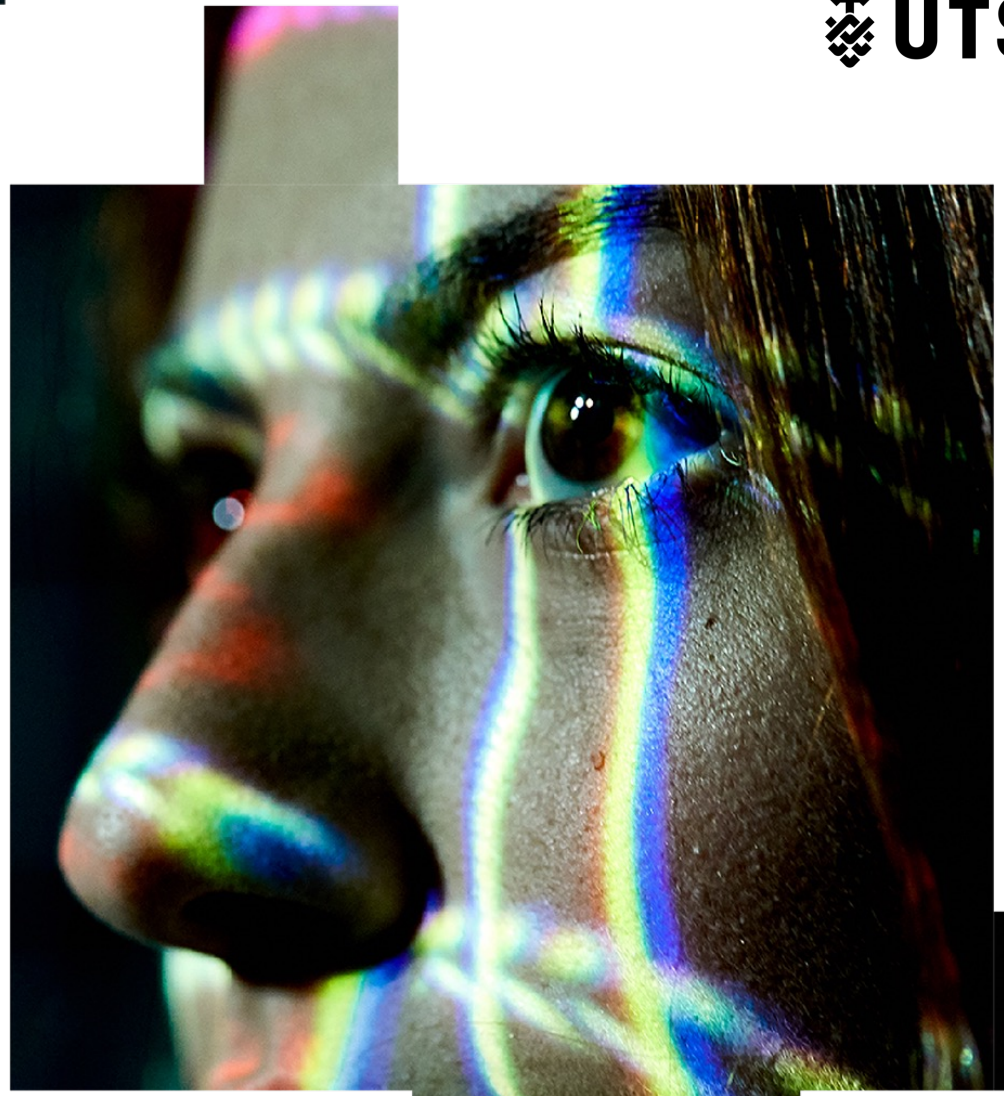
Requirements Engineering (RE)



