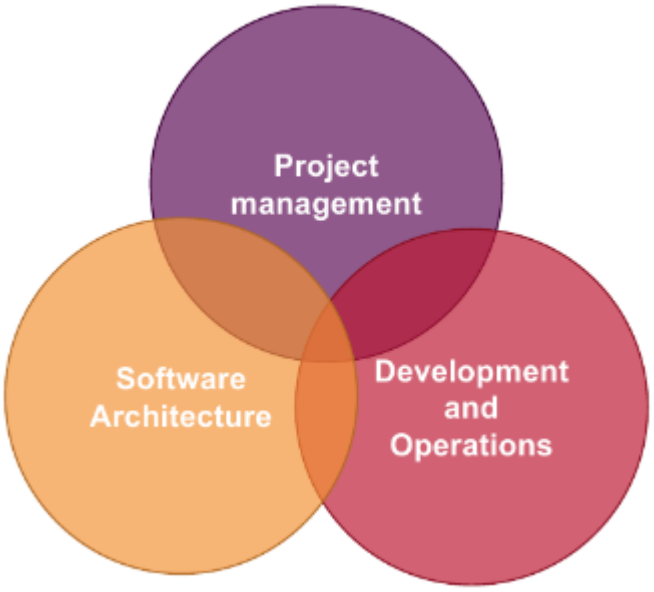
09-Agile Approach

Introduction

Three topics that an enterprise should consider if wants to be competitive:

- Project management: how to organize projects and people working at them → AGILE
- Software Architecture: how to design information systems to better support enterprise processes → MICROSERVICES
- Development and operations: how to design the software lifecycle management process to be effective and fast → DEVOPS



Key points for a successful Information System as you need to manage well people on the right technology. Probably AI will impact less Project Management as it still needs to be a social activity.

High level objectives

Understands concepts
Learn a common vocabulary
Understand something about the professional dynamics that will await you
Have some practice

The era of accelerated transformation

The reality is Volatile, Uncertain, Complex, Ambiguous because of:

- exponential technological innovation
- rapid changes in customer behaviour
- emergence of new competitors ad business models
- unexpected global events
  Small realities can compete with big company(example OpenAI that 5 years ago was only 5 people, but usually the quantity of people that can success in this is very small)

Predefined Processes: efficiency and stability

Context: mazimum operational efficiency, standardization, predictability
Advantages:

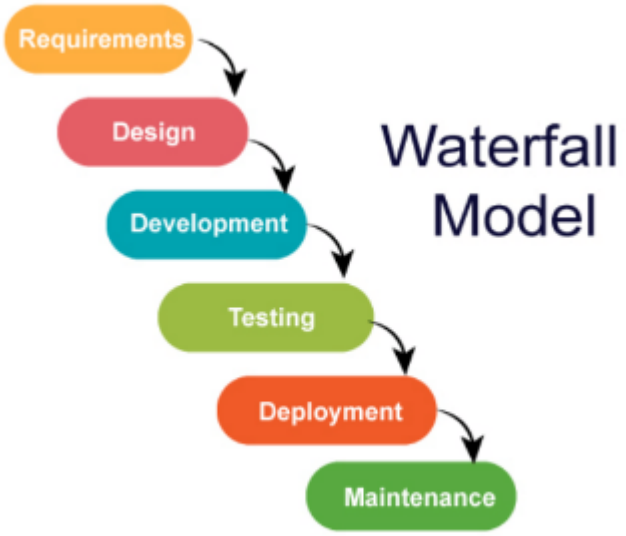
- Cost Optimization
- Quality and process control
- Scalability in stable environments
- Clear roles and procedures
  Disadvantages:

Tries to transfotm our organization from a machine to an organism.

Towards the flexible and adaptive organization

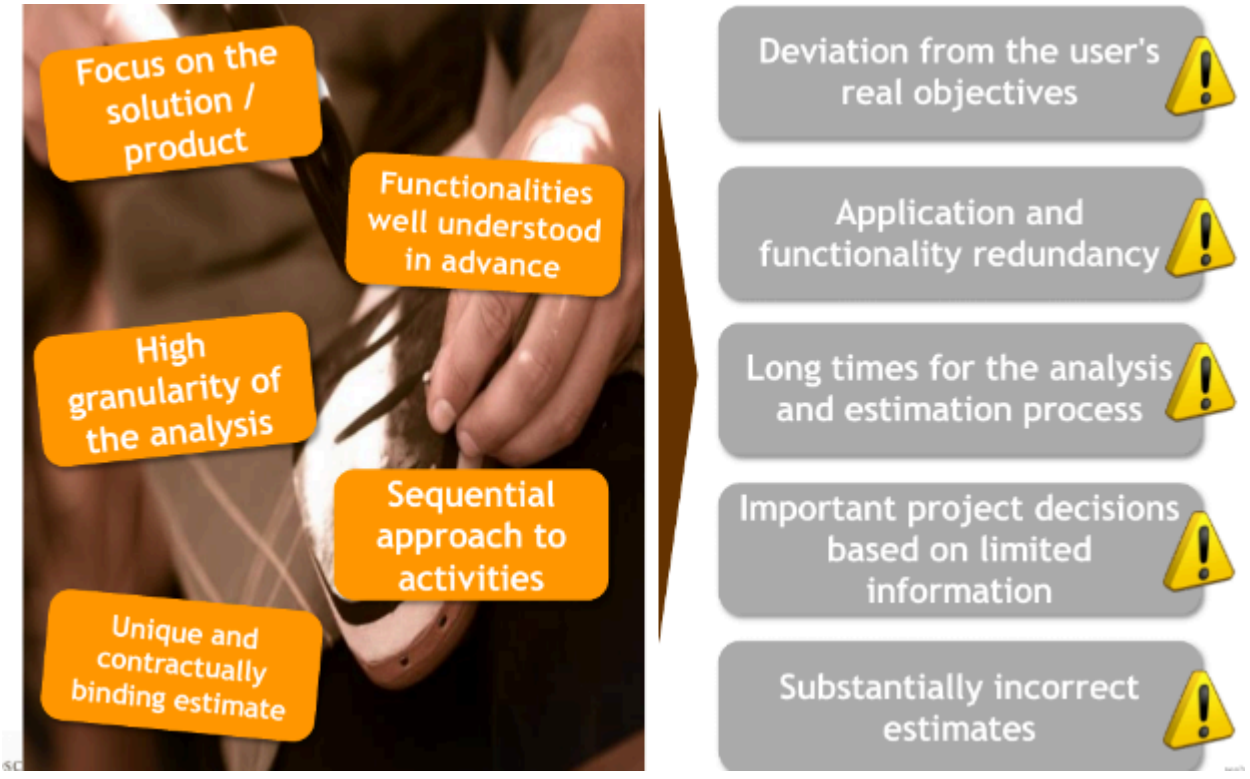
Cut the middle management.
Less hierarchical structure, team can decide all the technical details(other than the common practice as patterns)
Lean, agile and iterative processes. Culture based on trust, transparency and feedback.
Decentralized and faster decision making.

