graphic that communicates some information about this.

2020. [online] Available at: <https://www.youtube.com/watch?v=kFM2Vcu-BRo&t=12s> [Accessed 13 November 2020].

2020. [online] Available at: <https://www.youtube.com/watch?v=hbFOwqYIOcU&t=25s> [Accessed 13 November 2020].

## DevOps

DevOps is one of those terms where every person has a slightly different definition. For example, Amazon says: “DevOps is the combination of cultural philosophies, practice, and tools that increases a reorganization’s ability to deliver applications and services at high velocity” this definition really emphasizes that DevOps involves combining a bunch of things that are necessary to deliver value quickly.

Wikipedia says: “DevOps is a software engineering culture and practice that aims at unifying software development and software operation” this definition really focuses on breaking down the barriers between development and operations.

DevOps is what happens when everyone in a software delivery pipeline takes shared responsibility and uses their skills not only to do their own piece of the work, but also to optimize holistically for rapid and safe delivery of the final result.

An organization with a DevOps mind-set is going to use their programming skills to automate the deployment process where it can be made more efficient.

They are going to use their skills in operations to make sure that developers and users have easy access to testing environment and servers so automated and manual tests can provide feedback quickly.

A DevOps approach to change management is going put people from change management side by side with developers and operations to find ways to leverage automation to help guarantee quality.

There are all kinds of tools and technologies to enable the automation and collaboration used by DevOps organizations, but none of these things are what is important. What is important, is the way DevOps teams not only produce software but also use their skill to collaborate with everyone else to create an improve a comprehensive, integrated system to deliver that software.

2020. [online] Available at: <https://www.youtube.com/watch?v=aFWi8ToAjpU> [Accessed 13 November 2020].

# Agile Manifesto

## Background

Blkn jrwnhiujrw iujer nhiunh iu nheriuhb iuhiugoiu

The actual definition of Agile is found in the Agile Manifesto, the Agile Manifesto makes it clear that Agile is not a methodology or specific way of developing Software and not framework or process.

## Main Contributors

Most organizations that produce software want to say they are practicing Agile Development; to say we are practicing Agile Development we need to striving to follow the Agile Manifesto. The agile Manifesto is a collection of 4 values and 12 principles.

Note: important to note that these values and principles do not specify a methodology, both are very general.

The principles simply come out of trying to follow the Agile Values. There are all kind of procedures and methodologies people use to try to follow the Agile principles and values, but if teams start focusing on those more than the actual values and principles, then they are missing the point of Agile. they cannot follow an “Agile Methodology” to become Agile. they have to follow the values and Principles first and foremost. Perhaps a methodology will help, but if they forget (why) they are following that methodology, they are not following Agile.

There is a story about a young prince from northern India who married a young woman. Unlike some royal unions, they were obviously deeply in love with each other. After a year and a half of marriage the young woman was stung by a poisonous insect and tragically died. The prince was heartbroken and lay at the foot of her bed for two days without leaving. Eventually he ordered her body to be put in a silver coffin and placed in a sarcophagus made from alabaster. He then dedicated his life to building a monumental building around her body to remember their love. Year after year he put all his effort into improving the structure and making it more and more beautiful. He was constantly looking for ways to improve every aspect of the building that had his queen at the centre. After spending his life directing the work and rework of every aspect of the huge monument he kept having the nagging feeling that something was not quite right he knew that there was some change that would bring the magnificent monument closer to perfection. After years of making new additions and removing them in research of just the right thing to make it better, he pointed his finger at the sarcophagus in the centre of the otherwise perfect building and ordered it removed. This is what happens when teams get so focused on the (way) they are trying to follow Agile that they forget what agile really is. They forget why they are doing all the things they are doing. Successful Agile teams are the ones that understand the importance of the values and principles.

Their practice and methodologies are always merely ways to support the values and principles. Otherwise, like the Prince, they get so focused on what they are doing that they eventually throw out the very reason they are doing it.

