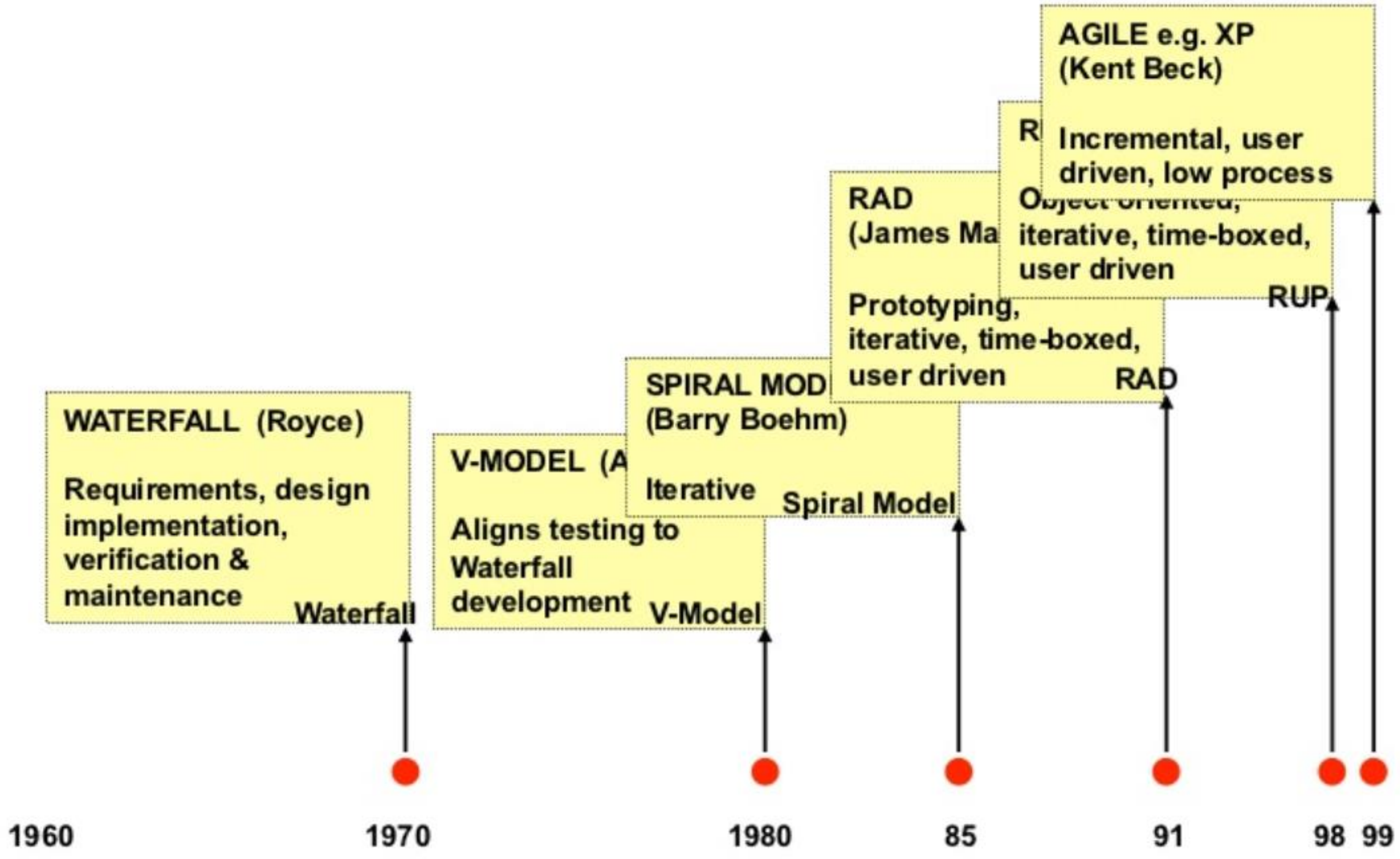
Software Development Life Cycle Models

dr. Asta Slotkienė
asta.slotkiene@vgtu.lt

The Six Phases of a (Big) Project

1.	Enthusiasm,	1.	Entuziazmas,
2.	Disillusionment,	2.	Nusivylimas,
3.	Panic and hysteria	3.	Panika ir isterija
4.	Hunt for the guilty,	4.	Ieškome kaltų
5.	Punishment of the innocent,	5.	Bausmė nekaltiems,
6.	Reward for the uninvolved.	6.	Atlygis už nedalyvavimą☺

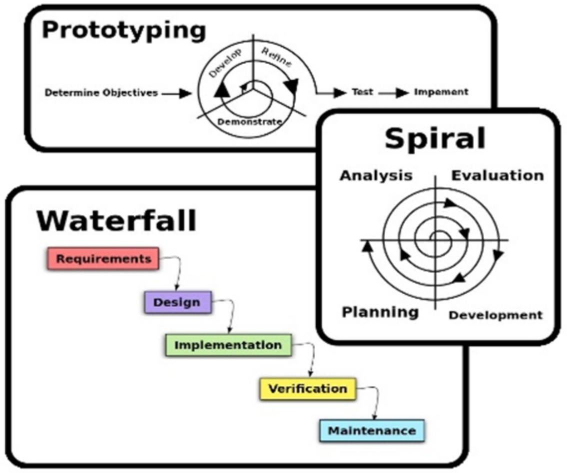
History of Software process model


AGILE e.g. XP
(Kent Beck)
Incremental, user driven, low process
R
Object oriented, iterative, time-boxed, user driven
RUP
RAD
(James Ma
Prototyping, iterative, time-boxed, user driven
RAD
SPIRAL MOD
(Barry Boehm)
Iterative
Spiral Model
V-MODEL (A
Aligns testing to Waterfall development
V-Model
WATERFALL (Royce)
Requirements, design implementation, verification & maintenance
Waterfall
1960
1970
1980
85
91
98
99

Software process models

A framework that describes the activities performed at each stage of a software development project.