SSTC 2022
Module 1 - Lecture 2

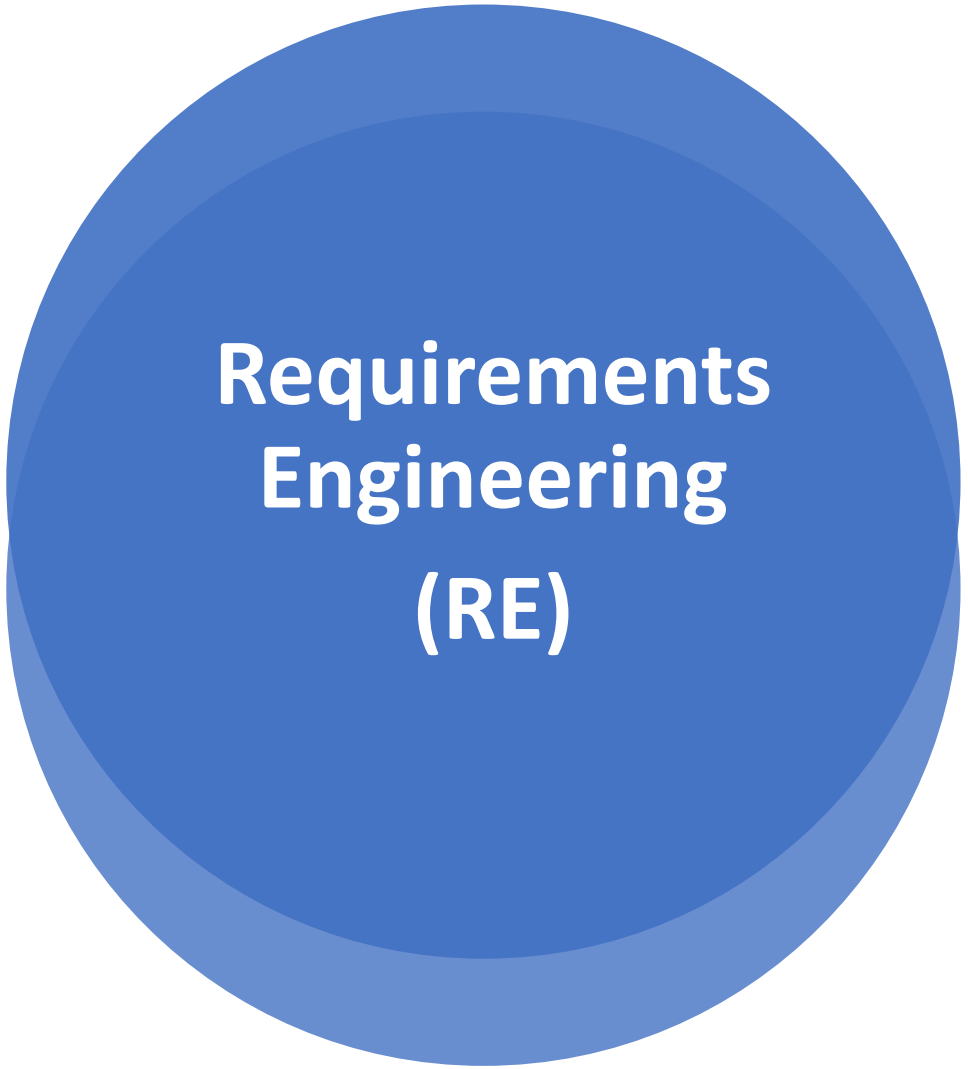
Dr. Salvatore Flavio Pileggi

SalvatoreFlavio.Pileggi@uts.edu.au
https://www.uts.edu.au/staff/salvatoreflavio.pileggi

School of Computer Science, Faculty of Engineering and IT
University of Technology Sydney (Australia)