## Four Pillars

Individual and interactions over process and tools. During the time of waterfall method, a lot of attention was given to the processes being followed. Waterfall method itself was a 5 step process which had to be followed in order to develop software. Following stringent processes used to make the development process rigid and changes could not be brought easily when required. Additionally, a well-defined process is not the answer to every problem.

No matter how detailed and well defined a process is there will always come some barriers which could not have been anticipated by any process or tool.

Such barriers are overcome when individuals sit together, discuss and find the solution of the problems. Also innovation does not come from rigid processes but it comes from free minds. This is why the agile manifesto says that we should focus more on having competent people working together effectively.

Process and tools should be used but only until they are making things easy and effective for us.

Working software over comprehensive documents, the waterfall method used to start with heavy documentation process. A lot of documentation used to be done before even starting to build the software the features, specifications, layouts, requirements, test cases etc… were documented long before their arrival time.

This approach had some drawbacks like, what you have documented will be outdated by the time you are done with it, you may end up creating many features which may never be used by the end user, the time spent on the documentation of such features also goes in vain.

Too much documentation also delays the project which later becomes an opportunity cost for the business.

Because all of these drawbacks, agile manifesto favours working software over comprehensive documents, instead of wasting a lot of time in creating long documents and then taking feedbacks on the basis of those documents, we should take the feedback on the basis of working software. A base of software with all the basic features should be created first and then it should be given to the end users and stakeholders, then on the basis of the feedback received from the users, further development should be carried out. In this way al lot of time is saved, also the created software has the features which provide values to the end users.

Customer collaboration over contract negotiation, in the traditional method of software development, the customer was involved at the beginning of theproject for control negotiation, one more time if there are any changes to be made in the contract and finally at the end of the project.

The development process was more focussed on what is written in the contract instead of being focussed on what the customer wants. This approach neglected the value of customer feedback during the project. When software is developed without the constant feedback of the customer, the end result is often a product which does not satisfy the needs of the customer completely.

This is why the Agile Manifesto favours customer collaboration over contract negotiation.

Agile emphasises on taking constant feedbacks from the customer throughout the development process so that whatever is being developed is also being seen by the customer. Whenever the customer feels that the projects is going in the wrong direction then he/she can steer the project by providing inputs to the developers.

Respond to change over following plan, Traditional software development methods were not in the favour of change. The traditional approach wat to create detailed plans in the beginning and then following them throughout the project.

Being rigid does not inly lead to the creation of an improper product but it also makes us miss the chances if bringing changes which could make our product better.

This is why agile manifesto favours responding to change over following plans.

When software is developed with agility we start with a plan which give us enough idea of the direction towards we want to move. Then as we proceed and see new events unfolding in front of us we incorporate those realities into the ongoing work.

New events are not seen as obstacles but anything new in our way becomes an opportunity for us to provide additional value

## Twelve Principles:

The first principle our highest priority is to satisfy the customer through early and continuous delivery of valuable software, it is true that takes a lot of time and effort to build a product but at the end of the day what makes a customer happy is a valuable working software this is why this principle emphasizes on early and continuous delivery of the software.

The second principle Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.

Agile believes in empiricism, we become more knowledgeable when we have spent time working on something, in the beginning we may have a lot of knowledge, but not as much as we have after the commencement of work.

Similarly, our customers understand their needs even more clearly after seeing the increment provided by us. This is why this principle welcomes changing requirements from the customers so that the process remains empirical and the customer gets what is required to stay in the competition.

The third principle Deliver Working Software Frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. This principle emphasizes on the frequent delivery of the product as frequent delivery helps the development process to be more agile. With frequent delivery, you mitigate the risks in your release, you get faster feedback from the customer and hence you can bring changes in the product before it is too late.

The fourth principle Business people and developers must work together daily throughout the project. This principle helps in keeping the business aspect and the technical aspect of the product to the same page. When business people and developers work together, they gain a common understanding of the direction, towards which the project is moving the development team also gets an end user view from the business side. It brings transparency and fluency in the whole development process.

The fifth principle Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

This principle emphasizes on having motivated team members. When people are motivated to do their work they produce results which can never be produced by people who are forced to get the job done. Therefore, give them the environment and support they need so that they can create superior products.

