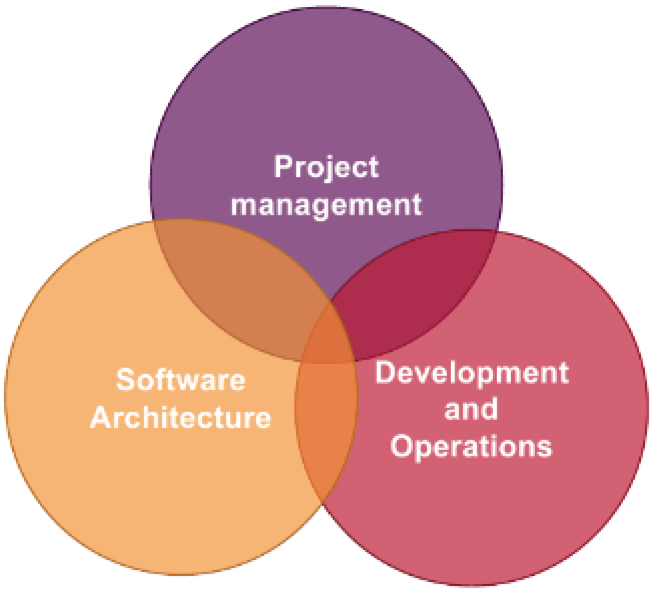
09-Agile Approach

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:

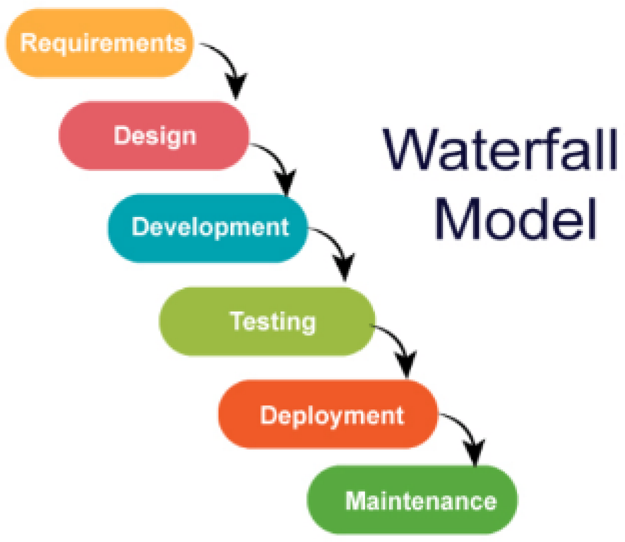
- Cost Optimization
- Quality and process control
- Scalability in stable environments
- Clear roles and procedures
  Disadvantages:

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

Towards the flexible and adaptive organization

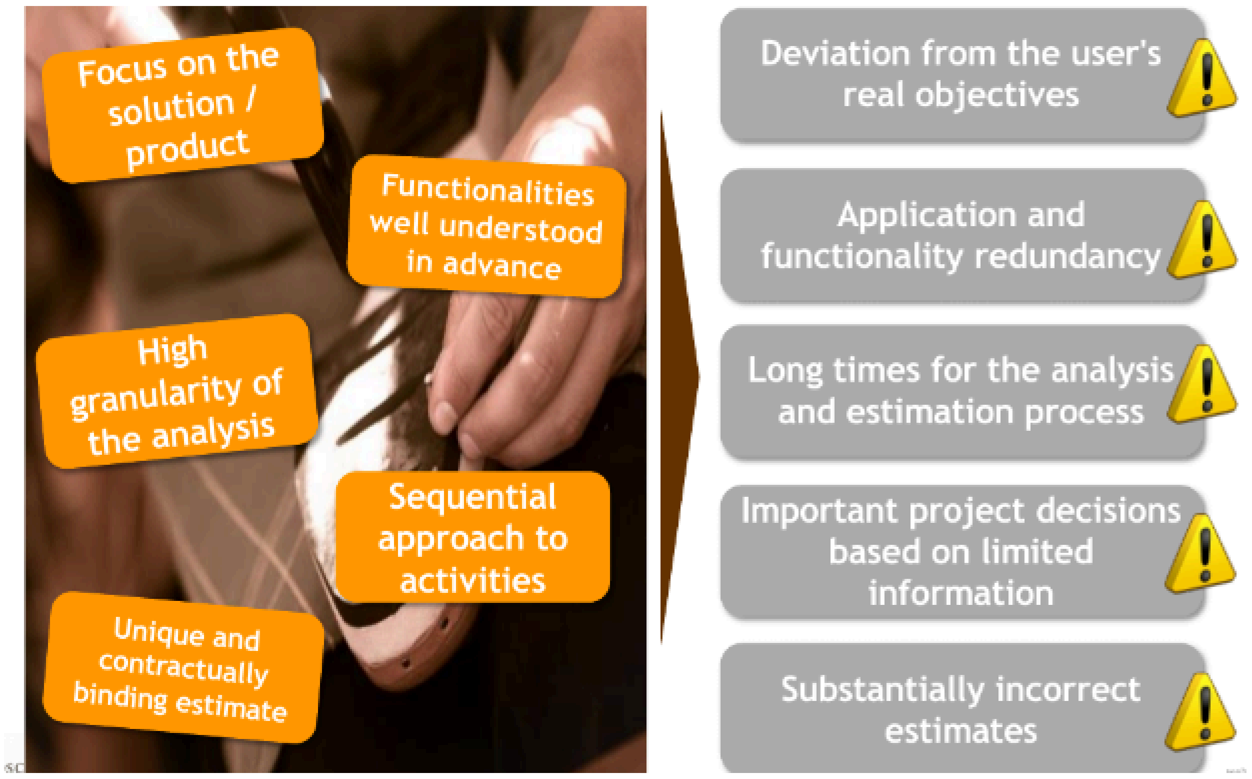
Cut the middle management.
Less hierarchical structure, team can decide all the technical details(other than the common practice as patterns)
Lean, agile and iterative processes. Culture based on trust, transparency and feedback.
Decentralized and faster decision making.

The management of IT projects: the Waterfall model



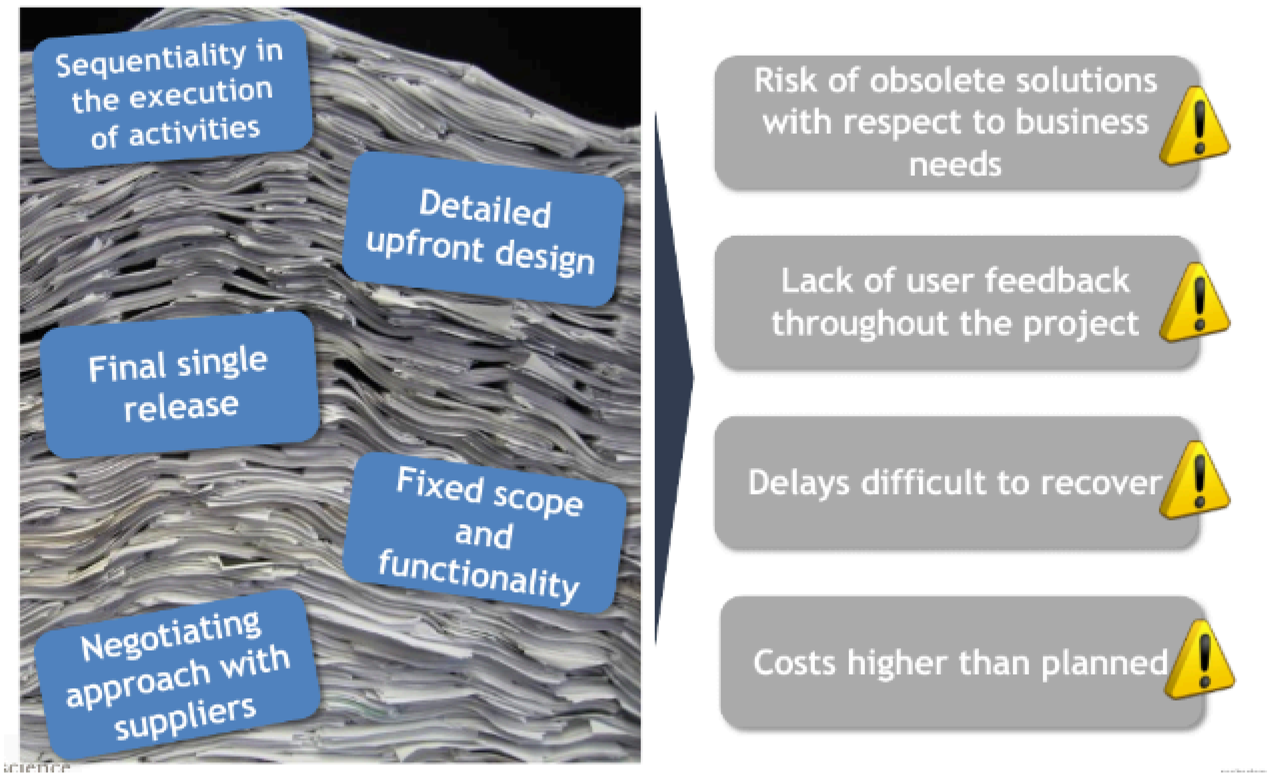
The requirement phase is extensive as it needs to gather all the various requirements of our application
Design phase: decide and develop all the design that the application will need
Development phase: develop the application using the documents created in the requirements and design phase
Testing phase: 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


