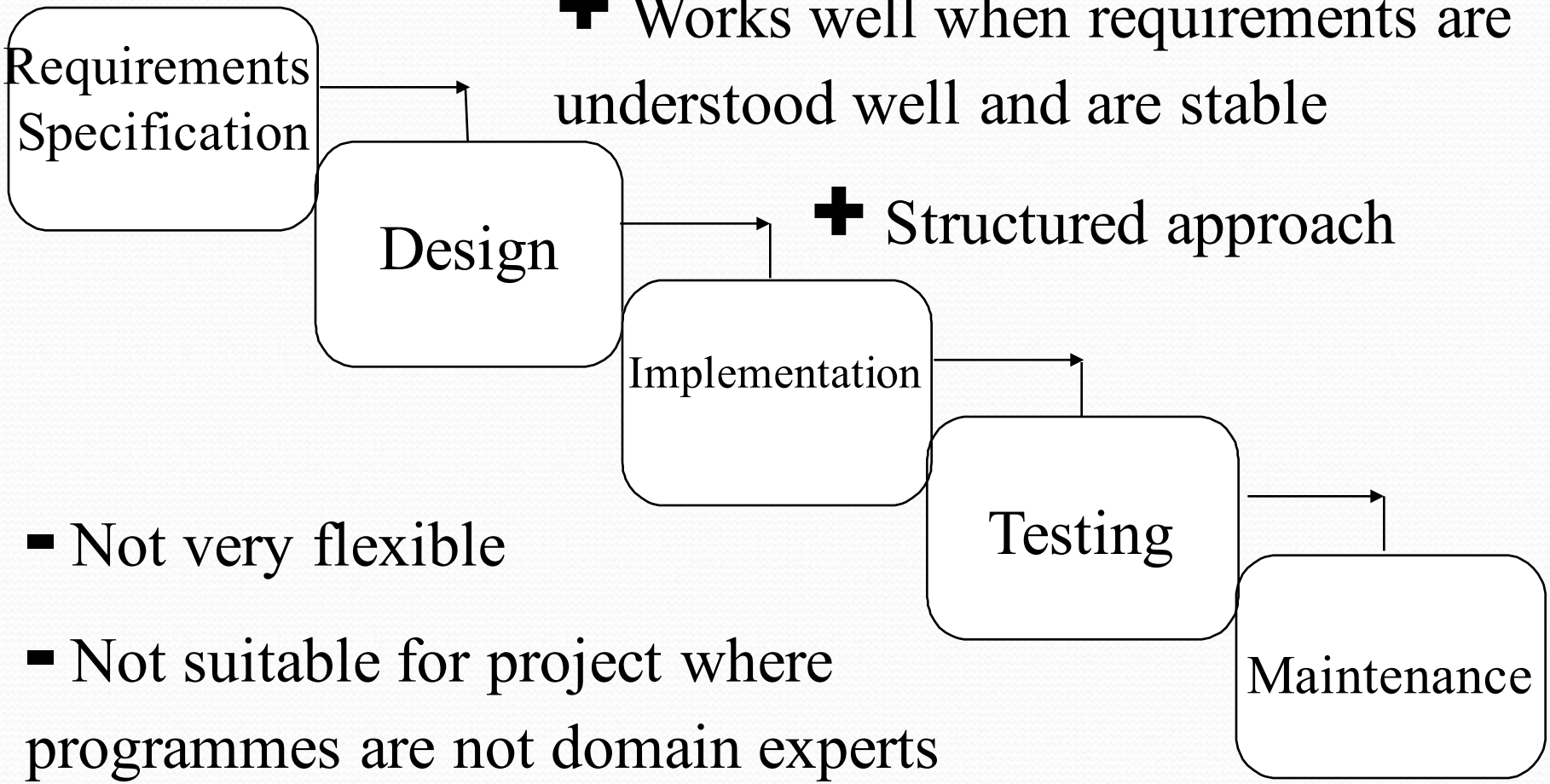
Software Process Model

- What is the Software Process Model
  - Waterfall Model
  - Spiral
  - Prototype
  - Incremental development
  - RUP
  - Agile
- Which Model should you choose?

