MANCHESTER 1824

The University of Manchester

COMP23420 Lecture 8

Software Development Processes

Kung-Kiu Lau

kung-kiu@cs.man.ac.uk

Office: Kilburn 2.68

Overview

So far we have used one (generic) development process

What other (kinds of) processes are there?

Here we look at a few

Software Development Processes

A software development process is an approach to building, deploying and maintaining software

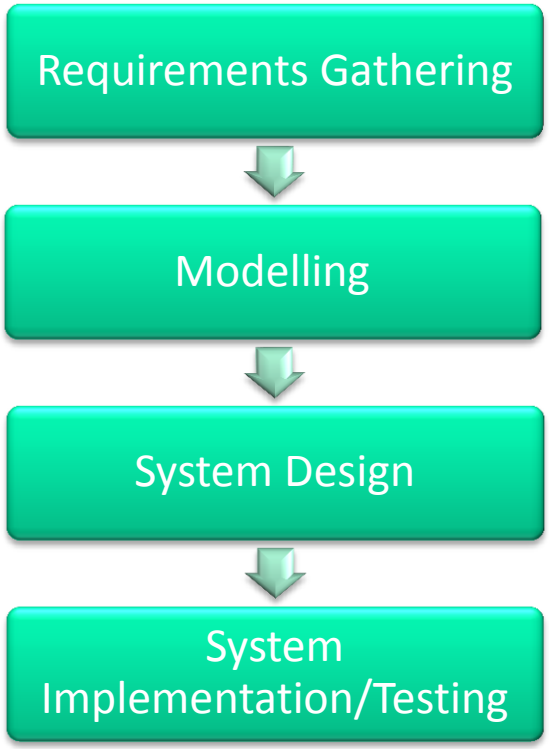
A good process will help us to plan, allocate resources, set a realistic budget etc…

… but good processes are hard to find

No one process is appropriate for all projects

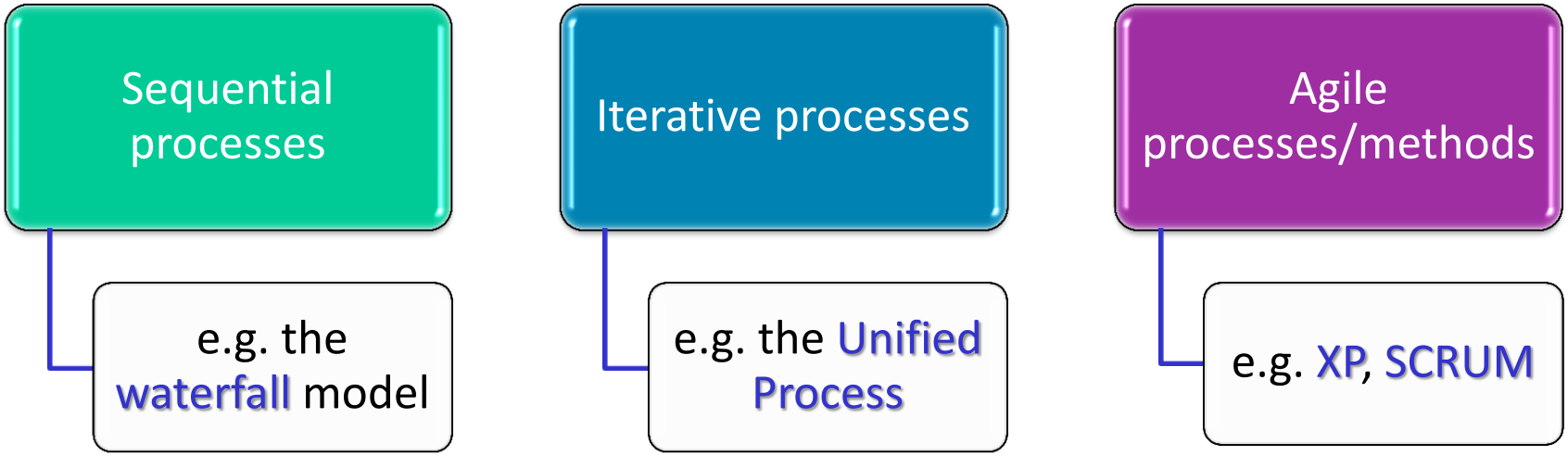
NB: Processes are sometimes also called life cycles, models, ...