Focus on the solution / product
Functionalities well understood in advance
High granularity of the analysis
Sequential approach to activities
Unique and contractually binding estimate
Deviation from the user's real objectives
Application and functionality redundancy
Long times for the analysis and estimation process
Important project decisions based on limited information
Substantially incorrect estimates

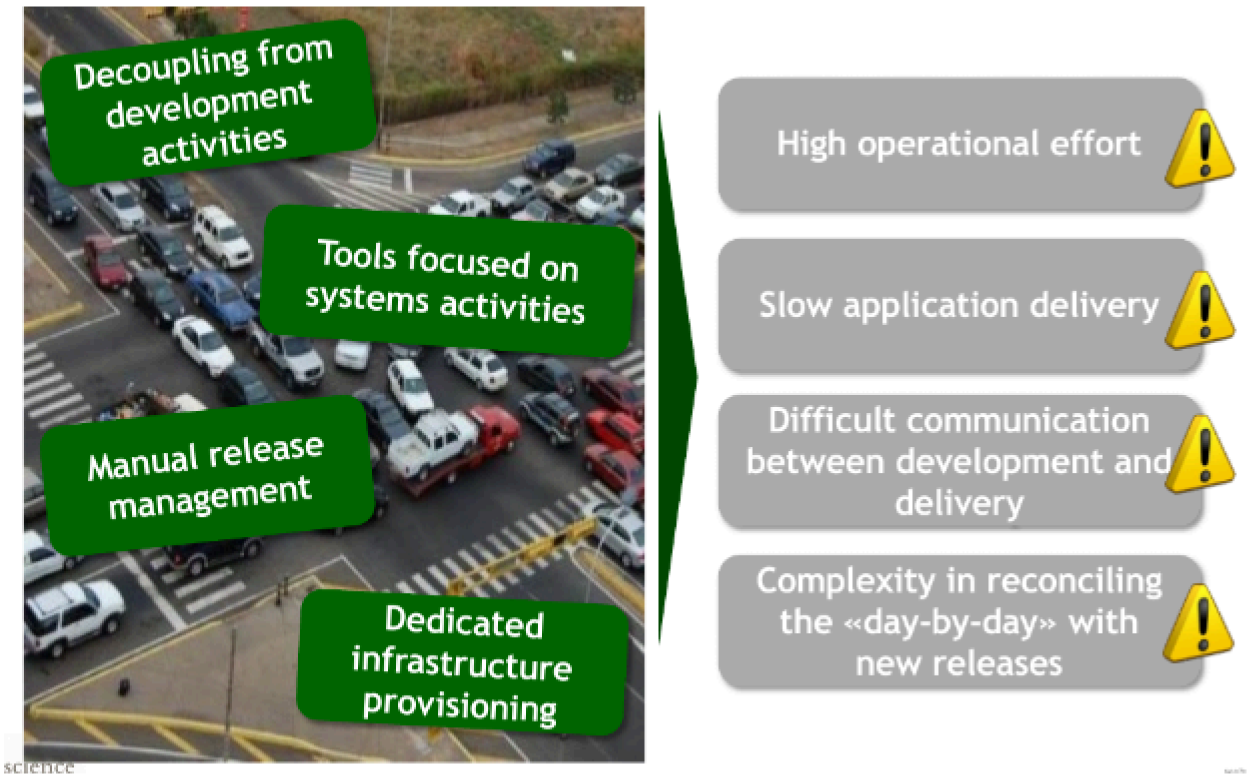
You need to understand all the functionalities in advance with also all the related requirements. So you produce a big document as precise as possible that also needs to survive the problem that will arose at the start of development. You also need to analyze everything as there will not be another analysis 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