The management of IT projects: the Waterfall model



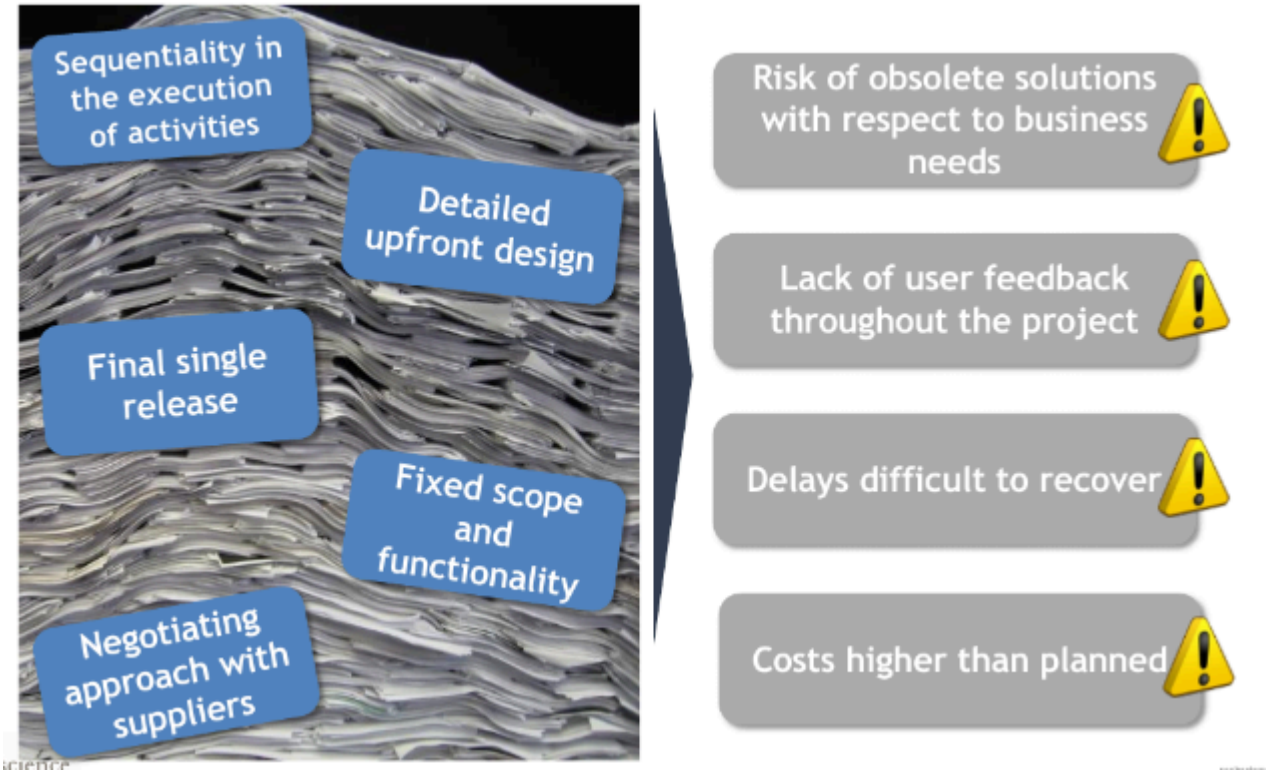
The requirement phase is extensive as it needs to gather all the various requirements of our application
Design phase: decide and develop all the design that the application will need
Development phase: develop the application using the documents created in the requirements and design phase
Testing phase: test the code that was developed, usually is a different team
Deployment phase: deploys the application in a production environment.
Maintenance phase: phase in which you will resolve all the possible problems that will arose from the application running