The Generic Development Process

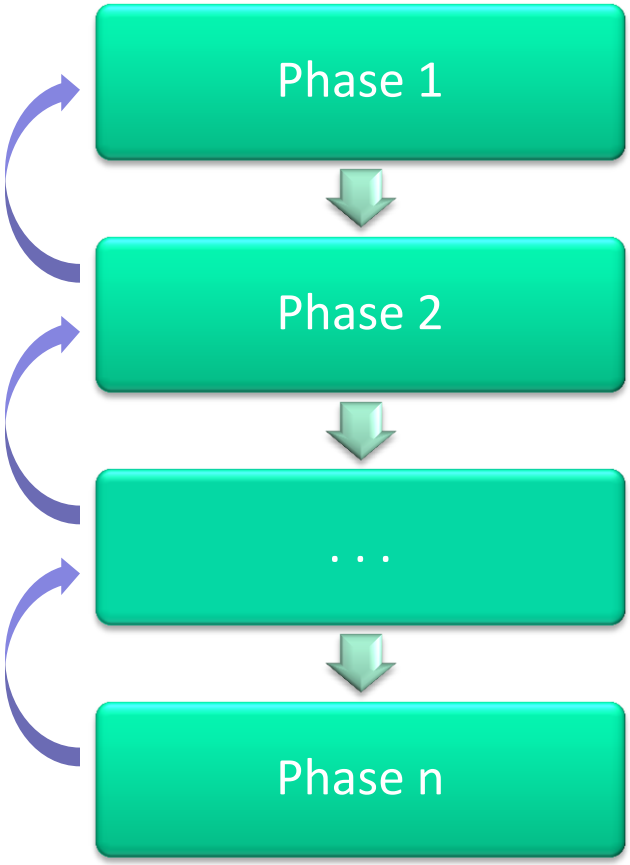


This is an example of a (predominantly) sequential process.

Kinds of Software Processes

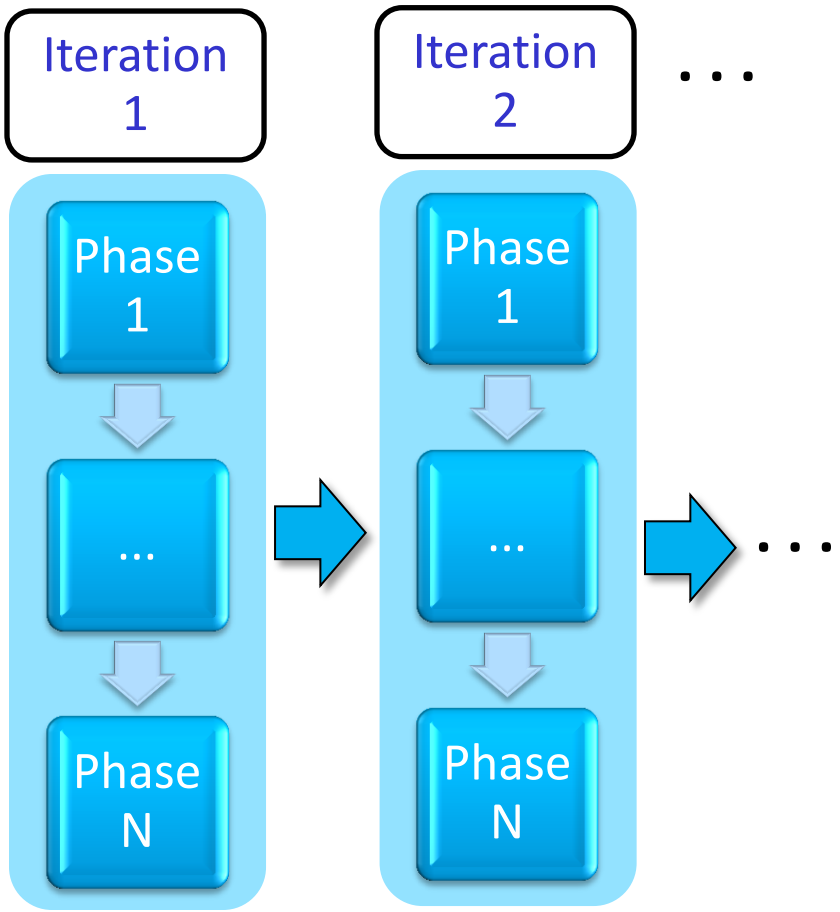


Sequential Processes



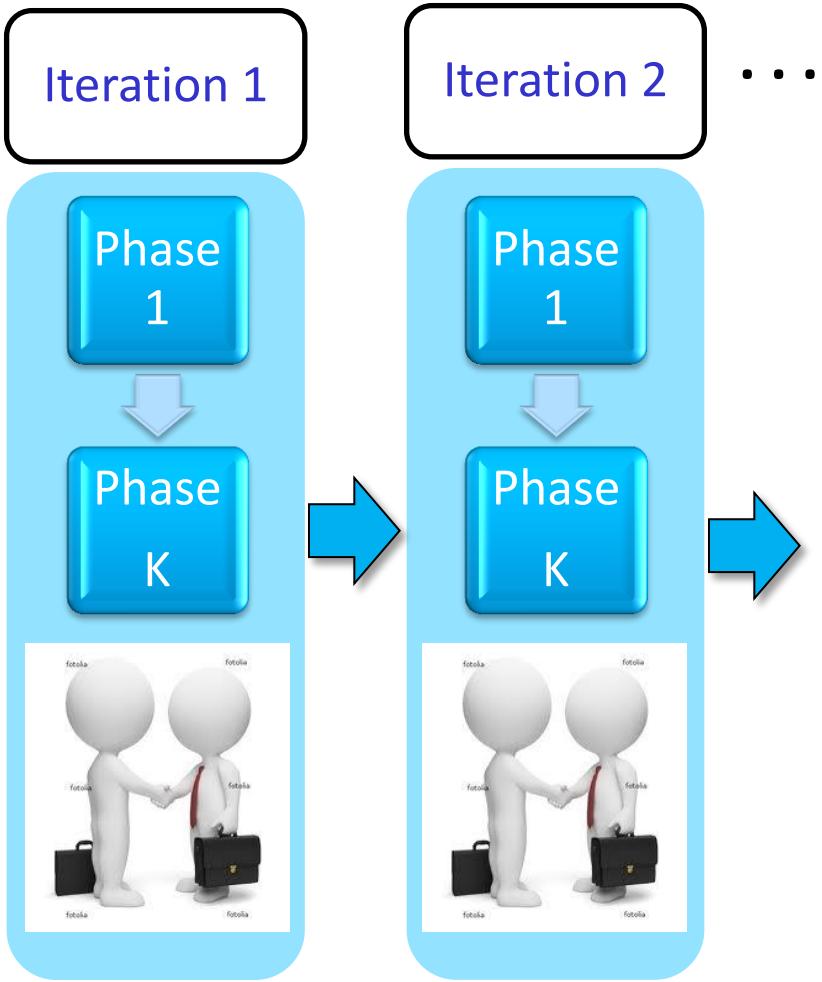
- Strong emphasis on well-defined, complete phases
- Each phase has well-defined deliverables
- Each phase deals with the complete system
- Feedback may cause adjusting previous phase