Choosing a Model

- Consider your understanding of the requirements
  - Will collecting these be easy or challenging
- Expected lifetime of the project
  - Will the software require maintaining
- What is the level of risk?
- Scheduling constraints
- Interaction with management & customer
- Expertise of the development team

The Waterfall Model



+ Works well when requirements are understood well and are stable

+ Structured approach

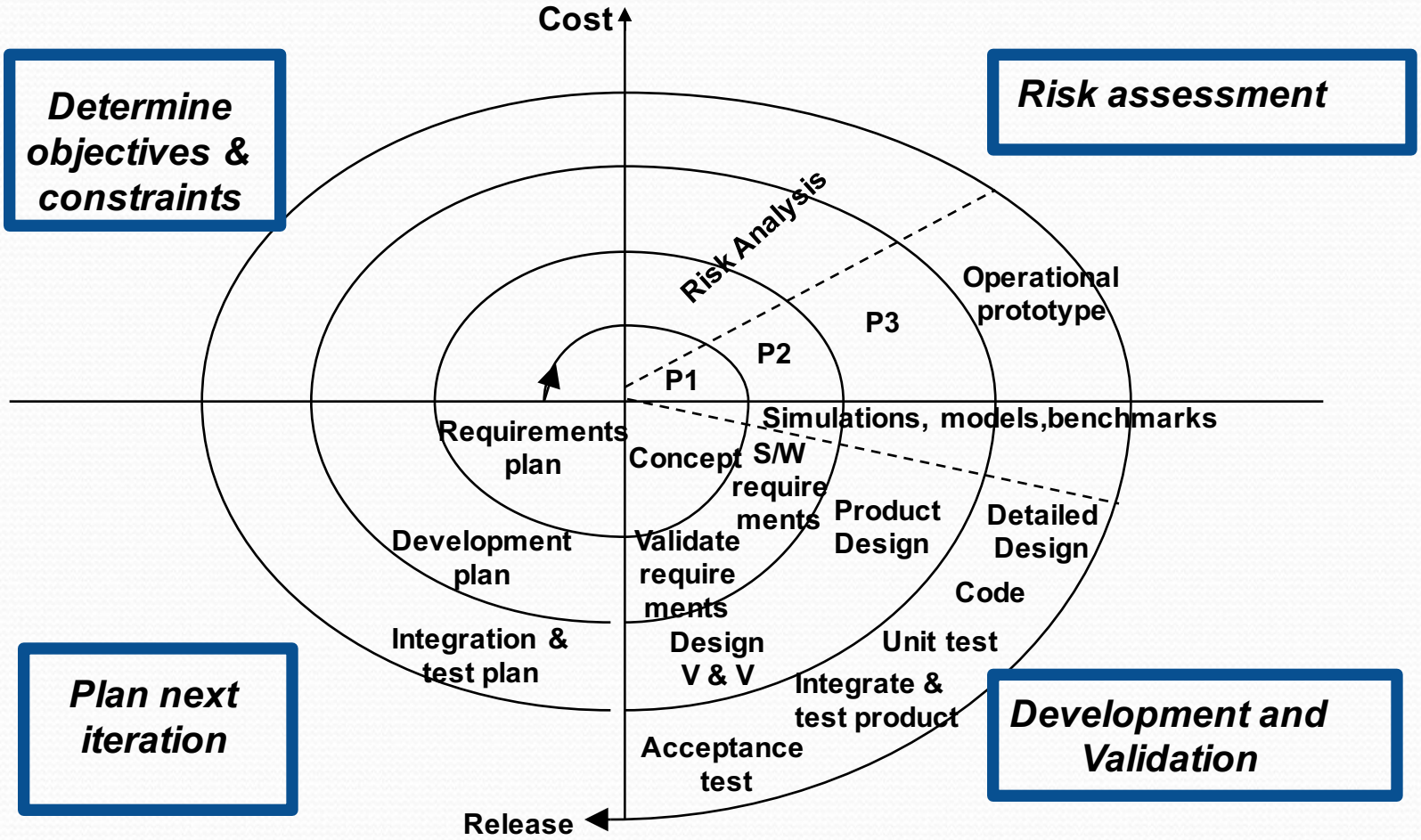
- Not very flexible

- Not suitable for project where programmes are not domain experts

- &+ Documentation heavy

Spiral Model Iterative, risk oriented model

Proposed by Boehm in 1986.



Spiral Model Incremental, risk oriented model

Proposed by Boehm in 1986.