Waterfall: analysis and planning



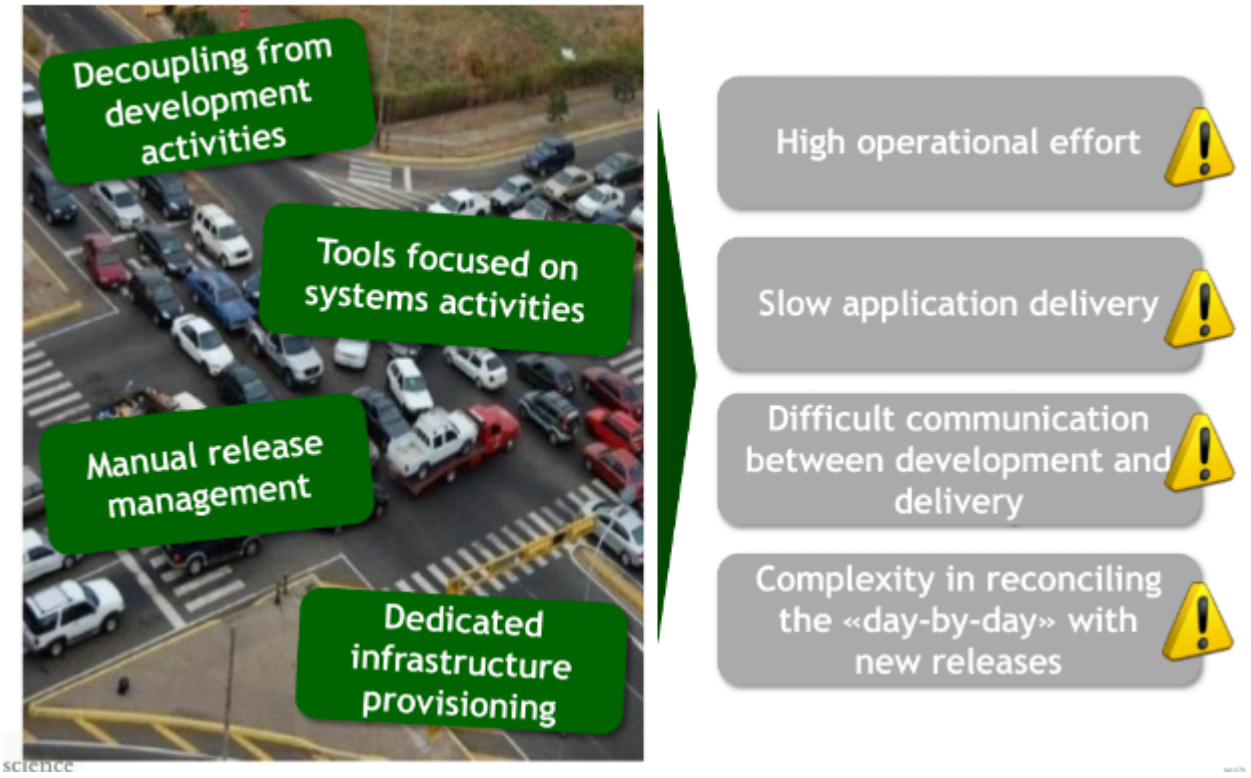
You need to understand all the functionalities in advance with also all the related requirements. So you produce a big document as precise as possible that also needs to survive the problem that will arose at the start of development. You also need to analyze everything as there will not be another analysis phase anymore. Also design need to be correlated to the application, sometimes developer could not understand the design document as it was detached from reality. Also you need to estimate the money needed for the project at the start, but this is no good as basically you will always be wrong. You use a Bottom Up analysis, starting from thinking the cost of development for day, calculate the time needed and then add the margins and all the managerial costs.

Waterfall: design and development



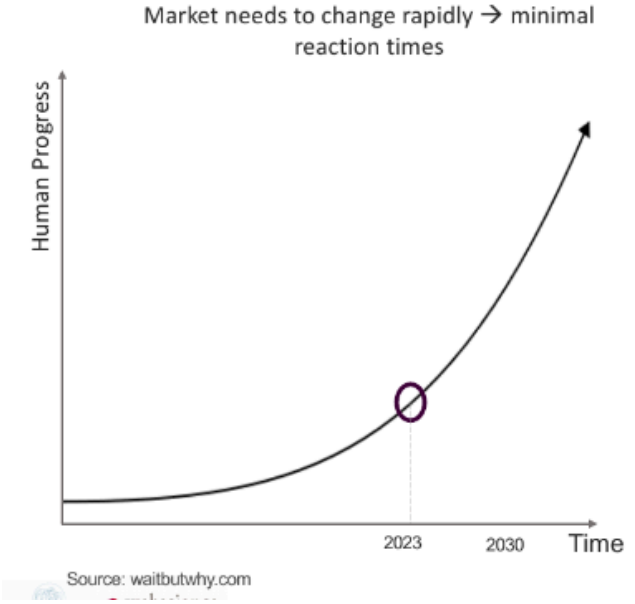
There is an unique and final release. This is bad as the client could be expecting something different from the thing you have understood(late feedback)
Also some functionalities could be not useful anymore and others need to be implemented(requires another contract).