Sequentiality in the execution of activities
Detailed upfront design
Final single release
Fixed scope and functionality
Negotiating approach with suppliers
Risk of obsolete solutions with respect to business needs
Lack of user feedback throughout the project
Delays difficult to recover
Costs higher than planned

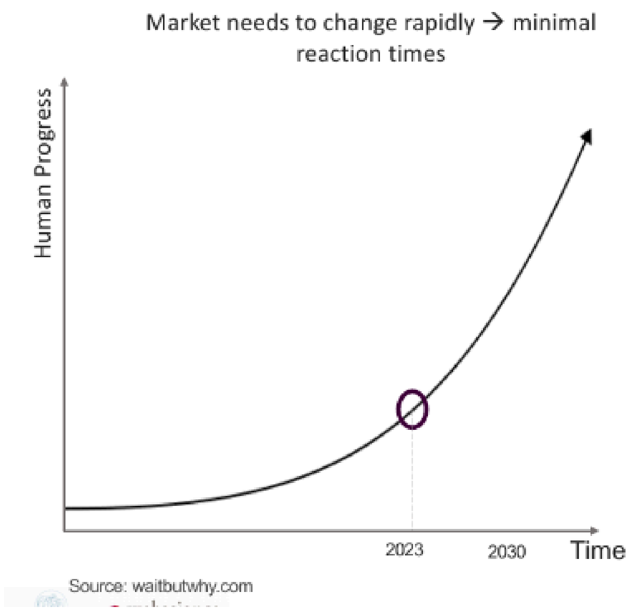
There is an unique and final release. This is bad as the client could be expecting something different from the thing you have understood(late feedback)
Also some functionalities could be not useful anymore and others need to be implemented(requires another contract).

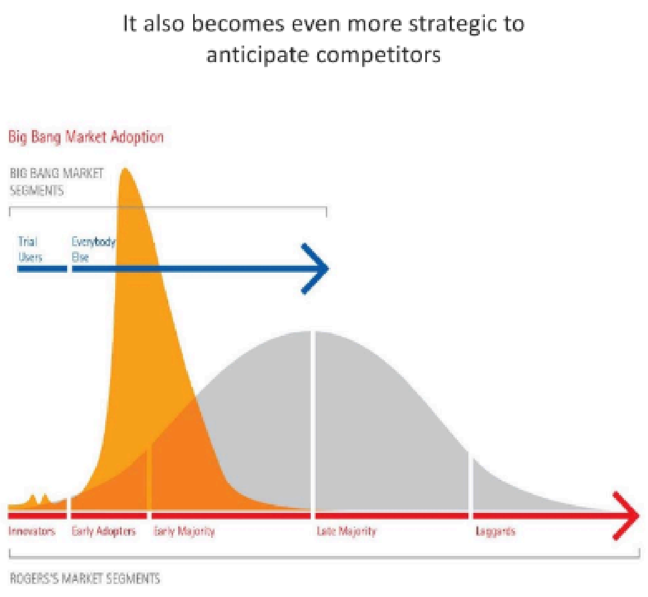
Waterfall: management in operation


Decoupling from development activities
Tools focused on systems activities
Manual release management
Dedicated infrastructure provisioning
High operational effort
Slow application delivery
Difficult communication between development and delivery
Complexity in reconciling the «day-by-day» with new releases
science