Learning Outcomes

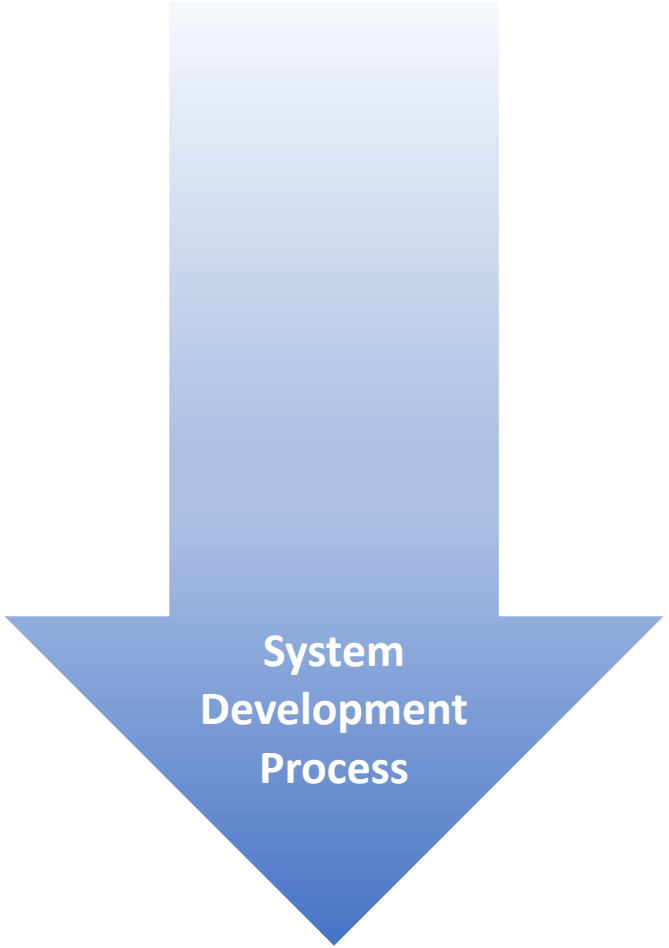
- Requirements Engineering (RE)
- Traditional Methodologies VS Agile



Requirements Engineering (RE)

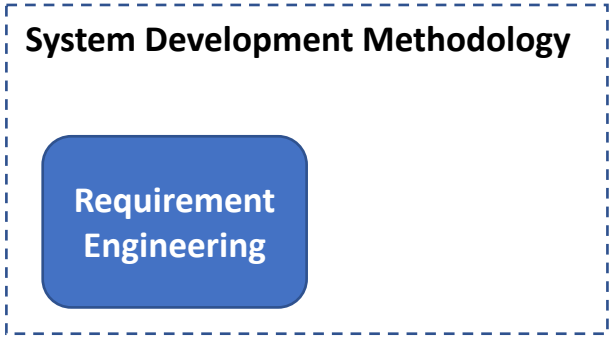
A simple definition …

Requirement Engineering (RE) is the ***process*** of defining, documenting and maintaining requirements



Requirements Engineering (RE)

How do we build requirements?

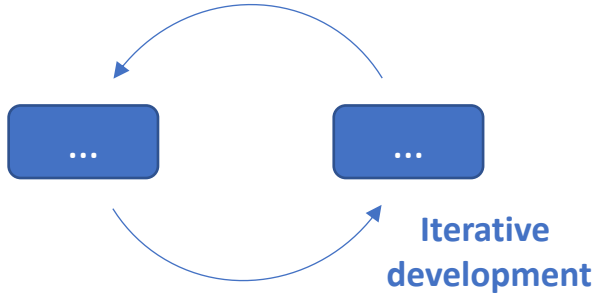


Different possible **theoretical** development approaches:

A number of **sequential activities/tasks**

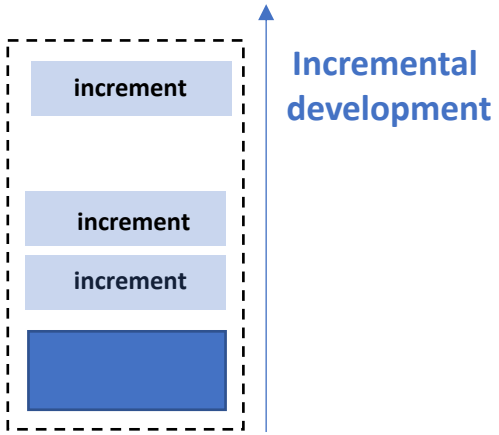
- Simple
- Ideal
- Unrealistic (in most cases)

Repeated cycles make a simple implementation of a subset of requirements evolving towards a final version



- More realistic (than linear)
- More flexible (than linear)

System functionality is sliced into **increments** (portions)