- WaterlFall model
- V-Model
- Component-based model
- Prototyping
- iterative,incremental and evolutionary models



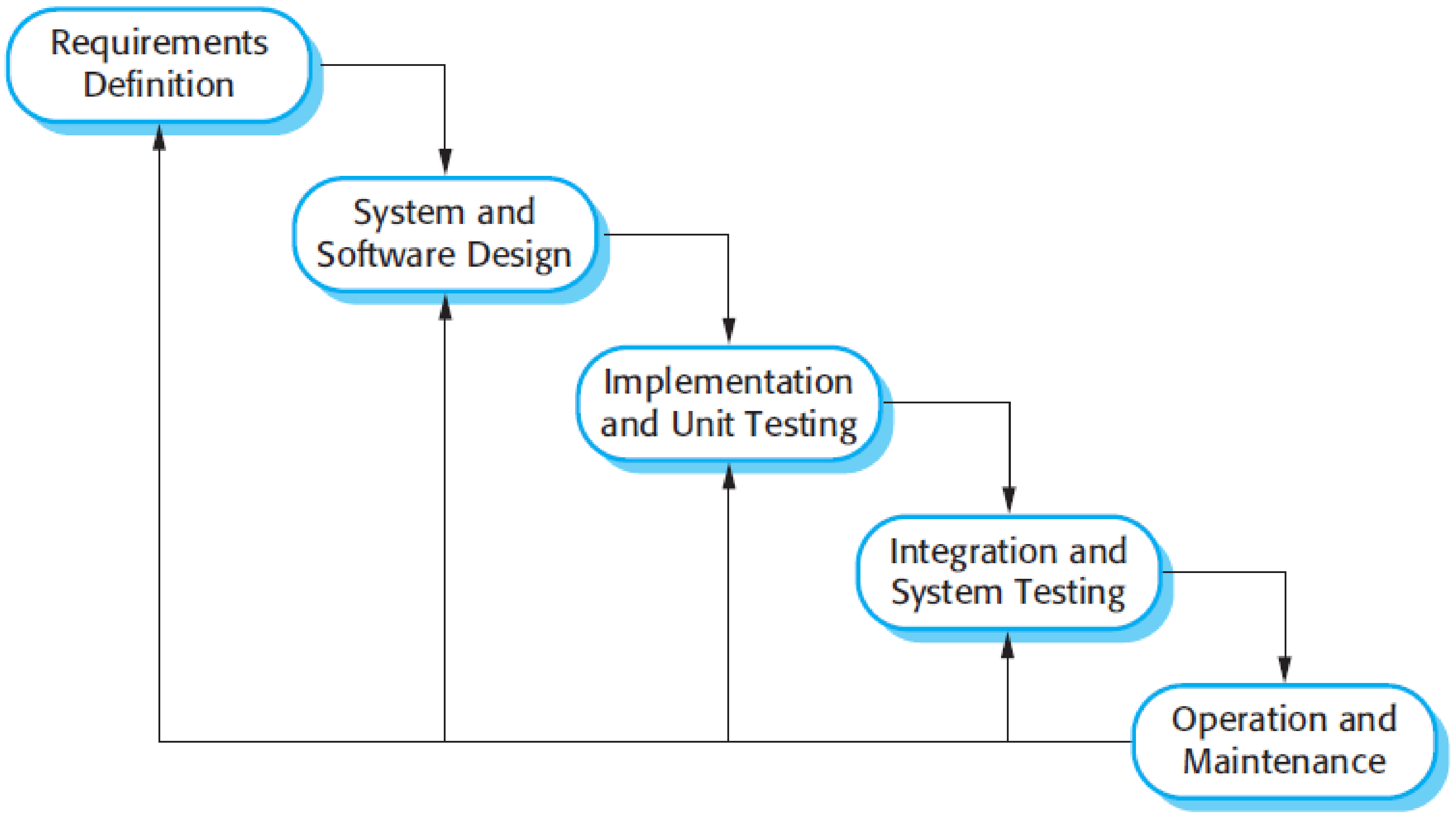
Waterfall by Royce

- The first published model of the software development process was derived from engineering process models used in large military systems engineering (Royce 1970)

- W.W.Royce about Waterfall sad,
- *„I believe in this concept, but* ***the implementation described above is risky and invites failure****.“*

- The Rational Model (waterfall) may seem naive to us today. But it is a very natural model for people to conceive (F.Brooks, The Design of Design)

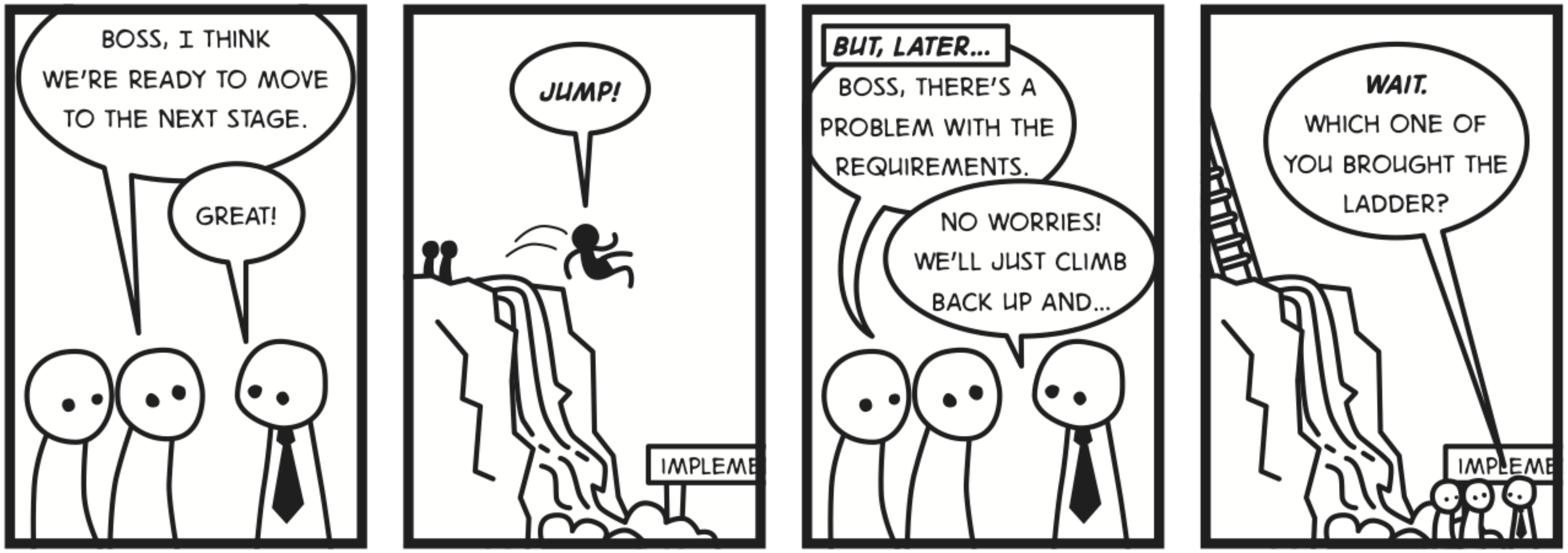
Waterfall Model


Requirements Definition
System and Software Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

Characteristics of Waterfall Model

1. The following phase **should not start until the previous phase has finished.**
2. The software process, in practice, is never a simple linear model but involves feedback from one phase to another.
3. The waterfall model is not the right process model in situations where informal team communication is possible and **software requirements change quickly**.
4. An important variant of the waterfall model is **formal system development**, where a mathematical model of a system specification is created