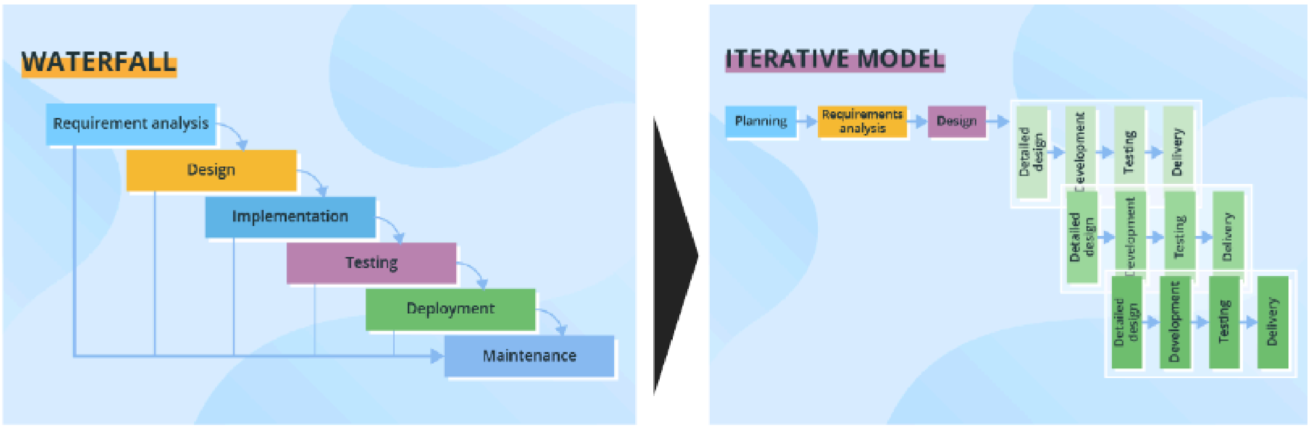
Usually if it is the first time you are producing a big application you will need at least two months. In agile you need less time but also hope that everything works.
You also need to dedicate an infrastructure(in the past more of a pain as you needed to buy all the HW beforehand). Nowadays you create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

Why it is necessary to reduce the Time to Market





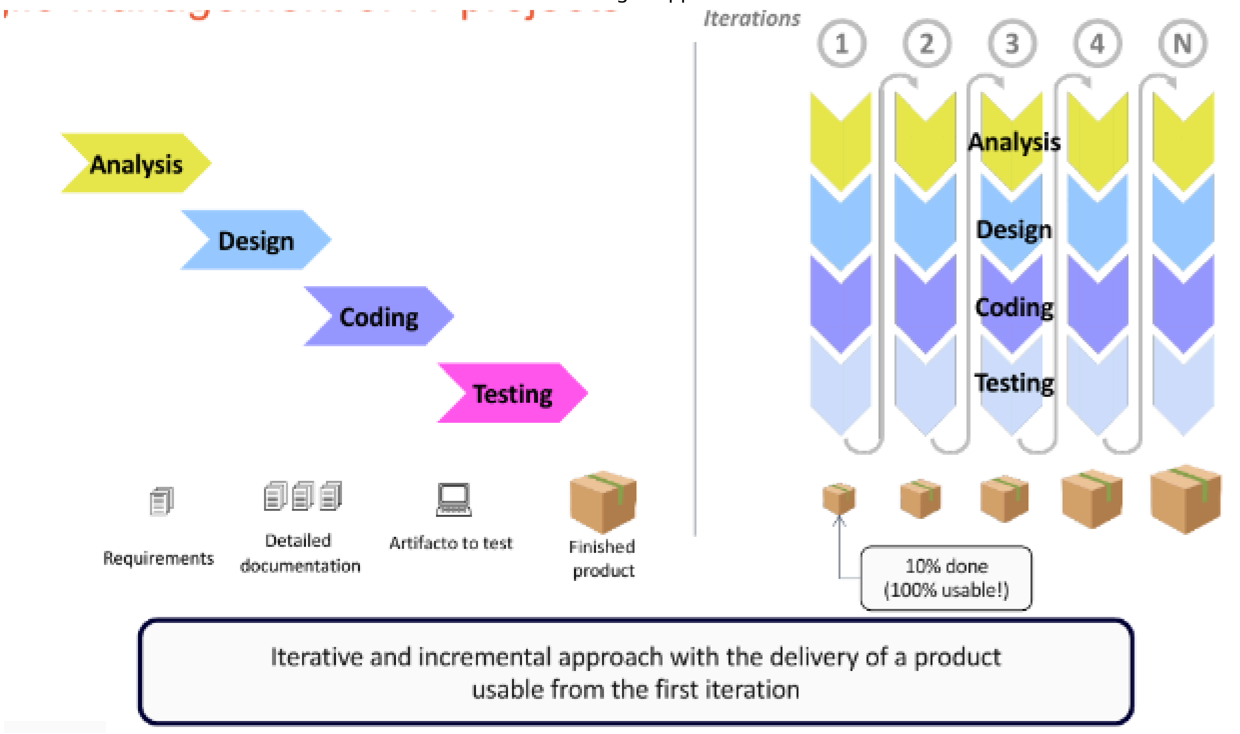
An agility that we can gradually achieve