Iterative Processes



- Iterates over well-defined cycles
- Each cycle has short phases
- Each phase has well-defined activities
- Each iteration produces a product release

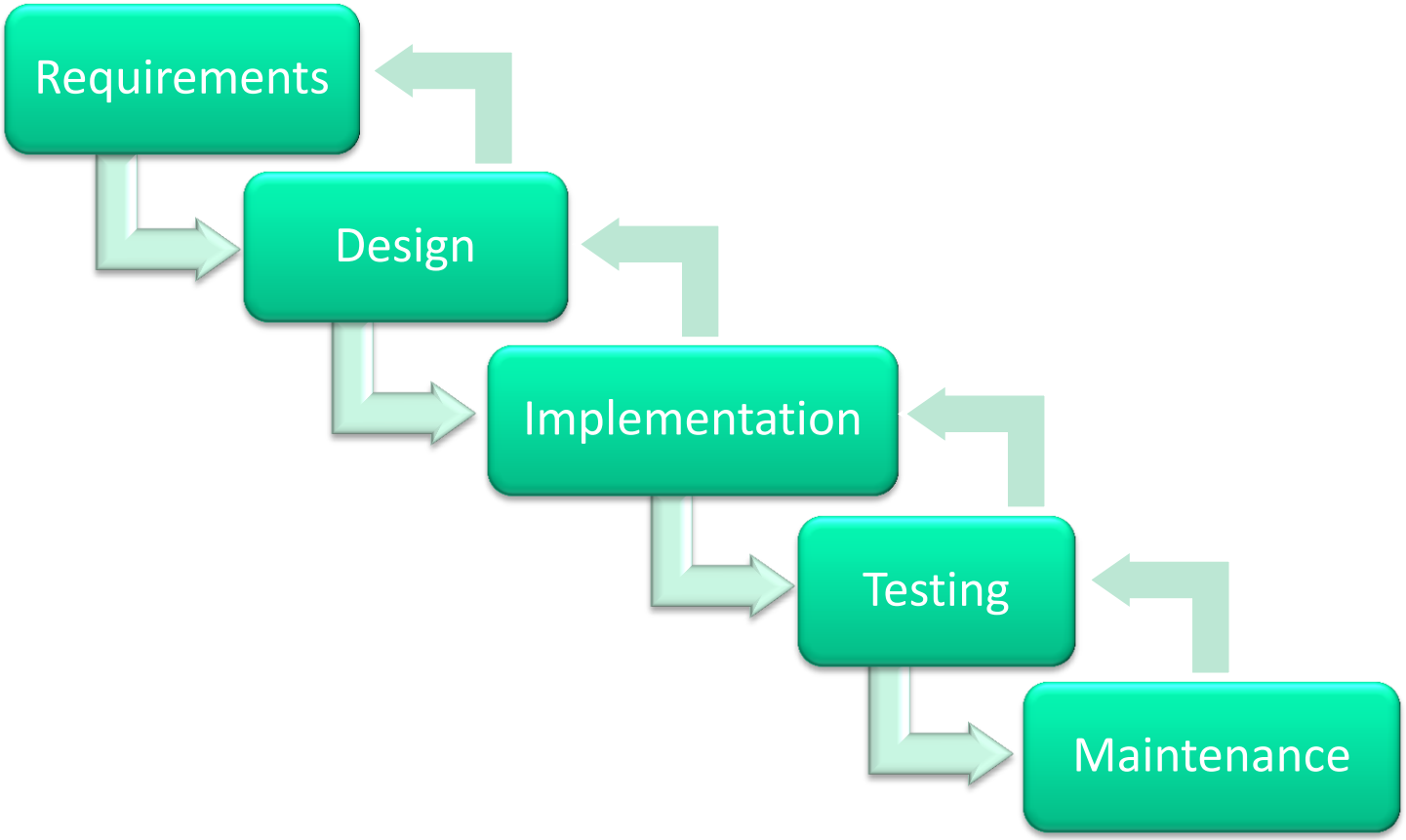
Agile Processes