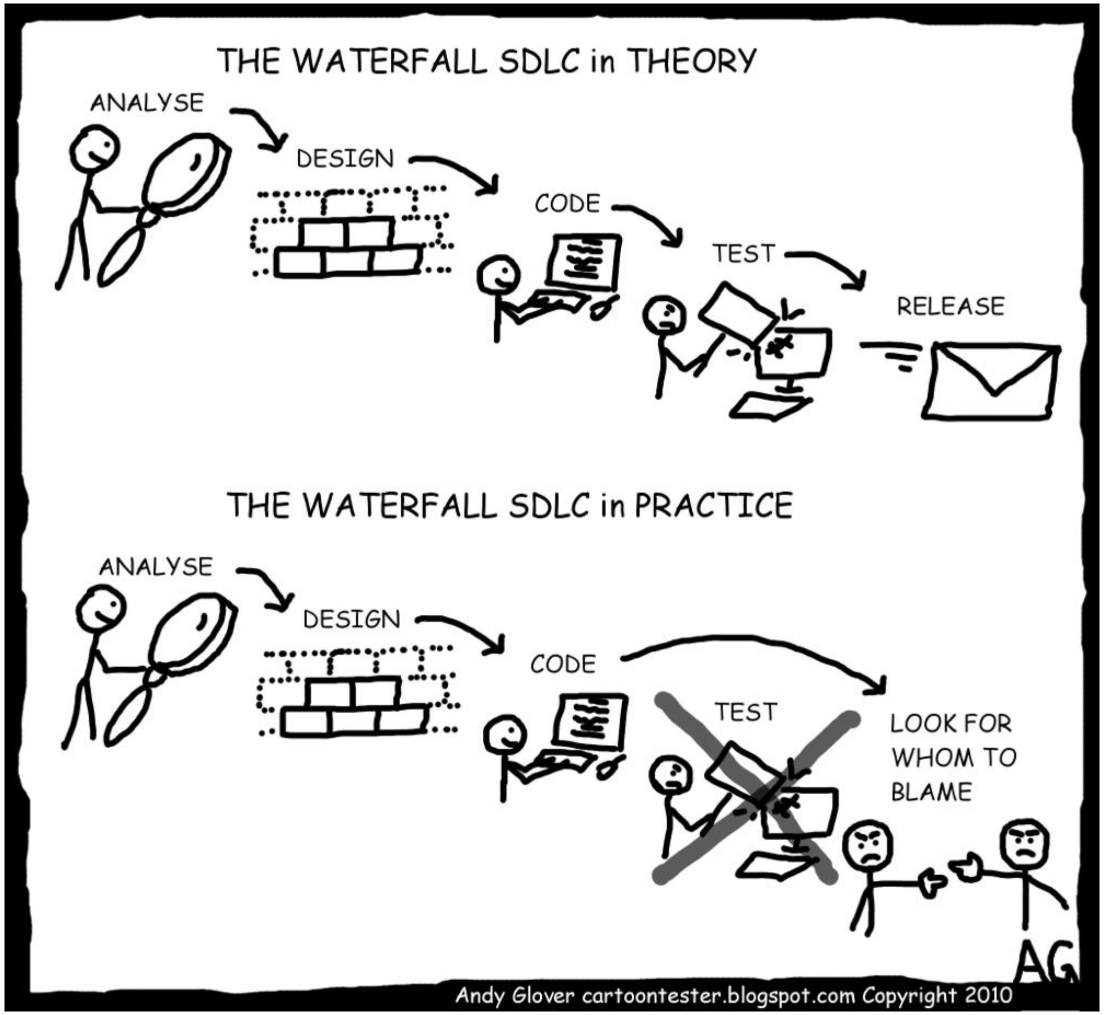
Waterfall Model


BOSS, I THINK WE'RE READY TO MOVE TO THE NEXT STAGE.
GREAT!
JUMP!
IMPLEME
BUT, LATER...
BOSS, THERE'S A PROBLEM WITH THE REQUIREMENTS.
NO WORRIES! WE'LL JUST CLIMB BACK UP AND...
WAIT. WHICH ONE OF YOU BROUGHT THE LADDER?
IMPLEME

Waterfall Strengths

- Easy to understand, easy to use
- Provides structure to inexperienced staff
- Milestones are well understood
- Sets requirements stability
- Good for management control (plan, staff, track)
- Works well when quality is more important than cost or schedule

WaterFall model


THE WATERFALL SDLC in THEORY
ANALYSE
DESIGN
CODE
TEST
RELEASE
THE WATERFALL SDLC in PRACTICE
ANALYSE
DESIGN
CODE
TEST
LOOK FOR WHOM TO BLAME
AG
Andy Glover cartoontester.blogspot.com Copyright 2010

Waterfall Disadvantages

- All requirements must be **known upfront**
- Deliverables created for each phase are considered **frozen – inhibits flexibility**
- Can give a false impression of progress
- **Does not reflect problem-solving** nature of software development – iterations of phases
- Integration is **one big bang** at the end
- Little opportunity for customer to preview the system (until **it may be too late**)

When to use the Waterfall Model

- Requirements are very well known
- Product definition is stable
- Technology is understood
- New version of an existing product
- Porting an existing product to a new platform.

When to use the Waterfall Model

- **High risk for new systems** because of specification and design problems.

- **Low risk for well-understood** developments using familiar technology.