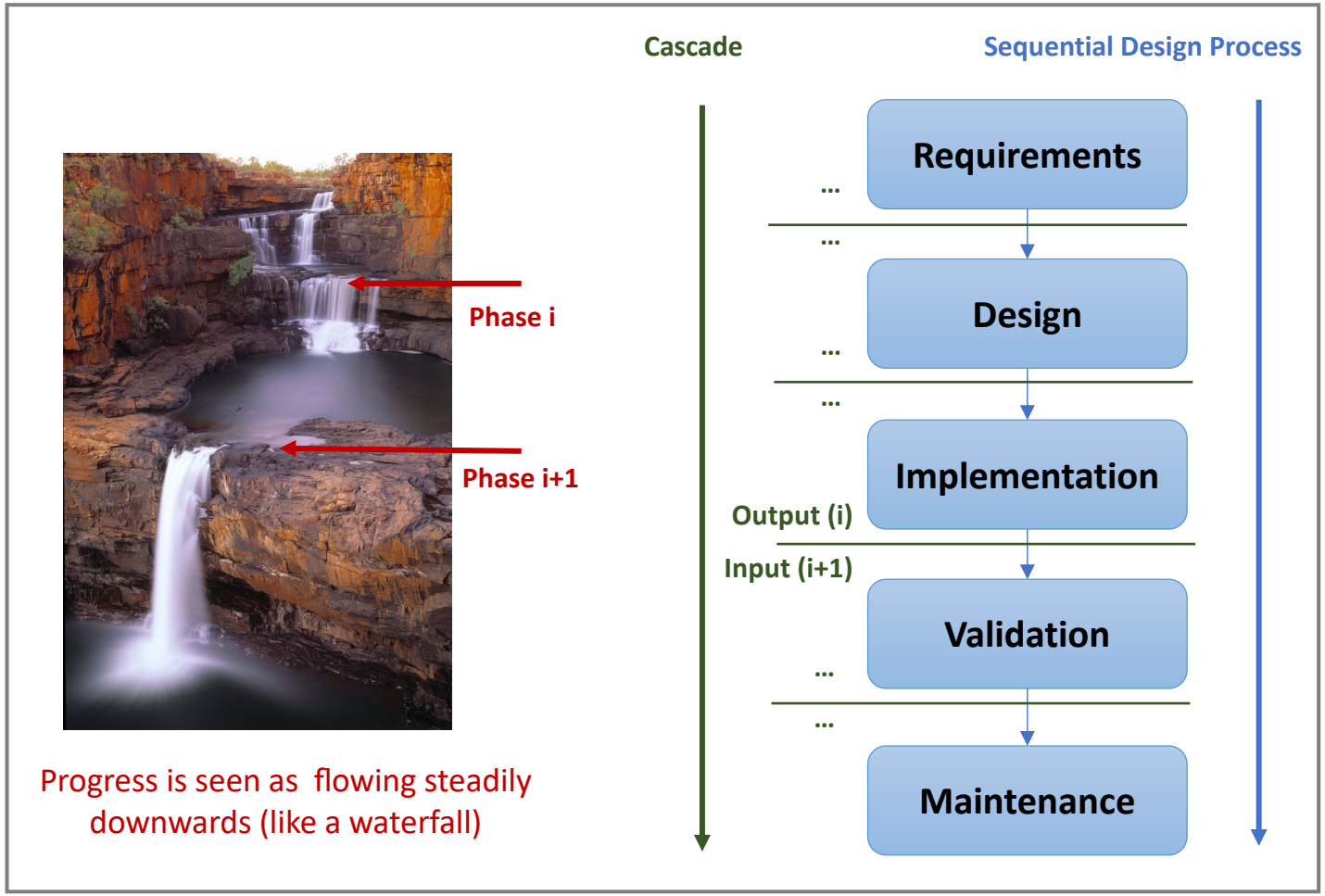


Requirements Engineering in Traditional Methodologies

Traditional Approach to System Development

Waterfall Approach

- The most intuitive (and probably simplest approach) to system design is known as **Waterfall**
- It is an activity-based process in which **each phase is performed sequentially**.