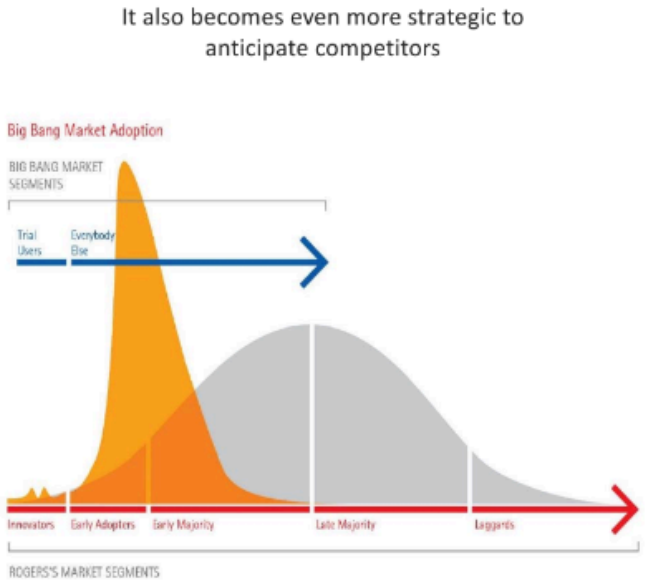
Waterfall: management in operation



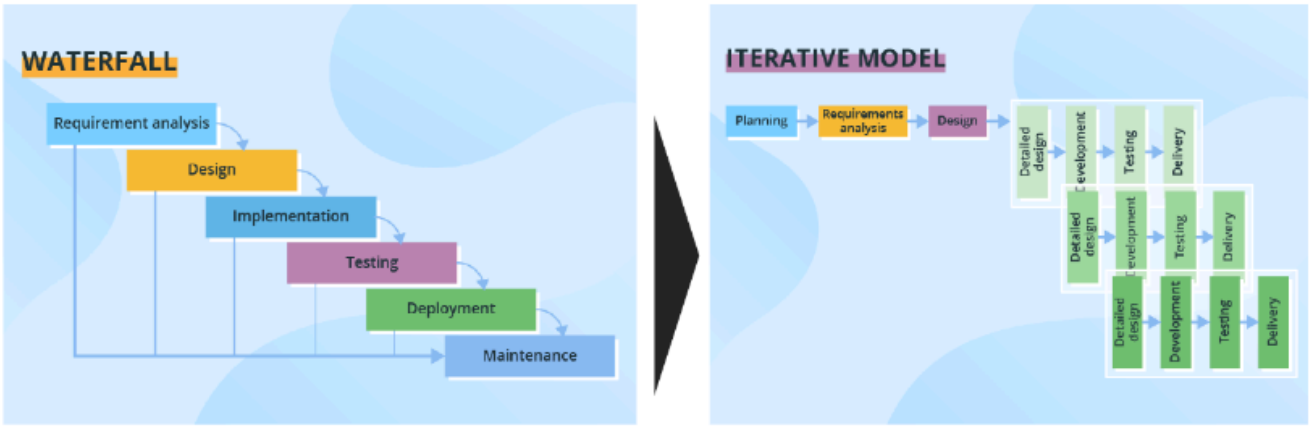
Usually if it is the first time you are producing a big application you will need at least two months. In agile you need less time but also hope that everything works.
You also need to dedicate an infrastructure(in the past more of a pain as you needed to buy all the HW beforehand). Nowadays you create various environment and the production one is the last one.

Why it is necessary to reduce the Time to Market





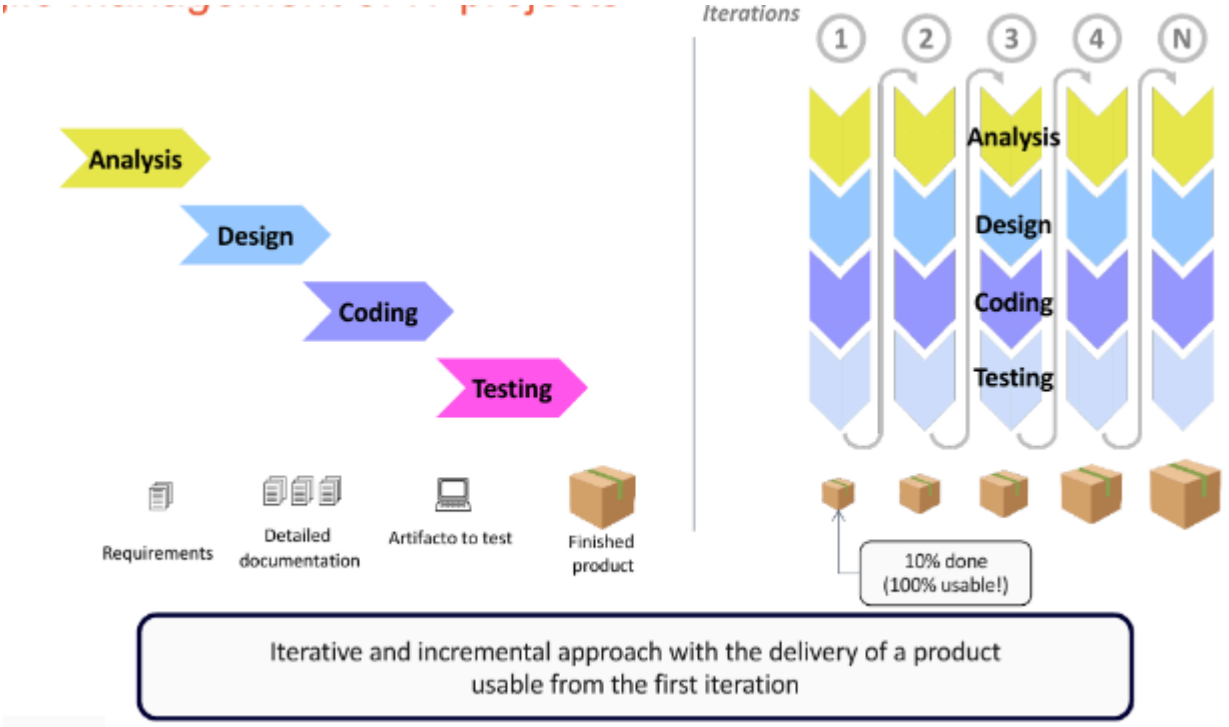
An agility that we can gradually achieve



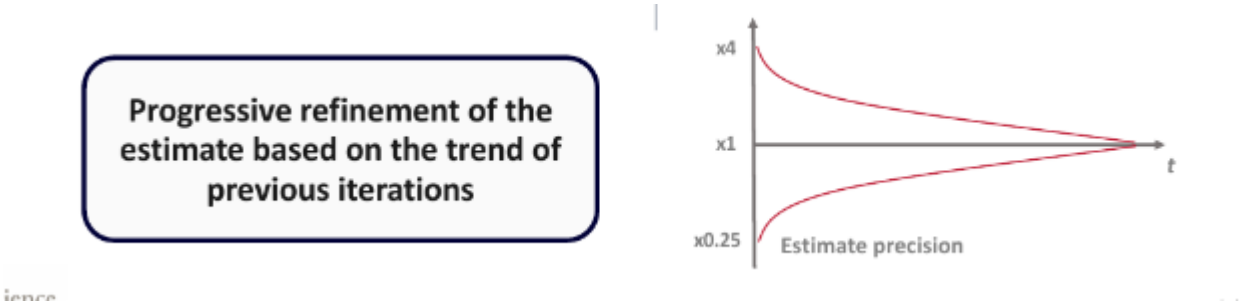
Iterative model still needs some level of organization but you design something designing it thinking about the iteration, you will design only the current iteration as there is no purpose of designing a function before you are going to develop it as there could be changes also in base of your experiences.
You can also fix something at design level(really cheap compared to changing something at design level).
Difficult to use agile to convince the client to deliver the most important things that need to be prioritised in the development process. The price should be calculated dividing the scope in different scopes and calculate the cost for each(need a detail analysis to understand the cost and the planning for each feature).
Garntt diagram is somehow a big lie as they represent what are you gonna do at the start of the project, but they will not represent well the development.