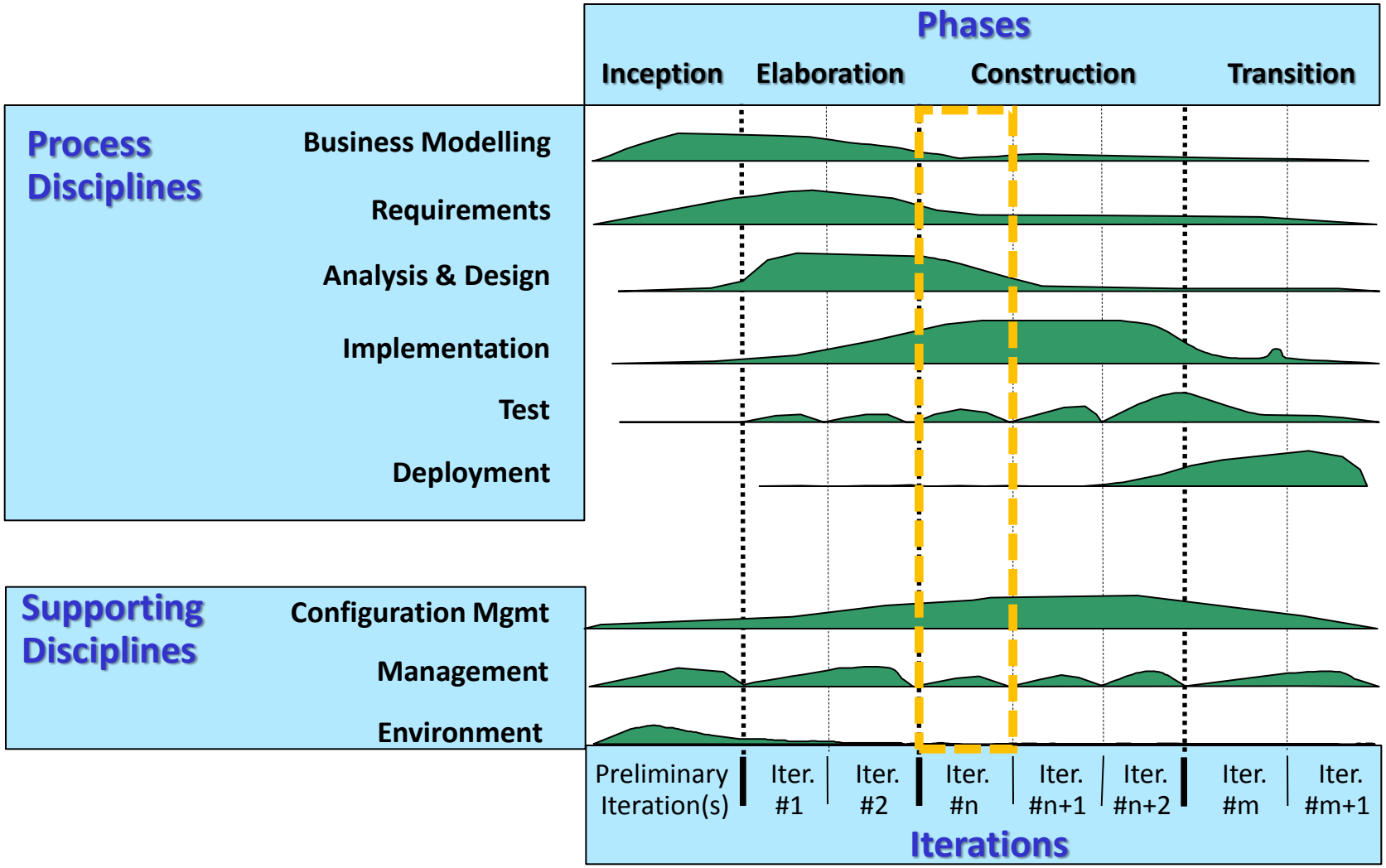
. . .

- Iterates over **short cycles**
- Each iteration produces **part of the software**
- Software **presented to customer** after each iteration
- **Requirements changes** checked with customer after each iteration