Traditional Approach to System Development

Waterfall Approach: is it “realistic”?

Simple, easy to implement and manage, probably ideal for small-sized “simple ”projects

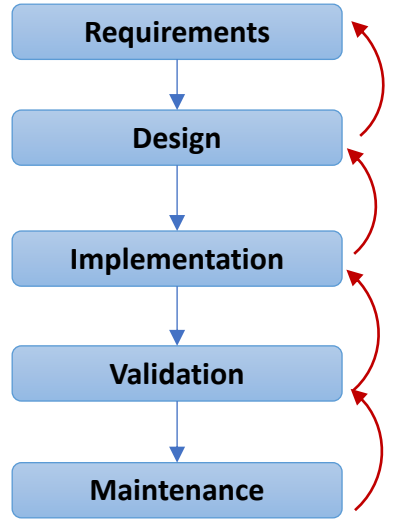
- However, the Waterfall approach is based on **unrealistic, if not flawed, assumptions at any stage of the process**.
- For instance, **requirements should be clearly defined and fully specified at a very early stage**, they shouldn't change.
- Additionally, **proof of concept, testing, validation are at a very late stage** of the process.
- **Limited interaction with “customers”**, mostly at the beginning and during validation.
- **It extensively relies on documentation** that has to be produced as part of the process.
- What if something wrong? **Inflexible**

Traditional Approach to System Development

Waterfall-like methodologies working in fact?

- In general terms, **iterations** recognize the reality of **changing requirements** as evidence shows that a significant part of final requirements may arrive after the analysis phase.
- An iterative process encourages **project participants**, including testers, integrators, and technical **to be involved earlier on and throughout**.
- Support **early error detection and correction** and enable to put into practice **lessons learned in prior iterations**.
- Iterative process allows at least in theory **larger developments**.

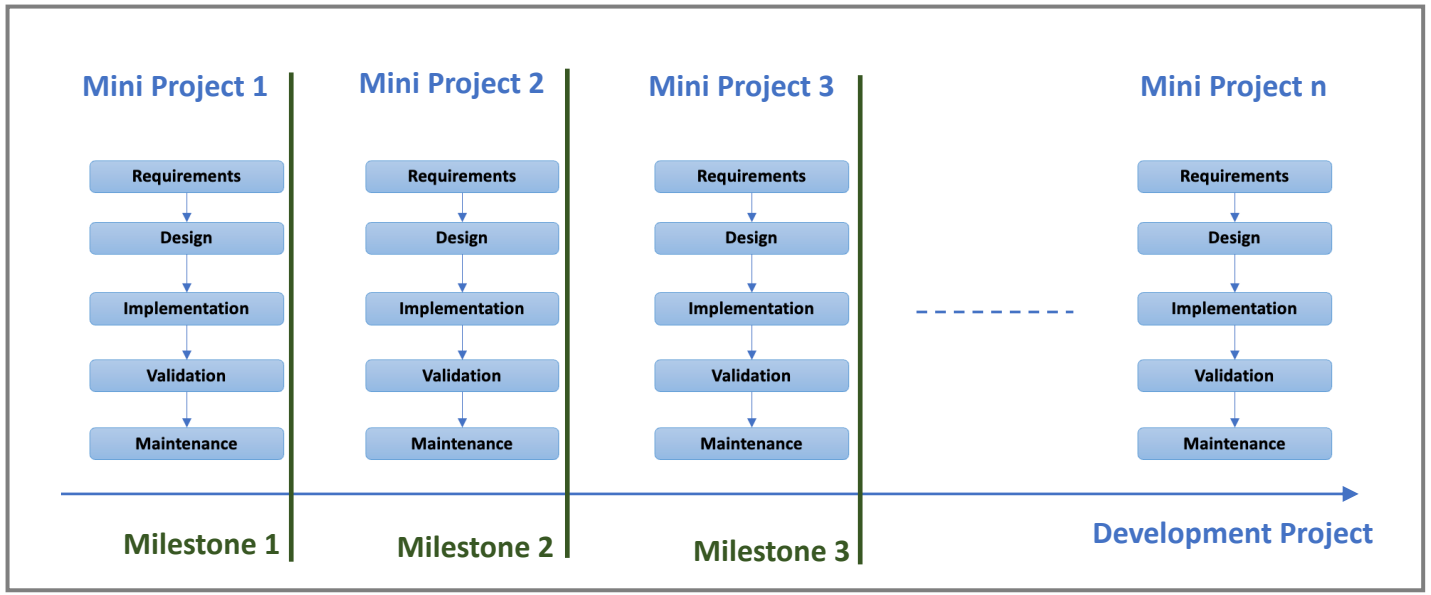
"Iterative" Waterfall extends the original model by providing feedback paths from each phase to its previous phases.