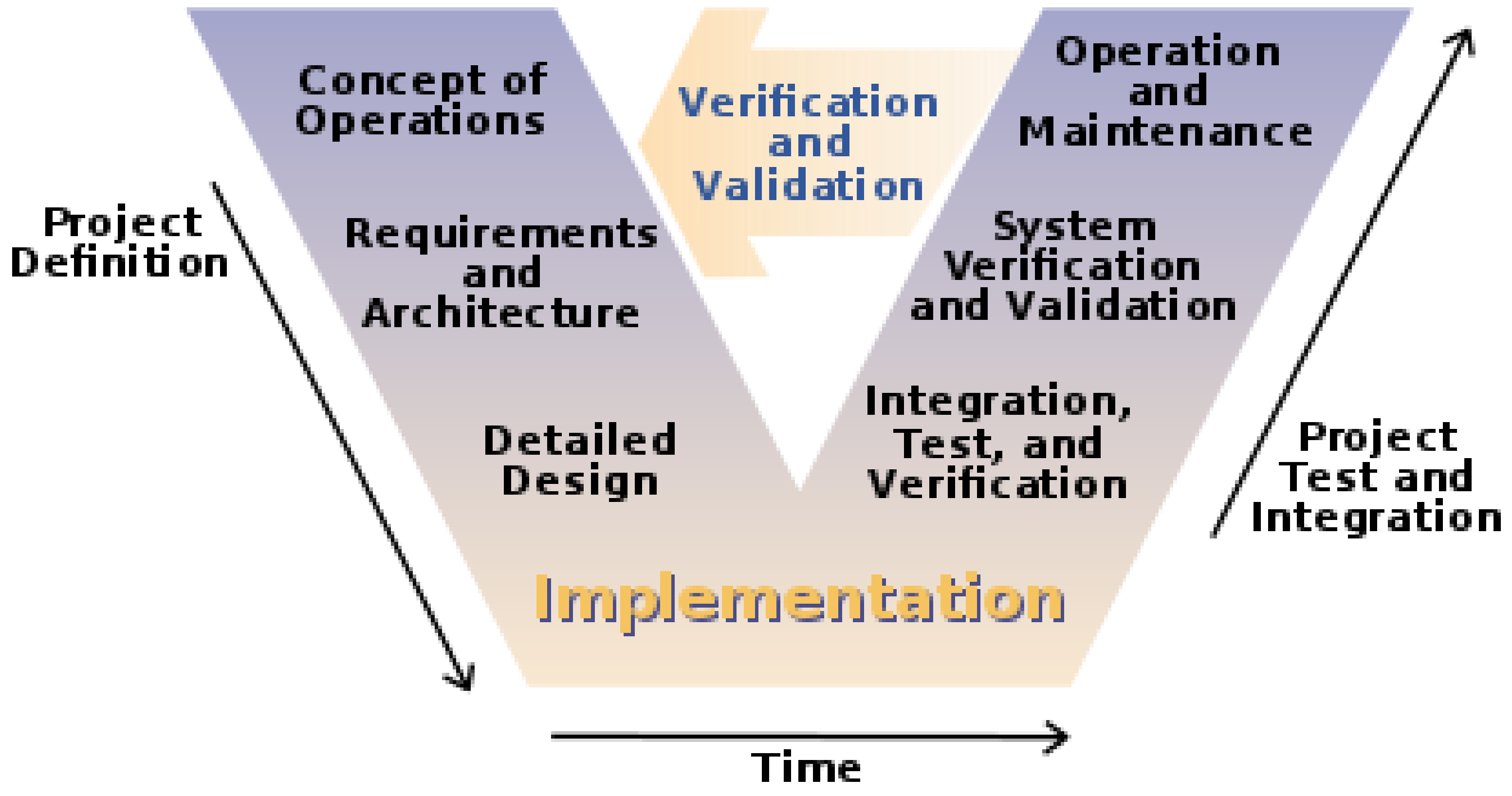
V model

- In order to put **more emphasis on the verification and validation tasks** compared to the waterfall model, **Boehm in described a V-shaped mode**l now often called the V-Model
- A variation of the waterfall model

V model

- In order to put **more emphasis on the verification and validation tasks** compared to the waterfall model,
  - Uses unit testing to verify procedural design
  - Uses integration testing to verify architectural (system) design
  - Uses acceptance testing to validate the requirements
- If problems are found during verification and validation, the left side of the V can be re-executed before testing on the right side is reenacted

V model


Concept of Operations
Requirements and Architecture
Detailed Design
Implementation
Integration, Test, and Verification
System Verification and Validation
Operation and Maintenance
Verification and Validation
Project Definition
Project Test and Integration
Time

Disadvantages of V-Model

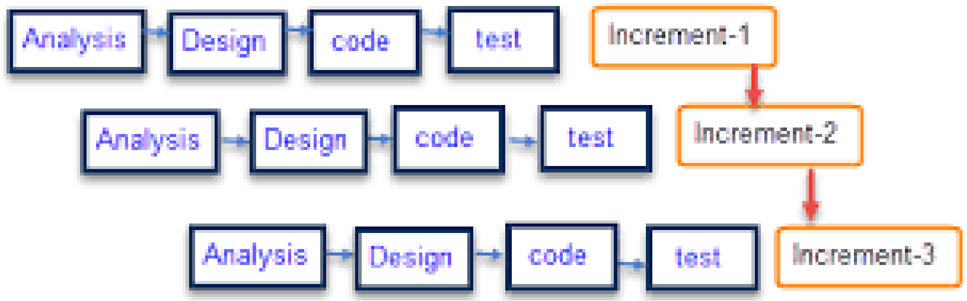
- The V-Model is emphasis on verification and validation, testing and other quality assurance tasks **start too late,**
- The V-Model does not sufficiently **support the early preparation of testing, nor the early review and analysis of design documentation**

Iterative, Incremental and Evolutionary Development

- **Increment**. A tested, deliver able version of a software product that provides new or modified capabilities
- **Iteration:**
  - – 1. process of performing a **sequence of steps repeatedly**
  - – 2. single execution of the sequence of steps in (1)

Incremental Development

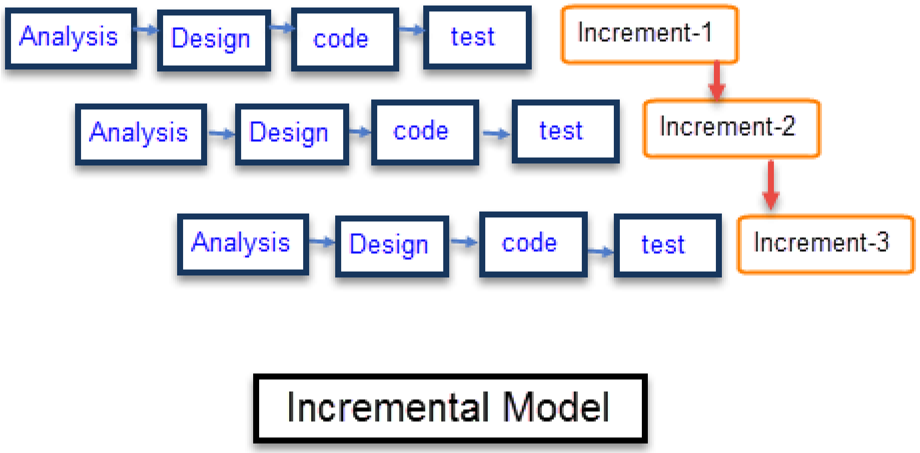
- **Incremental development**
  - Software development technique in which requirements definition, design, implementation,and testing occur in an overlapping, iterative (rather than sequential) manner, resulting in incremental completion of the overall software product



Incremental Model

Incremental software development

- The total extension of a system under development remains open;
  - it is realized in stages of expansion.
  - The first stage is the core system.
- **Each stage of expansion extends the existing system and is subject to a separate project**.
- Providing a new stage of expansion typically includes (as with iterative development) an improvement of the old components.