Sequential Process: Waterfall Model

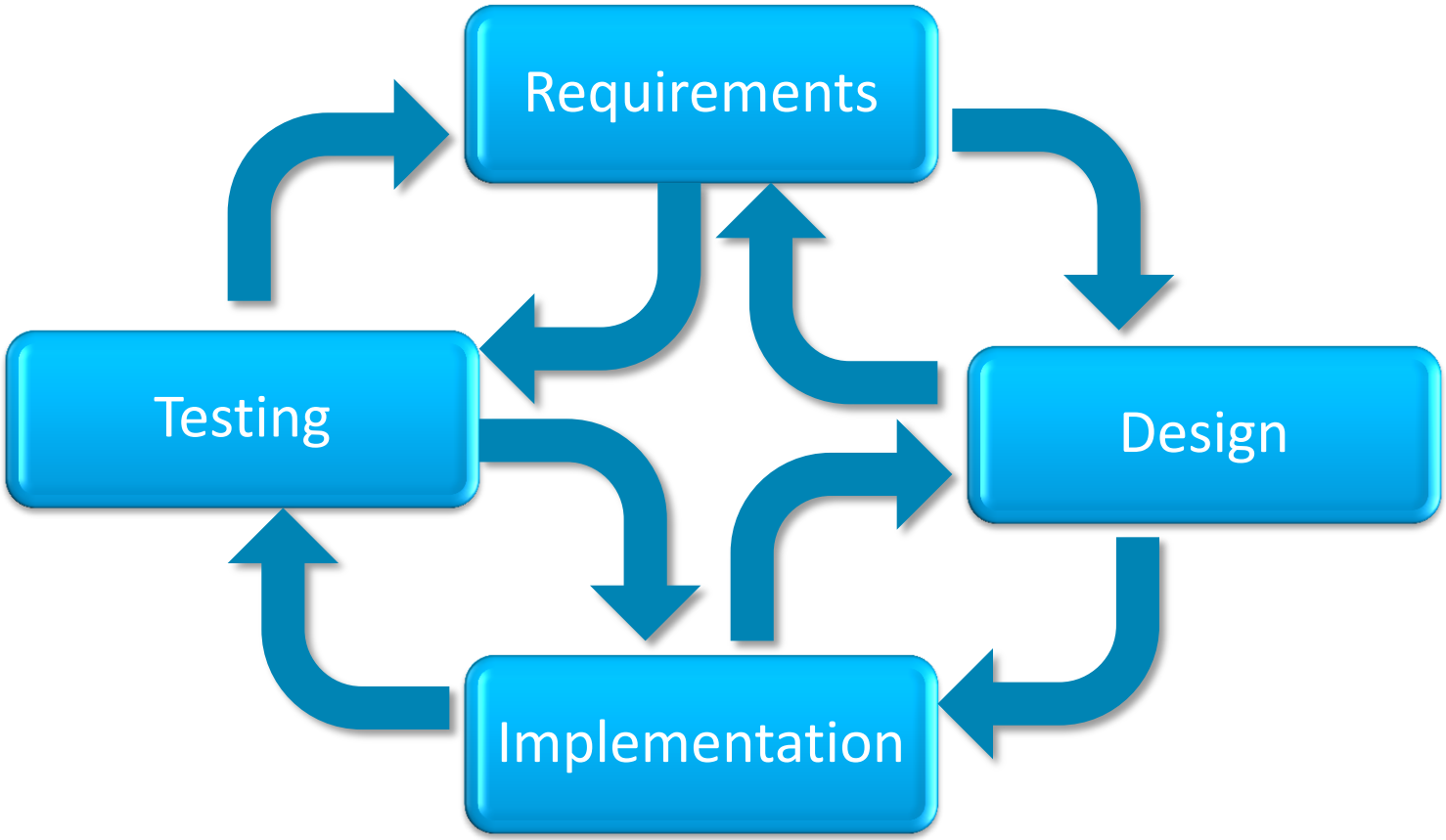


Iterative Process: Unified Process



(In an iteration you walk through all disciplines.)

Agile Process

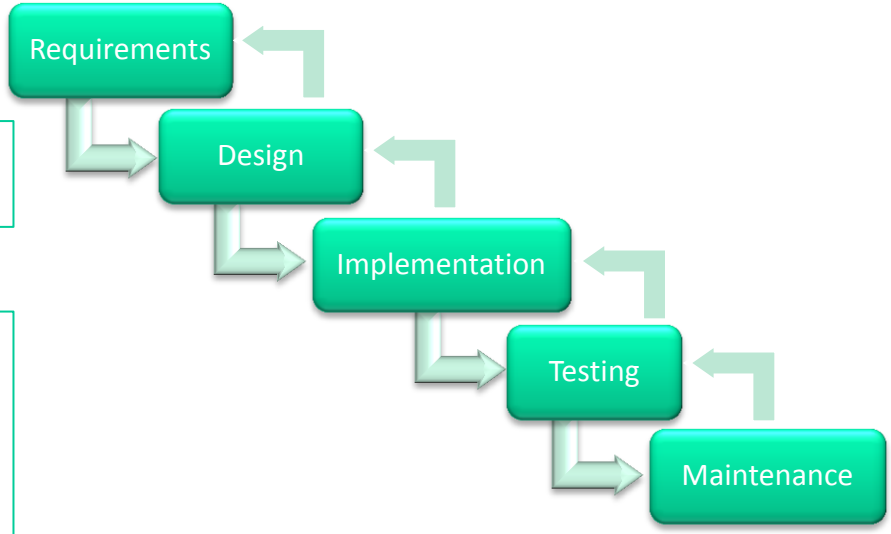


Sequential Process: Waterfall Model

Traditional, sequential

Proceeds in phases:

- Requirements gathering/analysis
- System design



In practice, the process often involves iteration between phases.

These phases happen in sequence, with milestones and deliverables at the end of each phase.