Agile management of IT projects

You can test if an idea is good delivering one iteration of your application to the client and asking feedback. You also start to train your maintenance team on how to answer to problems early. Also easier to adapt to few functionalities and easier to understand how to manage it.



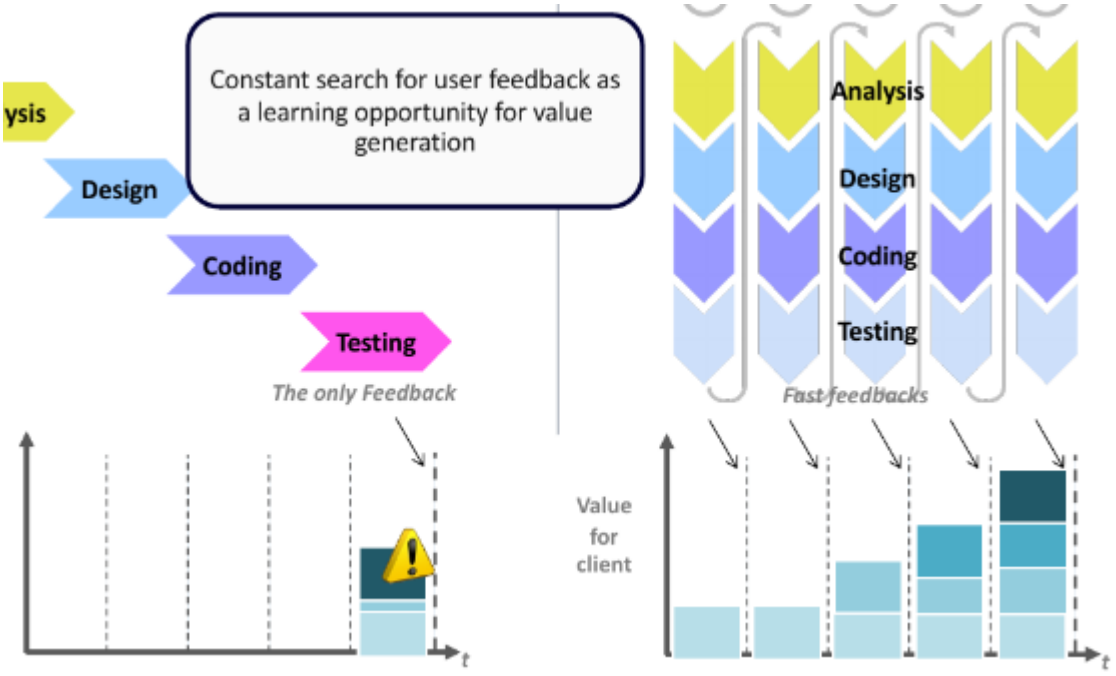
Agile also helps in not accumulate technical depth as you need to resolve problems before each iteration terminate. After each iteration you will deliver something that is production ready. Each team needs to define a DoD(Definition of Done) to decide when the feature can be considered done. Should test every edge case of each functionality.
E2E(End 2 End) test are tests that are doing the work of a final user to understand if the application is behaving correctly in a deployed situation.
Only if you pass each unit test you can consider the feature done, if you are late you still need to go back and pass the tests of the previous feature in the current iteration. Penetration tests are test used to intensively analyze the application. To do these test is usually better to have a different team also deriving from a different company. These tests are black box tests as you need to break the application without knowing how the application work.