The sixth principle the most efficient and effective method of conveying information to and within a development team face-to-face conversation. Most of the communication in a project is about project process or project content. It is crucial for the success of the project that the development team understands the right information correctly. Written communication is prone to ambiguity whereas face to face conversation provide the ground for prompt clarification and quicker communication. As a result, product development becomes faster and precise.

The seventh principle working software is the primary measure of progress. As we know that working software is most valuable for the customer. The planning of the product has no direct benefits for them until a working software is not given in their hands. Therefore, measuring of the project progress should be based on the actual working software. A software is not finished when it is successfully tested and delivered, it is finished when it is tested and accepted by the end user.

The eight principle agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Software development is like running a marathon. You have to maintain the speed but you should not run so fast that you exhaust yourself. Similarly, sustainable development happens when you do constant production of software features during a long lasting period.

The ninth principle, continuous attention to technical excellence and good design enhances agility. Agile focuses on development of software like a craftsmanship rather than just working on a regular task. Continuous attention excellence and good design helps is bringing this craftsmanship to the project.

The tenth principle, Simplicity the art of maximizing the amount of work not done is essential. More features make a project more complex. Complex code is difficult to test, maintain and develop. Sometimes some products have features which are so rarely that their development and maintenance results in a negative ROI.

Therefore, it is better to create a product which does a few things extraordinarily and is completely useful for someone rather than creating a software which tries to do everything, does nothing extraordinarily and is not useful to anyone.

The eleventh principle, the best architectures, requirements, and designs emerge from self-organizing teams. A self-organizing team is one that does not depend on or wait for others to assign work instead, these teams find their own work and manage the associated responsibilities and manage the associated responsibilities and timelines. They take on the responsibility of choosing the most effective and efficient way to complete their work and regularly look for ways to improve through experimentation. While self-organization teams don’t require a manager to assign work, set deadlines, and so on, they do require a mentor who can help grow their skills. Having self-organization teams promotes collaboration, teamwork, competency, regular growth, motivation, and commitment. And hence, the best architecture, requirements, and designs emerge from such teams.

The twelfth principle, at regular intervals, the team reflects on how to become more effective, then tunes and adjust its behaviour accordingly, agile encourages continuous improvements and productivity. When a team reflects on how to become more effective, and adjusts its behaviour accordingly at regular intervals, it brings continuous improvements in itself and increases its productivity.

# Agile over Waterfall?

Agile project management is going to be added to a tailoring process between agile and waterfall how does that work? What are the big three differences?

So if we put together just a little bit of information on the agile versus waterfall and really what the three most impactful differences are because they are very different in fact they are almost completely opposite from each other so how can we take a look at the three big differences and then make decision in our day-to-day about really how to tailor and pick and choose the best practices from each so when we have a look at the three most impactful differences,

What is waterfall? It is sequential process flow now what that means in English is basically we are planning in advance much like waterfall falls from top to bottom we are taking a look at different processes we initiate then we plan then we execute and then we monitor and control before we close everything out chances are scope is pretty well-known in the beginning we’re doing some kind of construction project or were mass producing something we are looking for something that is tangible and because scope of work is so well-known chances are we are better able to pre plan and the execute but in waterfall project management there is a very formal change control system because we go forward thinking this is going to be the scope of work we are planning on it we are dependent upon the requirements that we collected and we collect them fairly early in the process in fact right at the very beginning of planning so we have got the scope of work requirements that way we can better select our durations of time try and figure out what our finish dates are and at the same time looking at the cost estimate for resources equipment material and so on and set an appropriate baseline for the entire project and because of that it’s probably a longer term project maybe a year longer and most waterfall projects end up in something that is tangible not all but most when we take a look at the challenges of waterfall it’s really one of the things where if you’ve ever created something in a project a tangible deliverable and then how the customer come in and sign off on it and validate it at the end and not validated it at the end and then say well no I really wanted this instead of that. That’s very frustrating because if we set things on the front end and we gather the requirements the we are hoping that we’re producing the correct end result does not always work out that way and changes in the middle of this are super difficult in some cases to actually implement because we actually have a set plan another thing about waterfall is heavy documentation, but basically there are a lot of best practicing for documentation it is our job as project managers to pick and choose what documentation we are going to do but because we are heavily pre-planning chances are we are heavily documenting as well now if we get too many scope changes that can actually cancel our project and we would have to start over, if we think about it the customer wants us to mass produce them bicycles and in the middle of the project they decide they want a motorcycle instead that would result in a brand new project.