Risk Analysis requires expertise

Complex & Costly

Risk Reduction

Tries to eliminate errors early

Functionality can be added

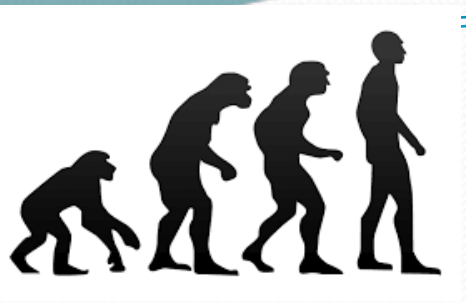
Software produced early

Allows for change/re-evaluation

Prototyping:

Two Types

- Throw away
  - used to refine the requirements
  - particularly useful for systems with an emphasis on the user interface.
  - inviting user feedback
  - often used when new technology involved



Evolutionary Prototyping

Initial Concept	Design & Implement initial prototype	Refine Prototype until acceptable	Complete and release prototype

- Start by developing the parts they understand
- Develop initial implementation exposing it to user comments / feedback.
- Refine it through repeated stages until an adequate system has been developed.

Immediate Feedback

Risk of implementing the wrong system to minimised

- Difficult to plan

Who long will the project take to complete ?

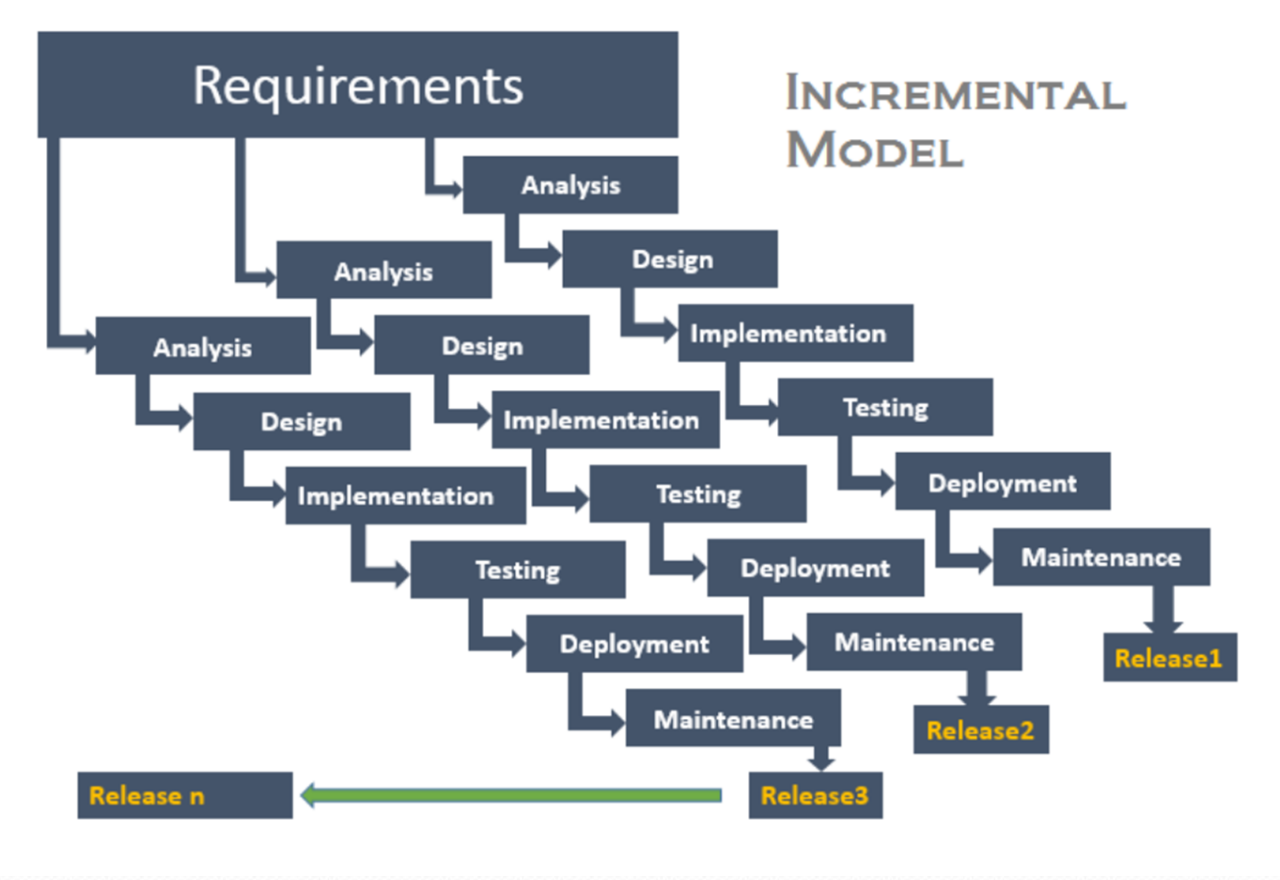
- Code and fix

Not good quality code, difficult to maintain later

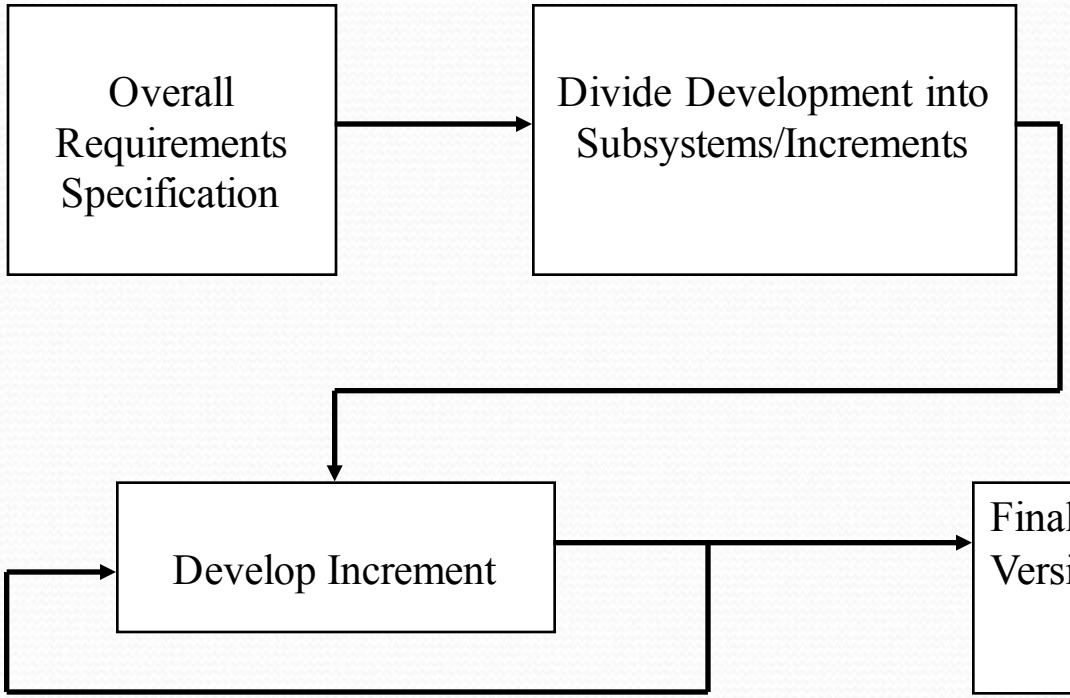
Incremental Development

Where used

- Used for the development of large systems where requirements may be subject to change.



Incremental Development



Each increment delivers a product which can be used.