Traditional Approach to System Development

Waterfall working in fact?

- The **Mini-Waterfall model** aims to risk mitigation and increased flexibility by **breaking down the whole system development project into mini-projects**.
- Such a strategy allows the definition of **milestones** which, in practice, may be associated with sub-systems, modules, architectural components, macro-functionalities, etc.



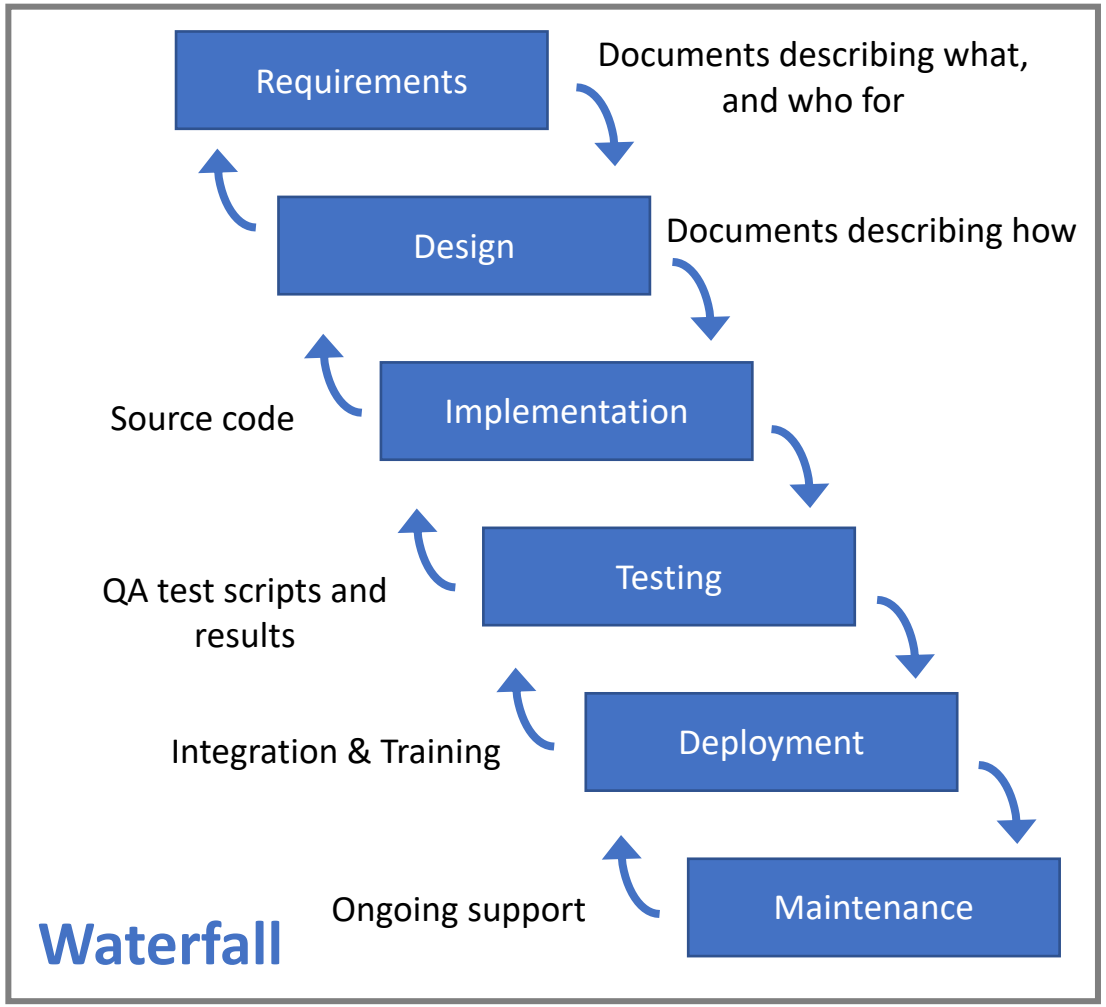
Requirements Engineering in Agile

From Traditional Methodologies to Agile

A change of mindset ….

- We need a **very clear view of requirements** before you design, and **very specific design** before you start building
  - Key learnings can happen during design and building phases
- There is a **big focus on documentation** as a gatekeeper
  - If team is communicating well and in close contact with clients, less needs to be documented.
  - If product works, no-one will care much about these documents.
- There is a **big delay between establishing the requirements and satisfying them**
  - People are prone to change their minds.
  - People (clients) often don't know what they want until they "see" it.

Can we do better? Can we be more "agile"?



Agile Methodologies

What means “agile”?

Agile approaches develop requirements and solutions through the collaborative effort of **self-organizing** and **cross-functional** teams and their customer(s)/end user(s).

It advocates **adaptive planning**, **evolutionary development**, **early delivery**, and **continual improvement**, and it encourages **flexible responses to change**.

Agile is a kind of **mindset** rather than a “simple” methodology.

Wikipedia - Agile software development
https://en.wikipedia.org/wiki/Agile_software_development

Agile Methodologies

Principles

The *Manifesto for Agile Software Development* is based on twelve principles:

1. **Customer satisfaction** by early and continuous delivery of valuable software