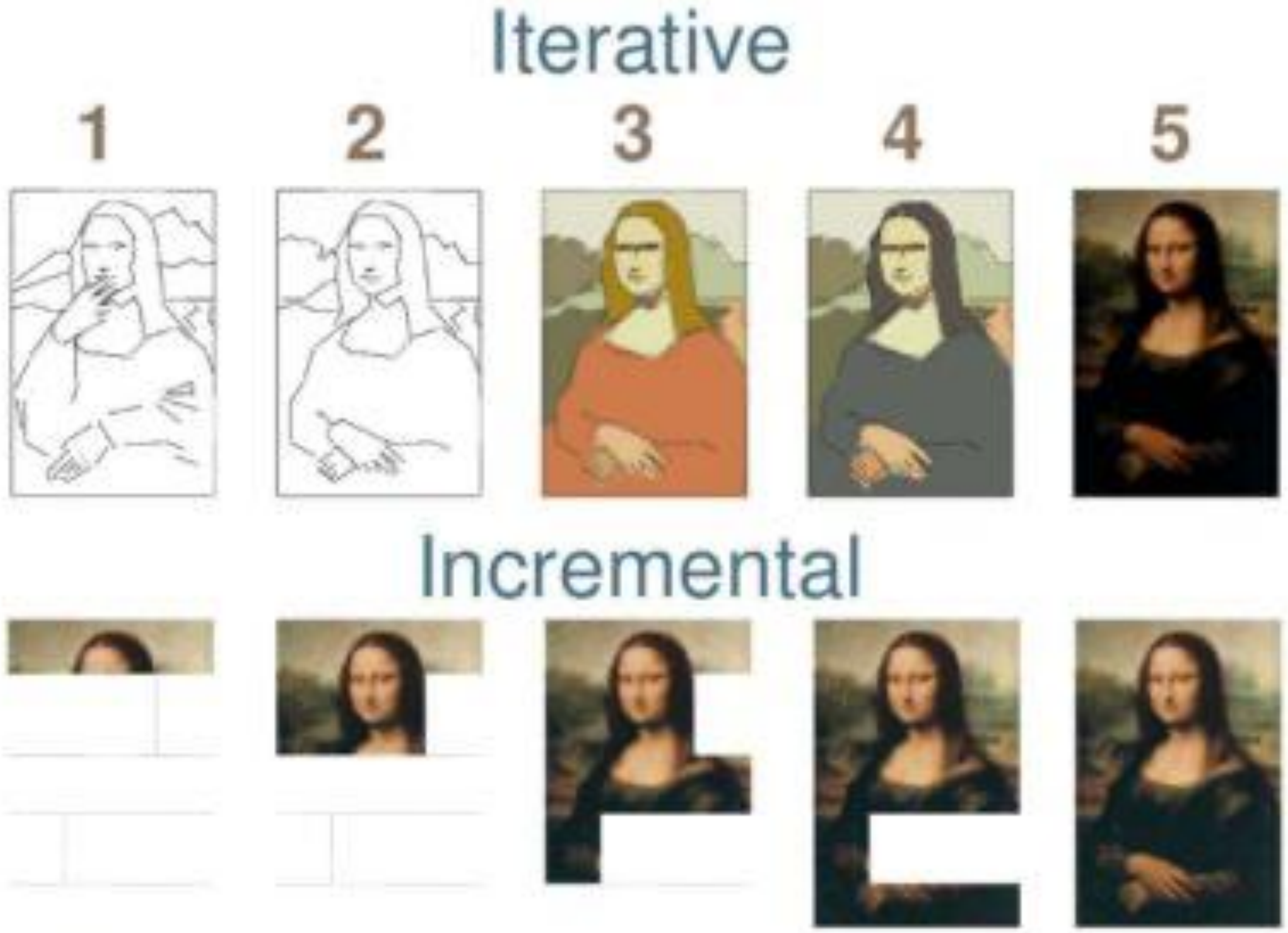


Incremental development

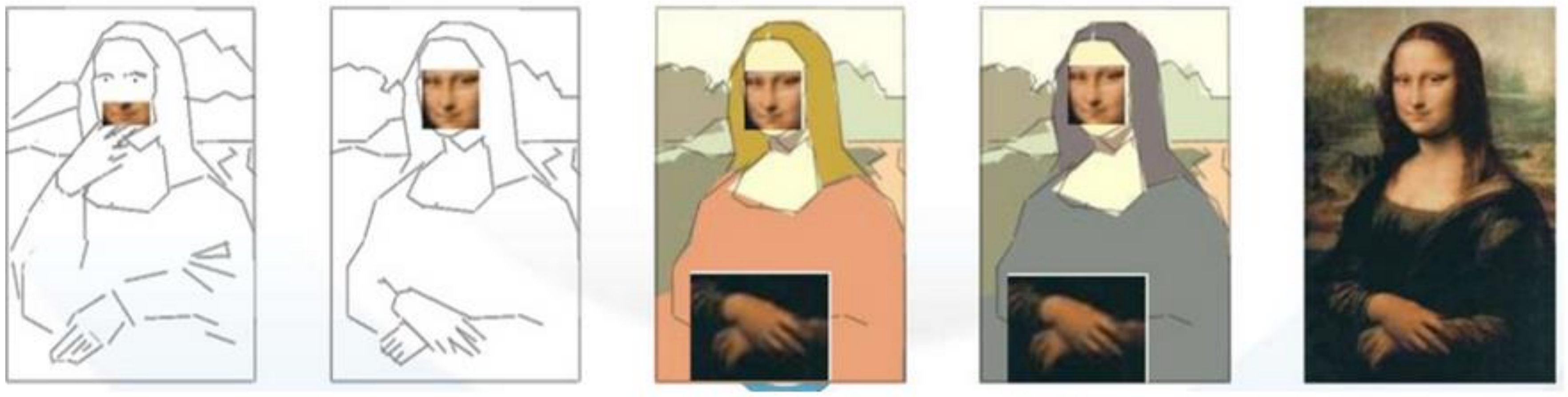
- Incremental development is based on the idea of:
  - developing an initial implementation,
  - getting feedback from users and others,
  - evolving the software through several versions until the required system has been developed

- By developing the software incrementally, it is **cheaper and easier to make changes in the software as it is being developed.**