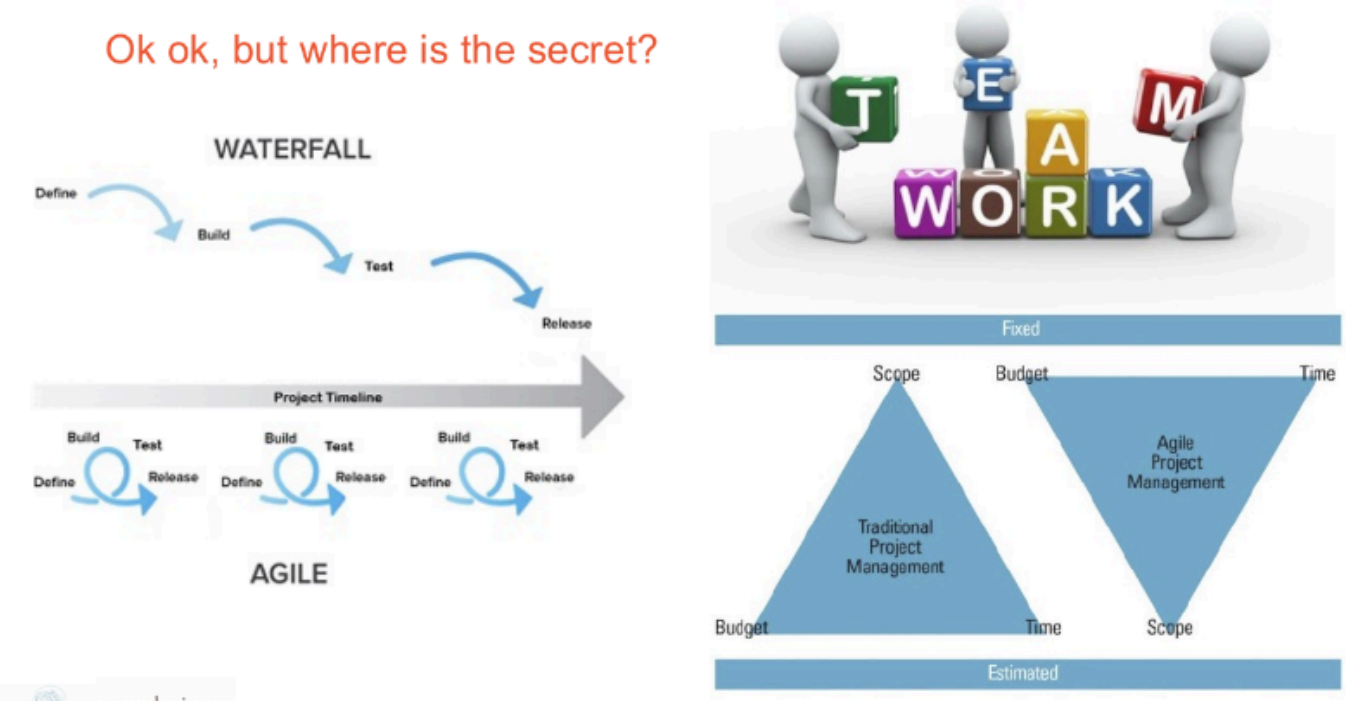
If you want to be clear with the client you should tell them that the cost could skyrocket.
Cone of uncertainty can represent the way you are estimating the cost of your project as the more you go on the more you can understand better where you are going to use your resources.
Also need to take in consideration the sustenaibility of your workload as too much will burnout your employees.
Agile nowadays estimate the cost with poker card estimations. Use a deck of Fibonacci series and see how the different team members see how much times is required and how much it will

cost. From these there can be discussions to better compromise between the different views on the feature.

In SCRUM the product owner will decide the priority of the functionalities. This helps in understanding better the needs of the client.



UAT(User Acceptance Tests): test written with the client and are the tests that the application needs to pass to be considered completed.
Contracts are different between a Waterfall and an Agile one. Do not try to use a methodology when the contract specify another one.



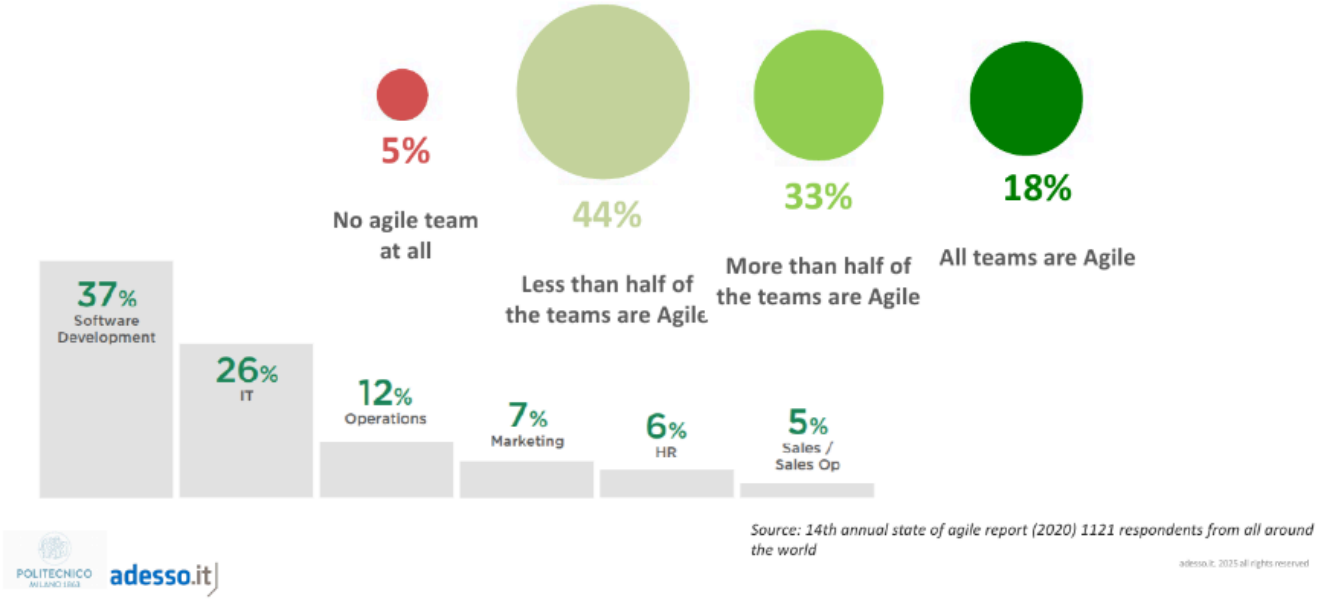
Scope: list of functionalities the application can do.
In waterfall the scope is fixed at contractual level, in agile we define a budget and time but we have the flexibility on the scope. Agile can cope also on the fact that the client changes mind during the development process. No more commitment on the scope only on time and budget.
From financial perspective waterfall projects are easier to manage, so sometimes are still preferred, also used to fix the scope of the project in a contract.
In agile you do not fix totally the scope, as you still want to know the theoretical functions of your application before you put money in it.
Agile also helps in recognizing if some documents are useless to the project as they do not represent the project needs anymore. A bad attitude that could happens is the client do not tell you specific features of the product so they can tell you have done a wrong job(usually in waterfall, always ask for clarification as much as possible).