It does not really allow the flexibility for a big or impactful scope changes so that a lot of times that becomes the challenge in waterfall.

The advantage of waterfall that it allows for departmentalization and control and also can be set with deadlines for each stage of development and the product can go through to the development process model phases one by one.

This consecutive order is suitable for smaller projects which are easy to start and one person could do it like a website.

Advantage: waterfall model uses clear structure if we compare the waterfall model with the other software development methodologies we can clearly see that waterfall model contains a clearest and specific set of steps

Easy to manage and control, early end-goal determination, high visibility, reduced number of problematic issues, the information is well transferred, quality assurance tests.

Simple and easy to understand and use

* Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
* Phases are processed and completed one at a time.
* Works well for smaller projects where requirements are very well understood.
* Clearly defined stages.
* Well understood milestones.
* Easy to arrange tasks.
* Process and results are well documented

The disadvantage for waterfall that is when we reach the end of the project as in testing, it is very difficult to go back and change something or add something that was not well-documented or thought upon the concept stage.

* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
* It is difficult to measure progress within stages.
* Cannot accommodate changing requirements.
* Adjusting scope during the life cycle can end a project.
* Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

The advantages of agile are:

* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
* It is difficult to measure progress within stages.
* Cannot accommodate changing requirements.
* Adjusting scope during the life cycle can end a project.
* Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

The disadvantages of agile are:

* Not suitable for handling complex dependencies.
* More risk of sustainability, maintainability and extensibility.
* An overall plan, an agile leader and agile PM practice is a must without which it will not work.
* Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
* Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
* There is a very high individual dependency, since there is minimum documentation generated.
* Transfer of technology to new team members may be quite challenging due to lack of documentation.

Agile models are preferred where:

* There is a lot of uncertainty
* The scale of the project is quite large
* Complexity is part of the project

Waterfall is when each role passes their deliverables on to the next, so the information architecture person, he may do his wire frames, his or her wire frames, and pass them on the designer and will do the design and then finally, in the next week the product develops it. It is a very linear way of doing things.

Whereas with the agile process, each of these people are doing a little bit every day, so the designer may be designing the buttons while the developer is starting to code a really rough prototype of those buttons and roughly around the week, getting that thing live like feature by feature, by feature… not the whole project at once. At the end of the day there is not really a right way or wrong way to do this. There is some things in our days that are waterfall at one month or depends, but for the most part, we are fairly agile. we are all working iteratively a big buzz word here when we talk about agile, because little by little we are building the thing. Waterfall’s more like, hey I’m going to launch a site, and I am going to launch it eight months from now. But eight months from now change, so people get fired, and ideas change, but with our iterative, agile process, we can change with the project as very flexible

2020. [online] Available at: <https://www.youtube.com/watch?v=TRcReyRYIMg> [Accessed 13 November 2020].

SDLCs, S., 2020. *Software Architecture - Sdlcs*. [online] Sites.google.com. Available at: <https://sites.google.com/view/softwarearchitecture/syllabus/sdlcs> [Accessed 13 November 2020].

2020. [online] Available at: <https://www.youtube.com/watch?v=ygVXGSn2DIQ&t=288s> [Accessed 13 November 2020].

# Bibliography

**DELETE THIS NOTE ------>** Any books, websites, journals etc. that you used and/or mentioned in the report should be listed here in an appropriate format. You should have AT LEAST 5 references!! Link below to reference generator **<----------DELETE THIS NOTE**

|  |
| --- |
| The links under each section below |
|  |
|  |
|  |
|  |