Incremental vs. iterative Development



Incremental/ iterative Development

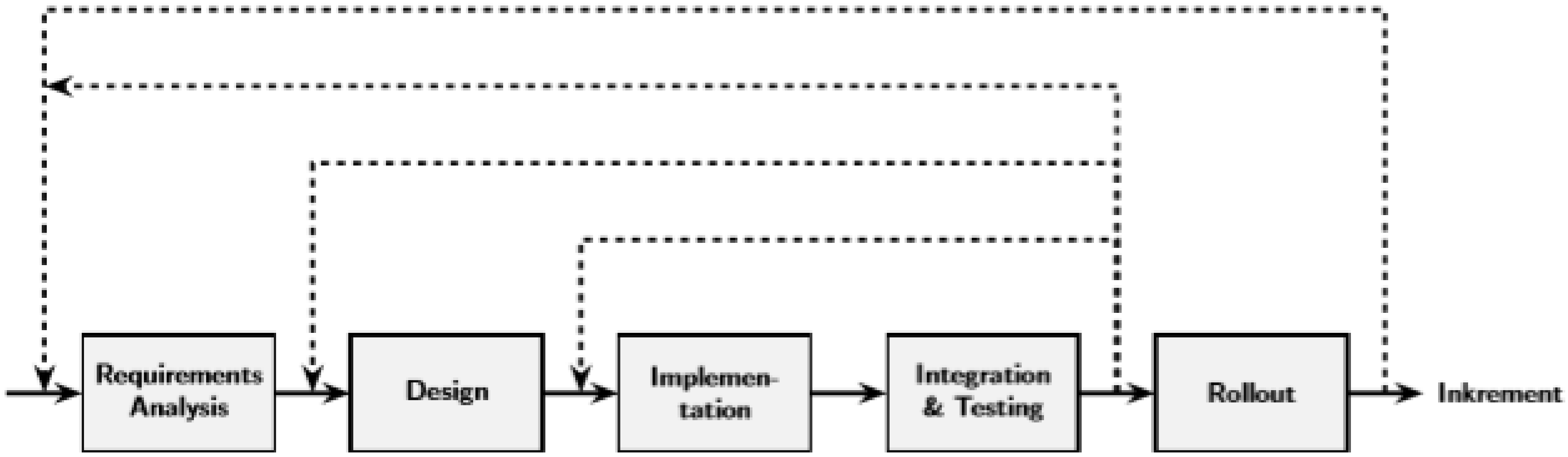


Incremental/ iterative Development

- **A project life cycle,** where the project scope is generally determined early in the project life cycle, **but time and cost estimates are routinely** modified as the project team understanding of the product increases.
  - Iterations develop the product through a series of repeated cycles, while **increments successively add to the functionality of the product**.

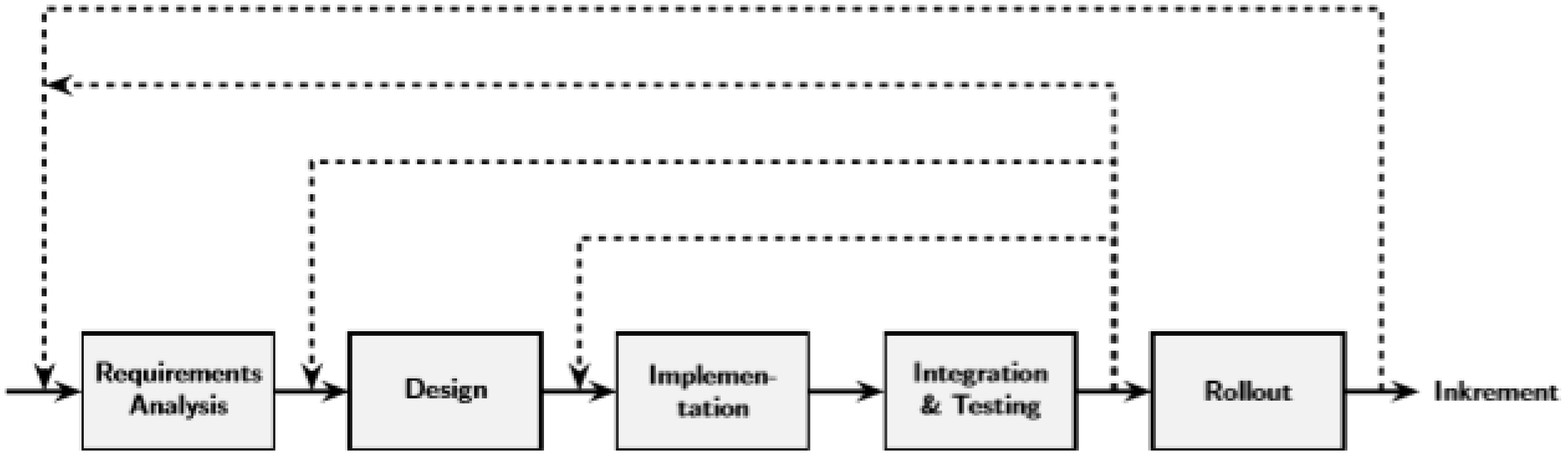
Different types of iteration

- An iteration refers to the sequence of steps which is repeatedly performed
- Alternatively, it may consist of a more **limited set of development steps,** such as iterating the requirements analysis steps, or the sequence **correction of bugs–test–delivery**



Different types of iteration

- An iteration **may or may not include the delivery** of a new version of the software product
- The **result of an iteration may be a new increment** if the iteration includes all the necessary steps for such a delivery.



Benefits of incremental development

- The main benefit of iterative development is the **reduction of risk** by collecting early feedback
- **Any new functionality can be put into production very soon** after having been implemented, thus providing the relevant benefit early on