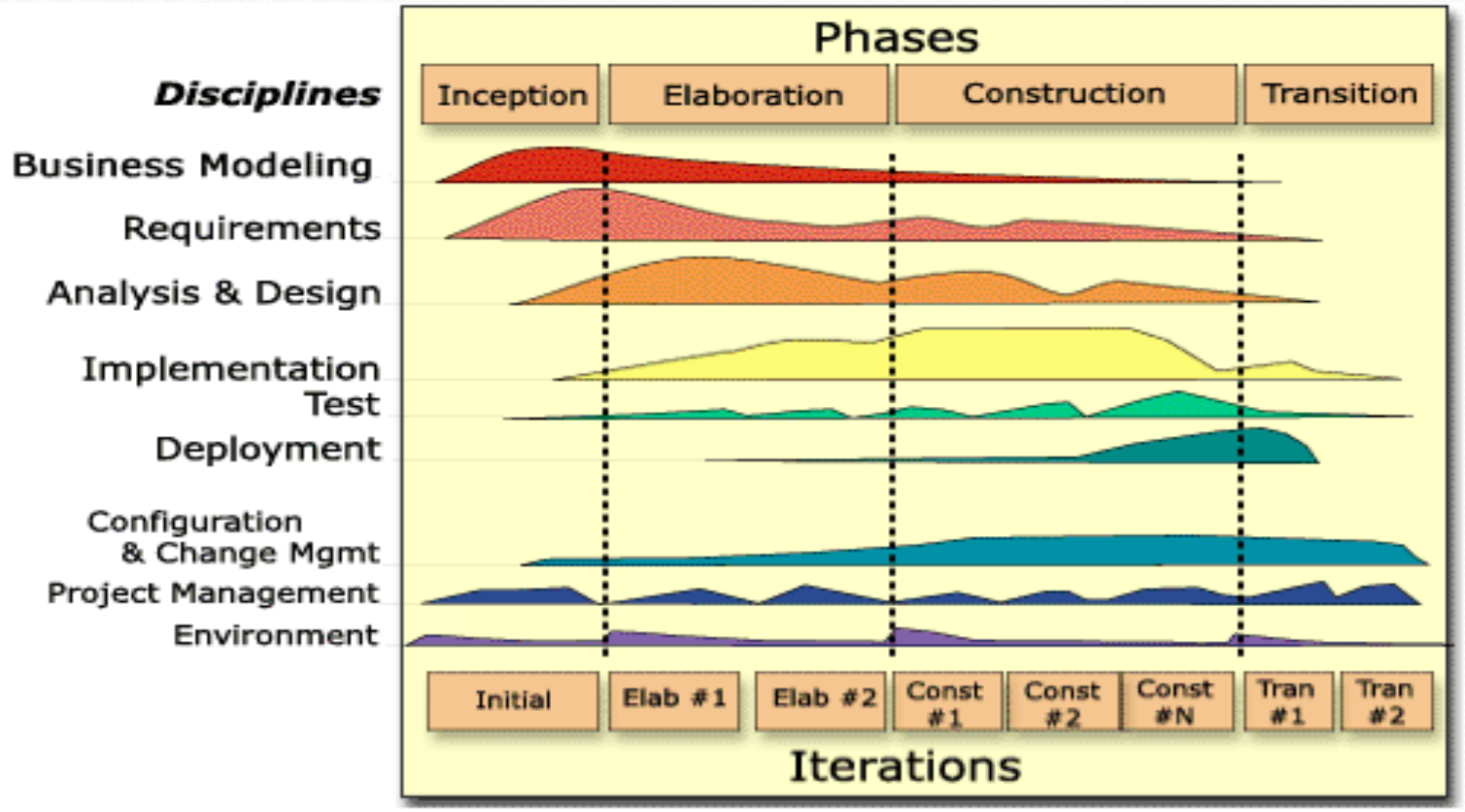
After each increment users can evaluate the system and provide feedback.

Pros and Cons:

- Core capabilities are delivered early in the project;
- Core capabilities can be evaluated by the customers early in the project;
- A ‘safe’ approach

- Can be difficult to split the problem into appropriate increments;
- System architecture has to be established before requirements are complete
- Extra time must be spent on testing, documenting and maintaining ‘temporary’ products until the full system is delivered.

The Rational Unified Process (RUP)



- A risk-driven, UML use-case-based, iterative development process

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

http://www.agilemanifesto.org/

Philosophy
a little at a time

- The plan changes a little at a time,
- Design changes a little at a time
- The team changes a little at a time

- Don't design, plan or code more than is needed at the moment so that options for the future remain open.

Philosophy part 1

- As each ‘piece’ is added developers learn what works and what doesn’t.
- Customer learns what value the system offers and what features are needed next.
- Do the simplest thing that works and meets current needs, don’t build in extra complexity ‘in case’.
- The team should take pride in delivering high quality.

https://vimeo.com/25121889

Agile Methods

- Group of methods based on highly iterative and incremental development.
  - Test Driven Development (TDD)
    - Write test cases for our requirements (before coding)
    - Write enough code to just pass the test cases
    - Refactor to improve code quality
  - XP & Pair programming
  - LEAN
  - Scrum (Future Lecture)

Coursework Coversheet

Student number for person submitting

Student Number

Module Code

Submission Date

Hours spent on this exercise

Special Provision

(Please place an x in the box above if you have provided appropriate evidence of need to the Disability & Dyslexia Service and have requested this adjustment).

Group Submission

For group submissions, *each member of the group must submit a copy of the coversheet.* Please include the student number of the group member tasked with submitting the assignment.

Student number of submitting group member

Student numbers of other contributing team members