Agility: the Best-Kept Management Secret On The Planet

- Entrenched in traditional ways of doing business,
- Hierarchical structures and a focus on stability and predictability.,
- Resistant to change due to concerns about disrupting existing processes and risking the status quo.
  This kind of mentality is disrupting the status quo of a company so it was difficult to be accepted inside companies, people find difficulties to adapt to the new mentality.
  Agile let have crystal views of the project, people cannot hide anymore that they do not have the capabilities to do the task required from them.
  Problem can derive from the fact you underestimate the cost and price in the contract to be more competitive, but then you need to cope with the fact that you do not have enough resources. Also can be used to give multiple contract and evaluate each iteration to

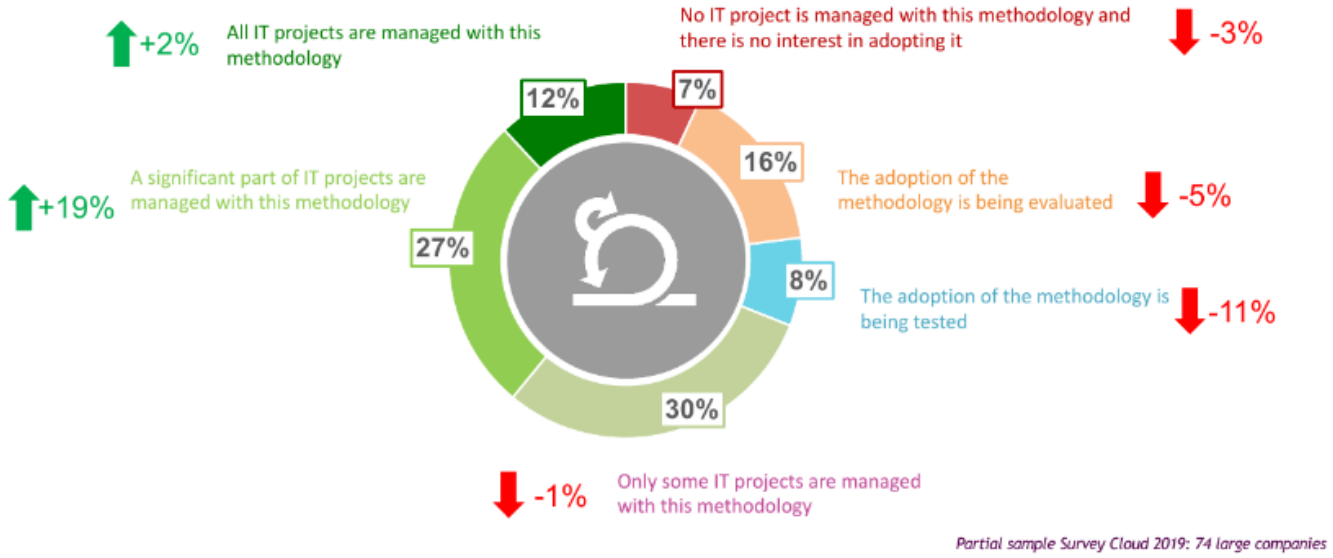
understand the current level of the team. Also easier to sign "small" contracts letting the employer decide if the team was good enough to continue the working relationship with the same team. Underestimating the cost of a project do not let you to hide some problems that you could have inside the team, you need to put everything on the table.

What is the diffusion in the world ?



Data from 5 years ago but proportion between categories aren't changing a lot, many project failed because they cannot adapt to the agile methodology.

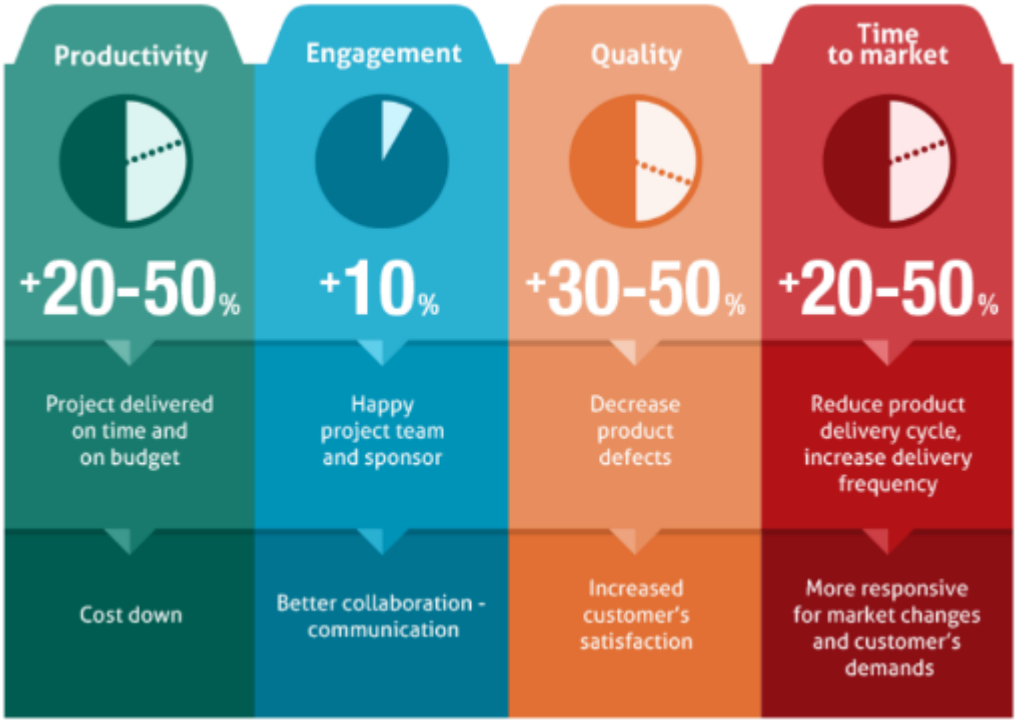
What is the diffusion in Italy



A process of gradual adoption

Done more in a legacy company, nowadays agile projects are 80% of the market, the transitions from waterfall methodology to agile methodology is really slow.