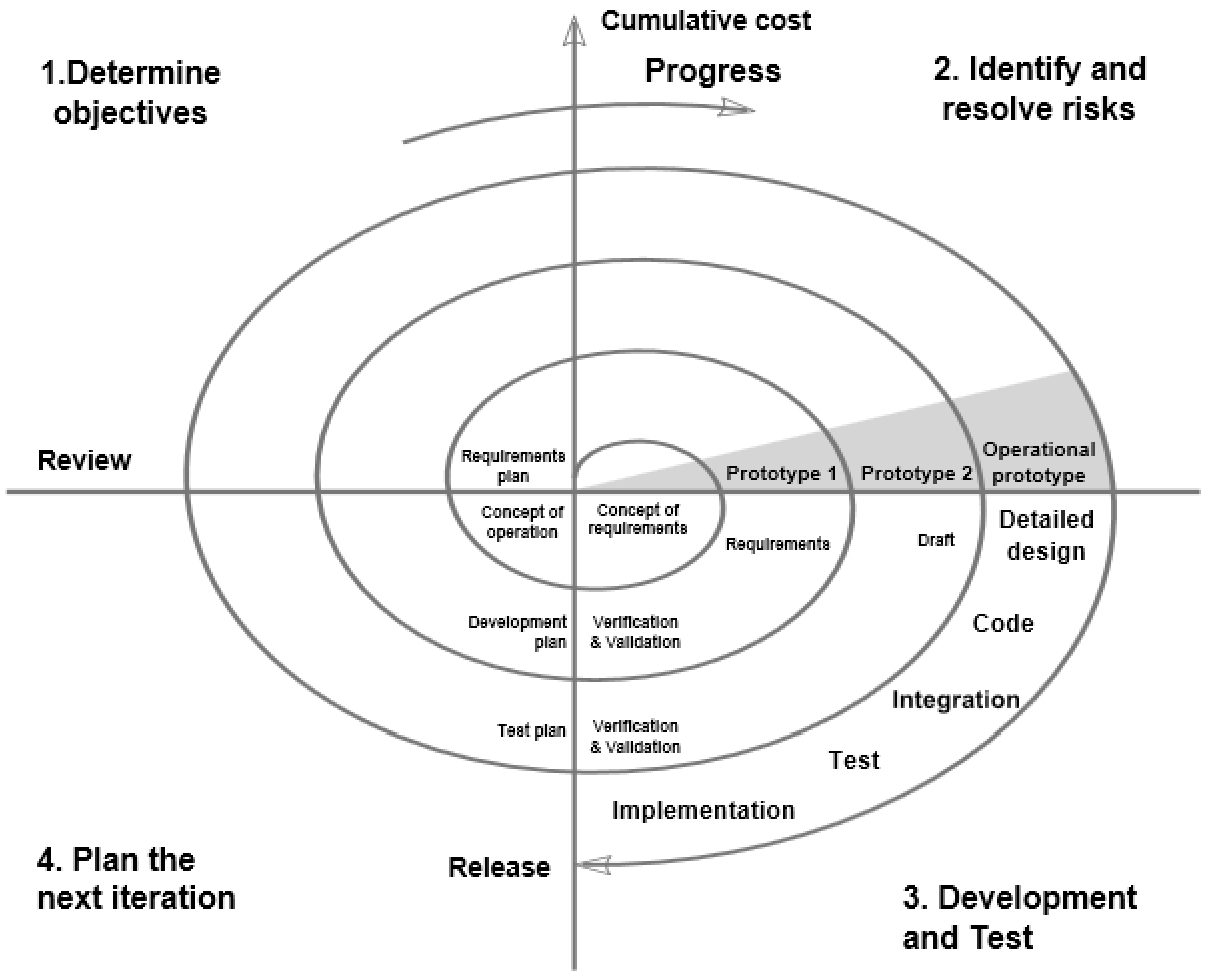
Spiral proces model

- Suggested by Barry Boehm (1988)
- Combines development activities with risk management to minimize and control risks
- Spiral model
  - Model of the software development process in which the constituent activities, typically requirements analysis, preliminary and detailed design, coding, integration, and testing, are **performed iteratively until the software is complete**

Spiral proces model

- The model is presented as a spiral in which each iteration is represented by a circuit around four major activities:
  - Plan
  - Determine goals, alternatives and constraints
  - Evaluate alternatives and risks
  - Develop and test

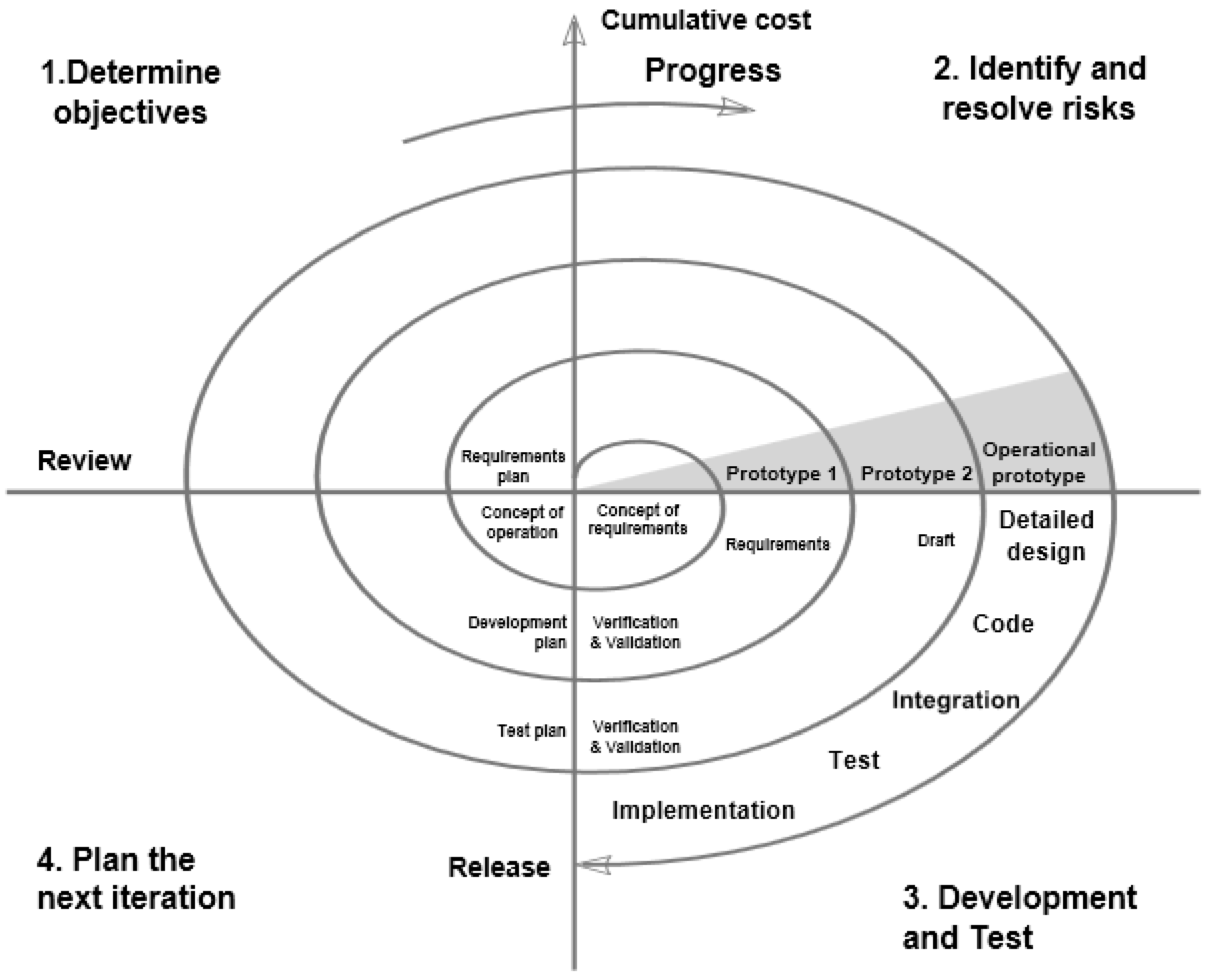
Stages of Incremental process



Incremental process
1 stage: Determine objective

- Objectives: functionality, performance, hardware/software interface, critical success factors, etc.
- Alternatives: build, reuse, buy, sub-contract, etc.
- Constraints: cost, schedule, interface, etc.