Waterfall Model: Motivation

The later a problem is found (especially after deployment), the more it costs to fix…

… so put a lot of effort into getting it right at the start

Well-defined milestones and deliverables make budgeting and planning easier

Everything is well documented, so causes of (and responsibilities for) problems can be clearly identified

Promotes specialisation, e.g. systems analysts vs designers vs programmers

Waterfall Model: Common Problems

Deliverables are often not produced on time, or are fudged (the Blank Tape Trick)

Analysis Paralysis often occurs

Specialisation can cause poor communication

The system delivered does not meet the users' needs – on average 45% of the features specified are never used (Larman)

The "maintenance" phase is often prolonged and traumatic

The project is usually grossly late and over-budget

Why/When the Waterfall Model Doesn't Work

Main reason is that, except for a few special cases you won't get the requirements right the first time

Users don't see anything working until very late, so they can't tell you that you misunderstood what they wanted…

… and anyway what they now want is different…

… and what they actually need is different again

A successful development process has to be able to deal with the reality of fundamental and rapid requirements change

What Causes Requirements Change?

- Experience with using the software ("The UI is too complex…")
- Change in business processes/management direction
- Marketing requirements ("I would buy it if…")
- Technology change
- Standards/regulations/certification etc.
- Misunderstanding and general user inconsistency

Agile Methods/Processes

Aim to respond in an "agile" way to changing requirements

Iterative – consist of short cycles where part of the software is produced

Frequent interactions with the customer

Strong emphasis on testing

Done in small "self-organising" teams with little specialisation or explicit leadership

Best known are Extreme Programming (XP) and SCRUM

Extreme Programming (XP)

Requirements are determined from "user stories" (similar to informal use cases)

Customer representative always available

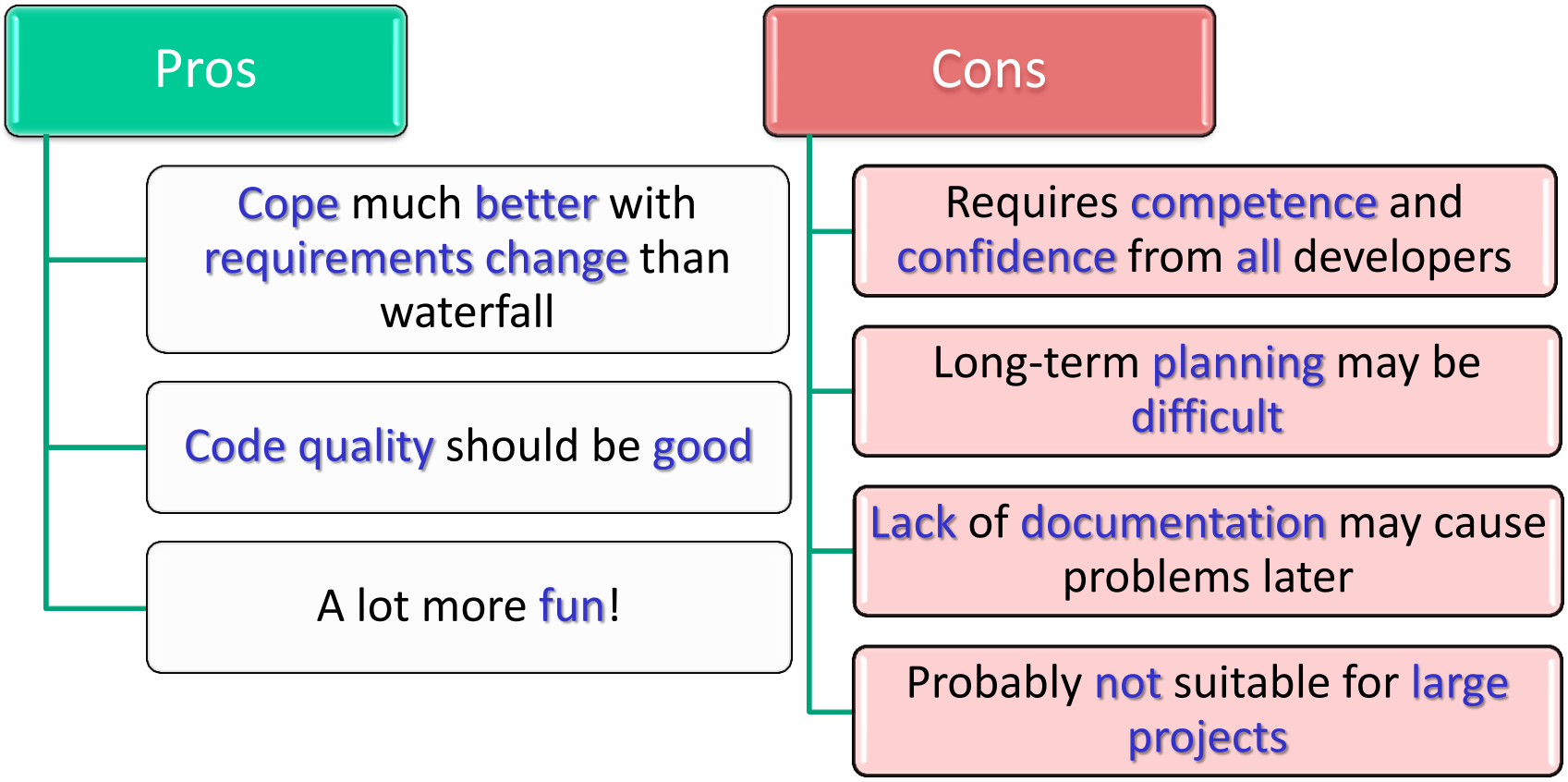
Unit tests are written before the code to be tested