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.

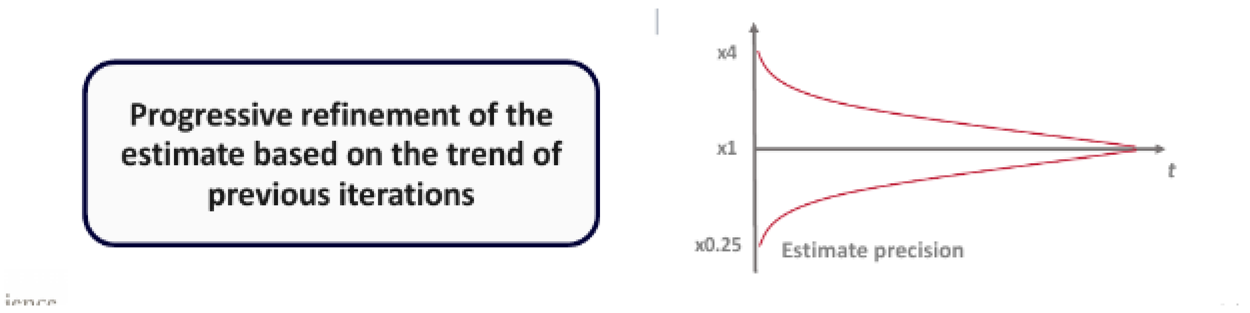
Agile management of IT projects

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.



Iterative and incremental approach with the delivery of a product usable from the first iteration

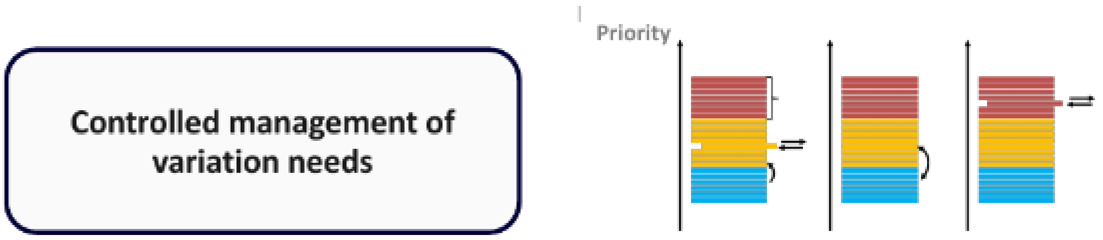
Agile also helps in not accumulate technical depth as you need to resolve problems before each iteration terminate. After each iteration you will deliver something that is production ready. Each team needs to define a DoD(Definition of Done) to decide when the feature can be considered done. Should test 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.