What are the benefits ?



Agile Manifesto

Ideas comes from Toyota as they were able to create a car in 4 years, instead General Motor took 10 years, all thanks to the lean methodology(precursor of the agile methodology).
You need to satisfy your customer, proving at each iteration that you are producing something concrete, having a demo forces the team to produce something production ready.
Always need to welcome changes in requirements as they are the life of the project, no changes means the project will end/die, deadline will be extended, good relationship with your

customer.
Do not deliver the project with a Big Bang style final deployment, but continuously deploy working products.
Business people and developers need to work together, need constant communication and feedback. The project is not only a technical activity but need to take in consideration the feelings of the people working inside it.
Process must be sustainable.
Continuous attention to have excellence and good design enhances agility.
Try to not over engineer stuff, maintain focus on the requirements of your stint, even if you are a good software architect.
Team needs to organize itself completely, form the time to the toolchains. Still need to respect non functional requirements about security, but inside their working they can choose the best design to use for the project.

SCRUM: together for a common goal

Composed by three facts:

- Roles
- Events
- Artifacts

Roles

- Product owner: the person that needs to maximixe the value of the project, accountable for the complete project, very difficult and stressful. Accountability for the maintenance of the product backlog, can change the product requirement and priority between iterations. He commit to a sprint backlog but he can change this backlog during iteration. Usually they are the stakeholder. Usually the product owner should be the client and not someone inside the company, or at least should work with people inside the client company. Cannot be one of the team member as he can try to push to do more than possible for the team capabilities.
- Scrum Master: try to make the people inside the team work the best possible together and act as a bridge between product owner and developers. Usually the person that cover this roles is 100% concentrated to the role. It is more of a social role. Need to understand looking at the velocity of the team if some members have difficulties. With remote working is more difficult to understand the social dynamic of the team. This is why scrum master organize dinners/events to better understand the team situation. **Do not select a developer**

to also be the scrum master as this will impede them to see all the dynamics inside the team as they will have difficulties in taking care of both responsibility. Should also have social/not technical background as it is a role

- Developers: people that work on the project. Usually there are 3 to 9 people and it is crossfunctional. Now ca be up to 90 people, we use LSCRUM, where we have multiple teams of up to 9 people, we have multiple scrum masters that need to coordinate the different teams. The team still needs to create a potentially shippable product increment. **No role or title is recognized within the development team**, can only use the different team member experience to better implement the requirements. Teams are self-organized and are empowered to find the best solution, if there is not trust you should change the team.

Events

- Backlog Refinement
- Sprint Planning
- Sprint
- Daily SCRUM
- Sprint review and Retro

SCRUM Events

- Sprint: iteration of the project, can last from 1 week to 4 weeks. Usually 2 weeks is selected as duration. If the team is not experienced is better sacrifice efficiency for more control, so you have iterations of 1 week so you show the work to the client continuously and receive continuous feedback. Changing the sprint backlog during the iteration should be avoided.
- Sprint planning: create the part of the backlog that will be implemented during the sprint. If not sure you can implement an item you should remove from the sprint. The team can decide if they can implement all the items inside the sprint backlog, but once committed to the sprint backlog they need to implements all the items inside the backlog. Something can happens during the sprint so with the right motivations during the demo you can justify why you do not have done the job completely. Usually takes 4-8 hours and the entire team participate(resistance from legacy manager). Usually divided in two parts: firstly it s discuss which PBIs to put in the sprint backlog and then secondly disccus how to implement the chosen PBIs. You make the information flow from the product owner to the developers. Good investment of time as it will simplify the work afterwards. Also need to take in consideration the capabilities and capacity of the team, can save a little of time for contingency, but not too much as otherwise you will waste time as you cannot ask to implement more stuff inside an iteration.
- Daily scrum: usually implemented at the start of the day and should only take 30 minutes(difficult as they contain a lot of technical information, usually because the scrum master is not present). Several pattern can be used: one is each person nominate three

things(what I have done yesterday, what I will do today and the impediments I found). This helps in putting everything on the table and responsibility is on the table.

- Sprint review and Retro: all results are evaluated against the sprint goals and the product owner can accept/reject the work done, the product owner can ask questions and change requirements for the next iteration. The team is receiving feedback. Only product owner can accept/reject the product, other stakeholders can ask questions. Useful to show what you have done, perceive the progress of the project. Usually take 4 hours usually located at the end of the stint.
- Retrospective: sum up what done in the sprint by the team, can discuss what can be done better, the technology used inside the project and can raise problems in the team, these problems can be put as object in the backlog if they can be critical for the project. The team can prioritize improvement items, but the product owner is the one that needs to accept these items(usually if he doesn't accept none he will pay more for the project).
- Backlog refinement: refine the backlog to identify the items that need attention immediately, sometimes only the senior member of the teams can participate in these meetings, this member can change but only one/few can participate in the meetings.- Product Backlog: estimate User Stories and make them ready following the order of priority. Be sure to put the items in the sprint backlog only when they are ready. Only the team can be decide when a story is done, checklist of quality that a story needs to be considered done. Can remove/add items to the scope to be possible to maintain the deadline possible(this can only be done by the product owner)

Artifacts

- Product Backlog
- Sprint Backlog: take from the product backlog the most important stories, they are described by the product owner to the team
- Final product: all the code and functionalities that are in the product the team deliver to the product owner

User Stories

These stories are functional, describe the functionality from a functional standpoint, no technical data in them to helps the fact that you do not have much technical knowledge at the start of the project. Similar to the requirements, can be non functional stories that are technical but they add a lot of details as you do not have enough information on the product. Describe a business requirement in a concise manner. Need enough details to estimate some items of the product log.