Programming in pairs – one codes the other reviews, swapping frequently

Strong emphasis on simplicity of design

"Refactor whenever and wherever possible"

Agile Methods: Pros and Cons



The Unified Process (UP)

Designed by the same "three amigos" who gave us UML

Designed for building object-oriented systems

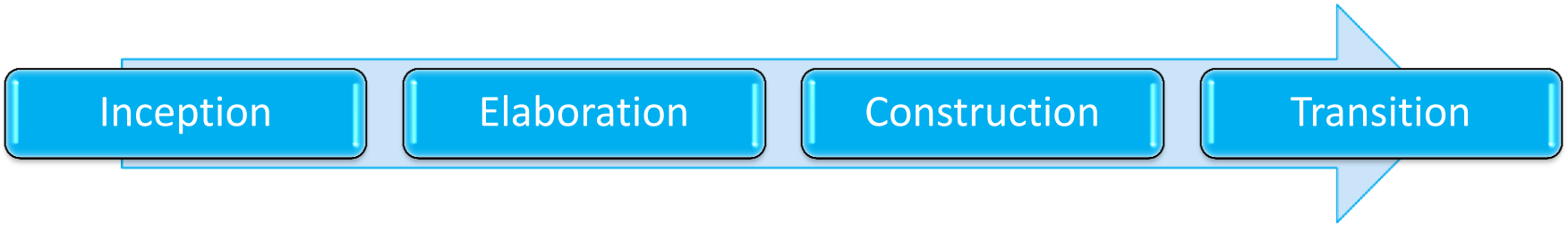
The nearest to an industry-standard process there is (currently)

An iterative process – assumes you can't get all the requirements right at the start

Distinguishes between phases, disciplines, and artefacts

Specifies lots of things you can do, but you don't have to do any of them – except write the code

Unified Process: Phases



Inception

- Define the scope of project – including initial "feasibility study and project go-ahead''

Elaboration

- Plan project, specify features, baseline architecture – more detailed study, production of high-risk parts of the system

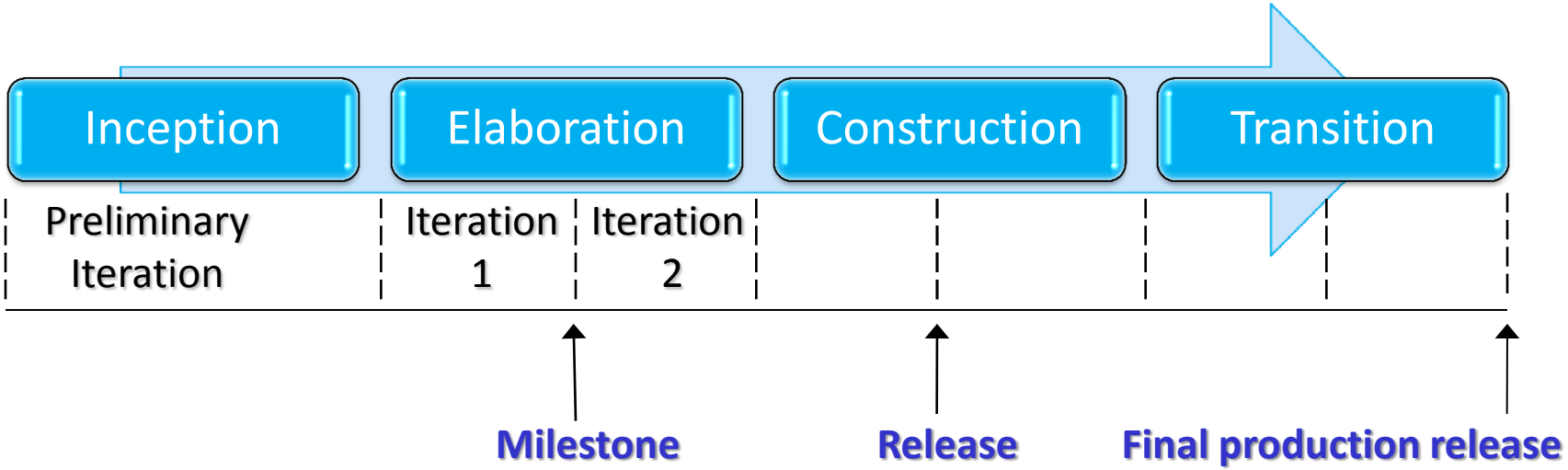
Construction

- Build the product

Transition

- Transition the product into end user community
- Acceptance testing, "maintenance" etc.