2. **Welcome changing requirements**, even in late development
3. **Deliver working software frequently** (weeks rather than months)
4. Close, **daily cooperation** between business people and developers
5. Projects are built around **motivated individuals**, who should be trusted
6. **Face-to-face conversation** is the best form of communication (co-location)
7. **Working software** is the primary measure of progress
8. **Sustainable development**, able to maintain a constant pace
9. Continuous attention to **technical excellence and good design**
10. **Simplicity**—the art of maximizing the amount of work not done—is essential
11. Best architectures, requirements, and designs emerge from **self-organizing teams**
12. Regularly, **the team reflects** on how to become more effective, and adjusts accordingly

WIKIPEDIA
The Free Encyclopedia

Agile Methodologies

Principles in brief

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

Requirements

AGILE Requirements Engineering

Challenges of traditional RE resolved/mitigated by agile RE practices:

- **Communication issues.** Agile RE promotes regular interaction with customer and among teams.
- **Overscoping.** Priority + gradual detailing.
- **Requirements validation.** Most relevant requirements (as per customer assessment) got a priority in every iteration and continuous prototyping facilitates the development of a progressive blueprint of the product.
- **Requirements documentation**. More face-to-face communication reduces ambiguities and the need for maintaining long documents + requirements expressed as an explanation of user demands.
- **Rare customer involvement.** Prioritisation of the requirements for all iterations ensures, to a large extent that the customer goals will be met.

Inayat, Irum, et al. "A systematic literature review on agile requirements engineering practices and challenges." *Computers in human behavior* 51 (2015): 915-929

Requirements

AGILE Requirements Engineering

Challenges of AGILE RE:

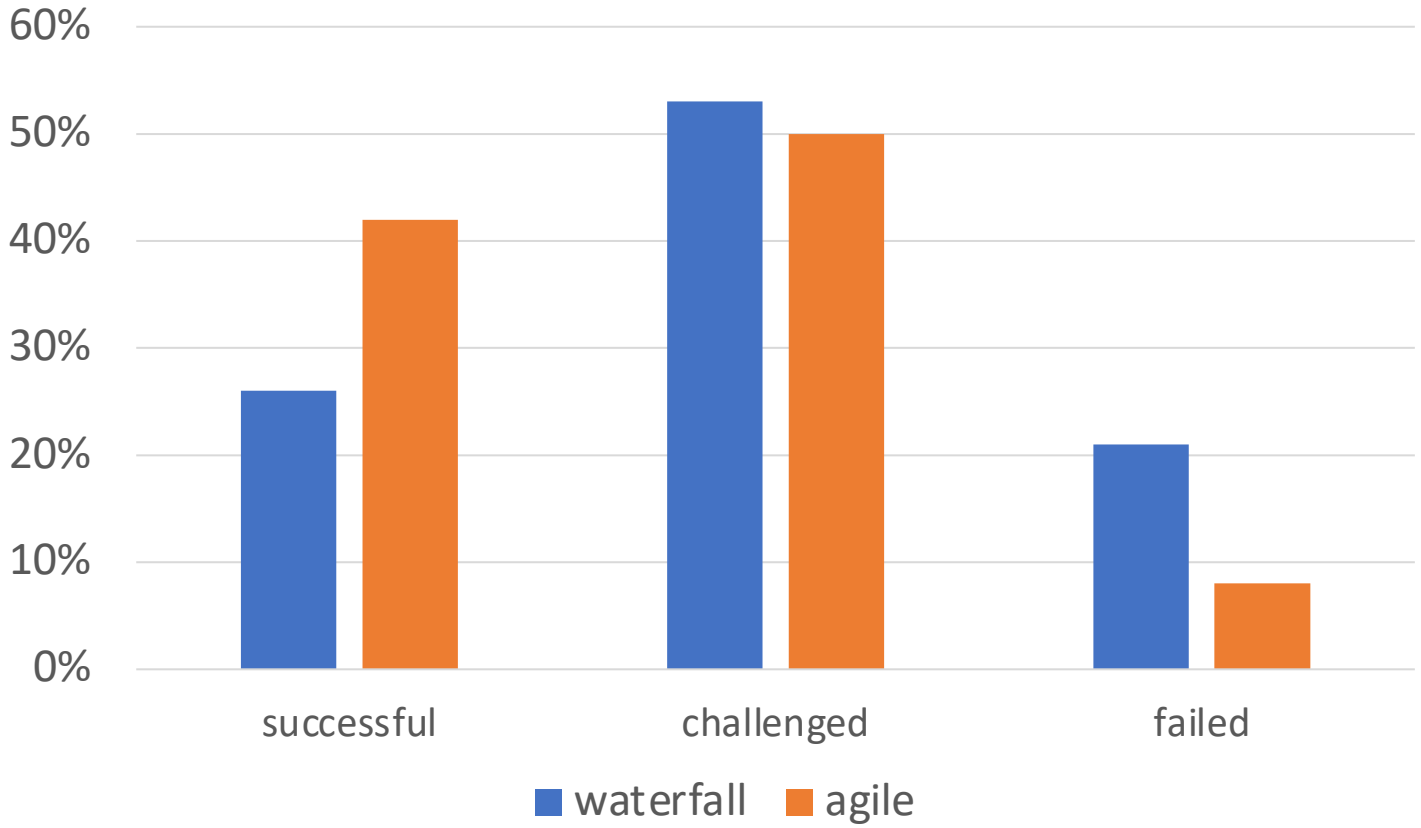
- **Minimal documentation.** Traceability issues
- **Customer availability.** Increase in re-work
- **Inappropriate architecture.** Increase in cost
- **Budget and time estimation.** Project delays
- **Neglecting non-functional requirements.** *There are different kinds of requirements, later on in the subject!*
- **Customer inability and agreement.** Increase in rework
- **Contractual limitations.** Increase in cost
- **Requirements change and its evaluation.** Increase in work delay

Answering a previous question: AGILE is NOT perfect!
But still very good!

Inayat, Irum, et al. "A systematic literature review on agile requirements engineering practices and challenges." *Computers in human behavior* 51 (2015): 915-929

Agile vs Waterfall

Who's "better"?



Agile projects are
2x more likely to succeed and **1/3 less likely to fail**
than waterfall projects

From the 2018 Standish Group Chaos Study


Thank You!