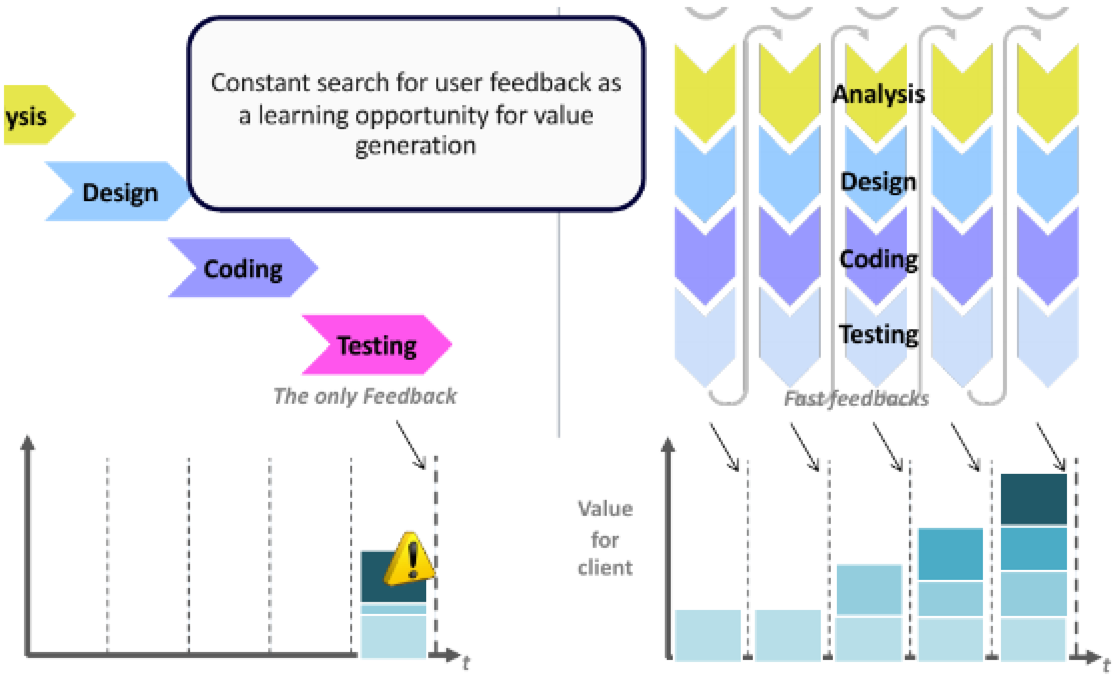
Progressive refinement of the estimate based on the trend of previous iterations

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

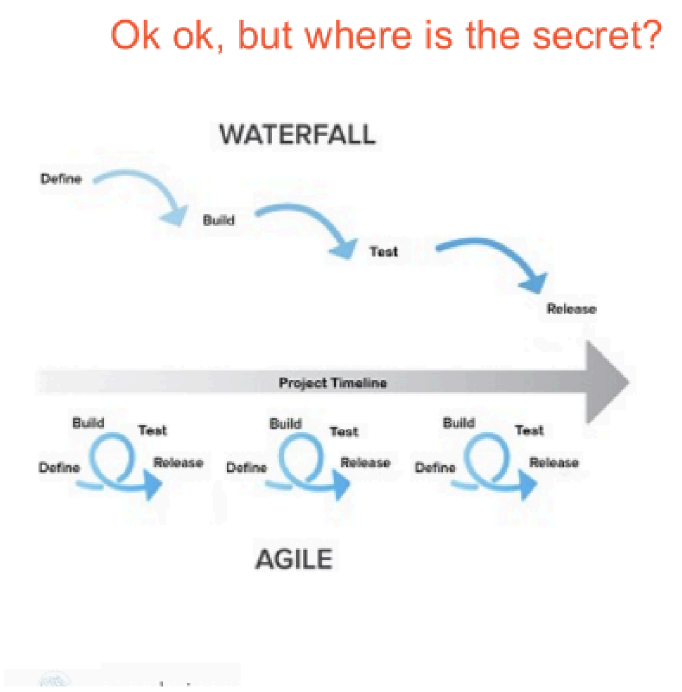
cost. From these there can be discussions to better compromise between the different views on the feature.

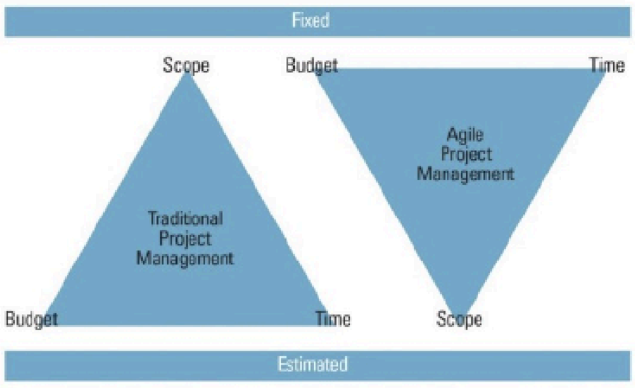


In SCRUM the product owner will decide the priority of the functionalities. This helps in understanding better the needs of the client.



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.