United Process: Iterations and Milestones

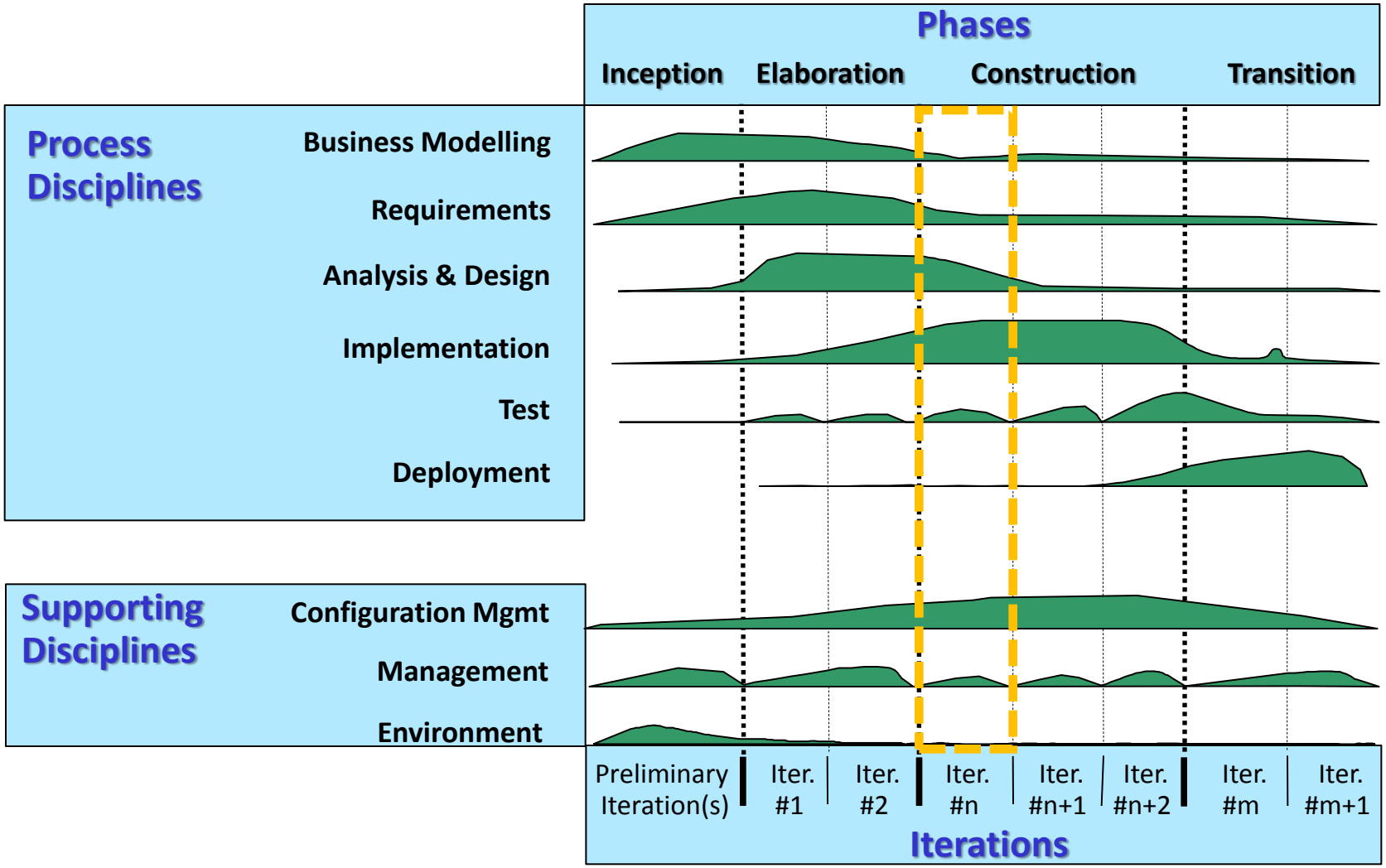


Each phase and iteration has some risk mitigation focus, and concludes with a well-defined milestone

The milestone review provides a point in time to assess how well key goals have been met and whether the project needs to be restructured in any way to proceed

The end of each iteration is a minor release, a stable executable subset of the final product

Unified Process: Disciplines



(In an iteration you walk through all disciplines.)

Iterative Development in the UP

Development progresses in a set of time-boxed iterations, typically 4-6 weeks

Each iteration produces a production version of a subset of the system – not the same as rapid prototyping

Time-boxed means if you don't get it done you don't extend the iteration, you change the plan

Summary: Waterfall vs Iterative

	Waterfall	Iterative
Tackling high-risk or difficult problems	no attempt to identify and tackle riskiest issues first	early iterations focus on driving down the risk
Requirements speculation and inflexibility	assumes that requirements can be fully specified and then frozen in the first phase of the project • stakeholders want to see something concrete very early • market changes	requirements tend to stabilize after several iterations

Summary: Waterfall vs Iterative

	Waterfall	Iterative
Design speculation and inflexibility	• dictates that the architecture should be fully specified once the requirements are clarified and before implementation begins • since requirements usually change, the original design will not be reliable • lack of feedback on design until long after design decisions are made	