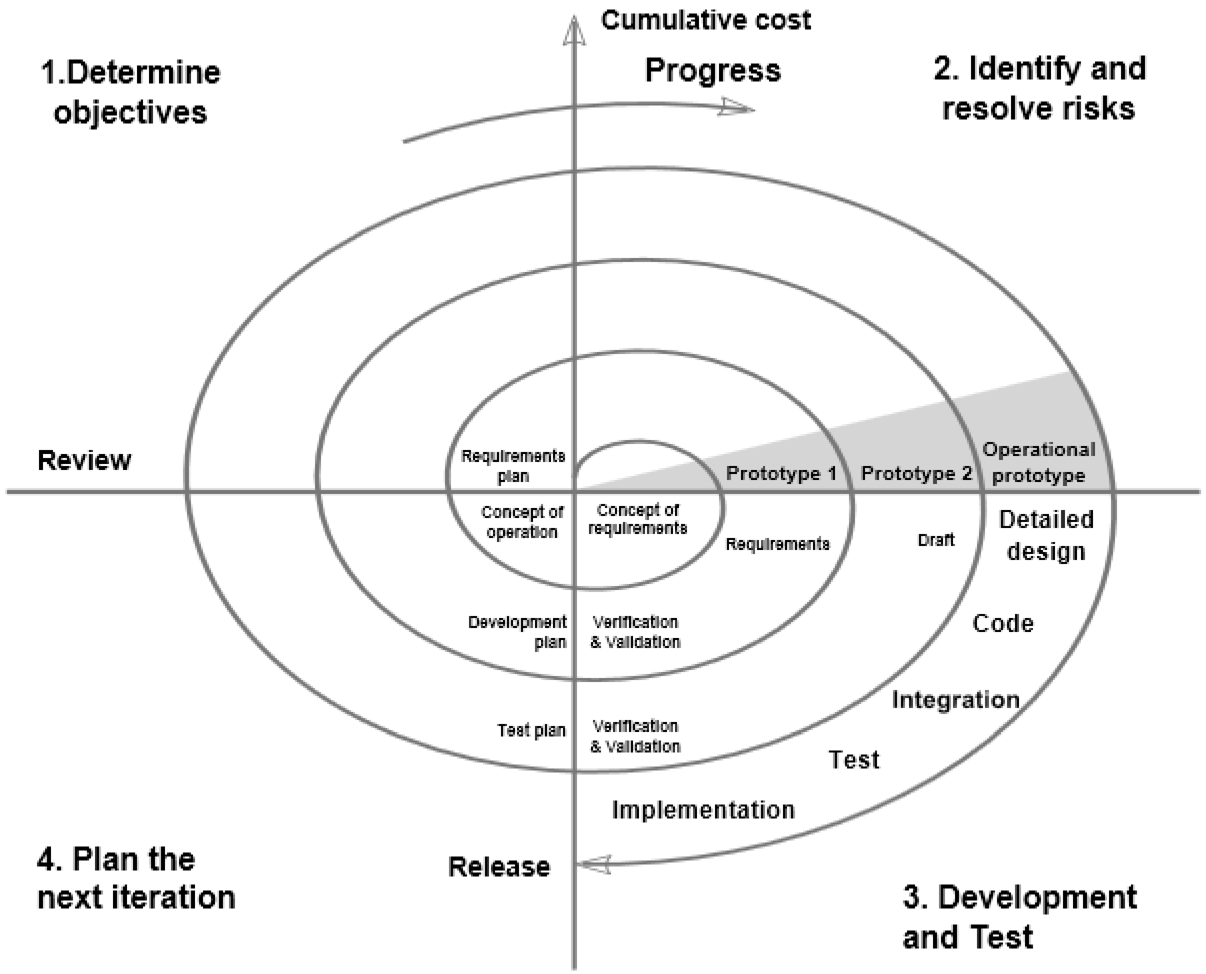
Stages of Incremental process



Incremental process
2 stage: identify and resolve risks

- Study alternatives relative to objectives and constraints
- Identify risks (lack of experience, new technology, tight schedules, poor process, etc.
- Resolve risks (evaluate if money could be lost by continuing system development

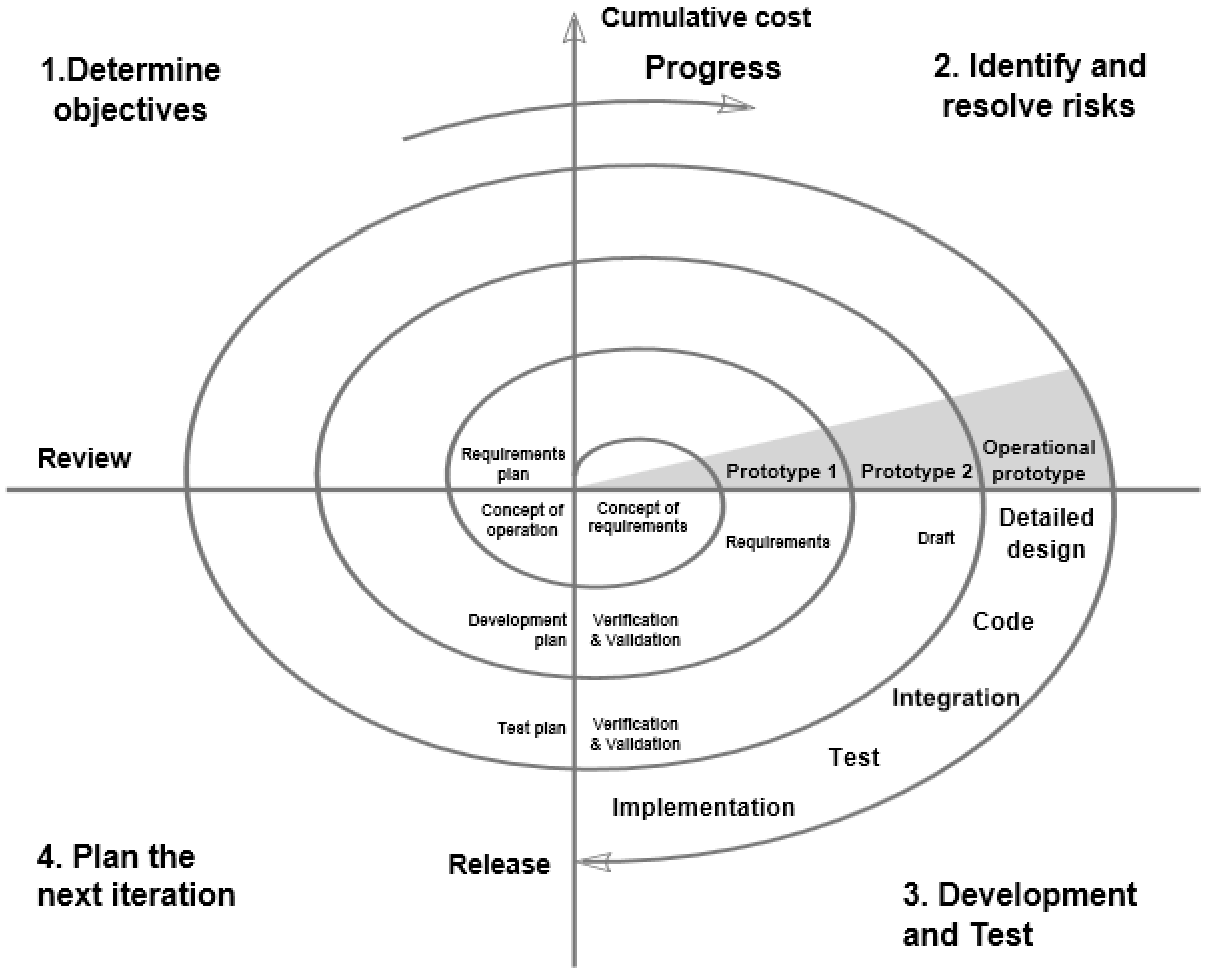
Stages of Incremental process



Incremental process
3 stage: Development and test

- Typical activities:
  - Create a design
  - Review design
  - Develop code
  - Inspect code
  - Test product

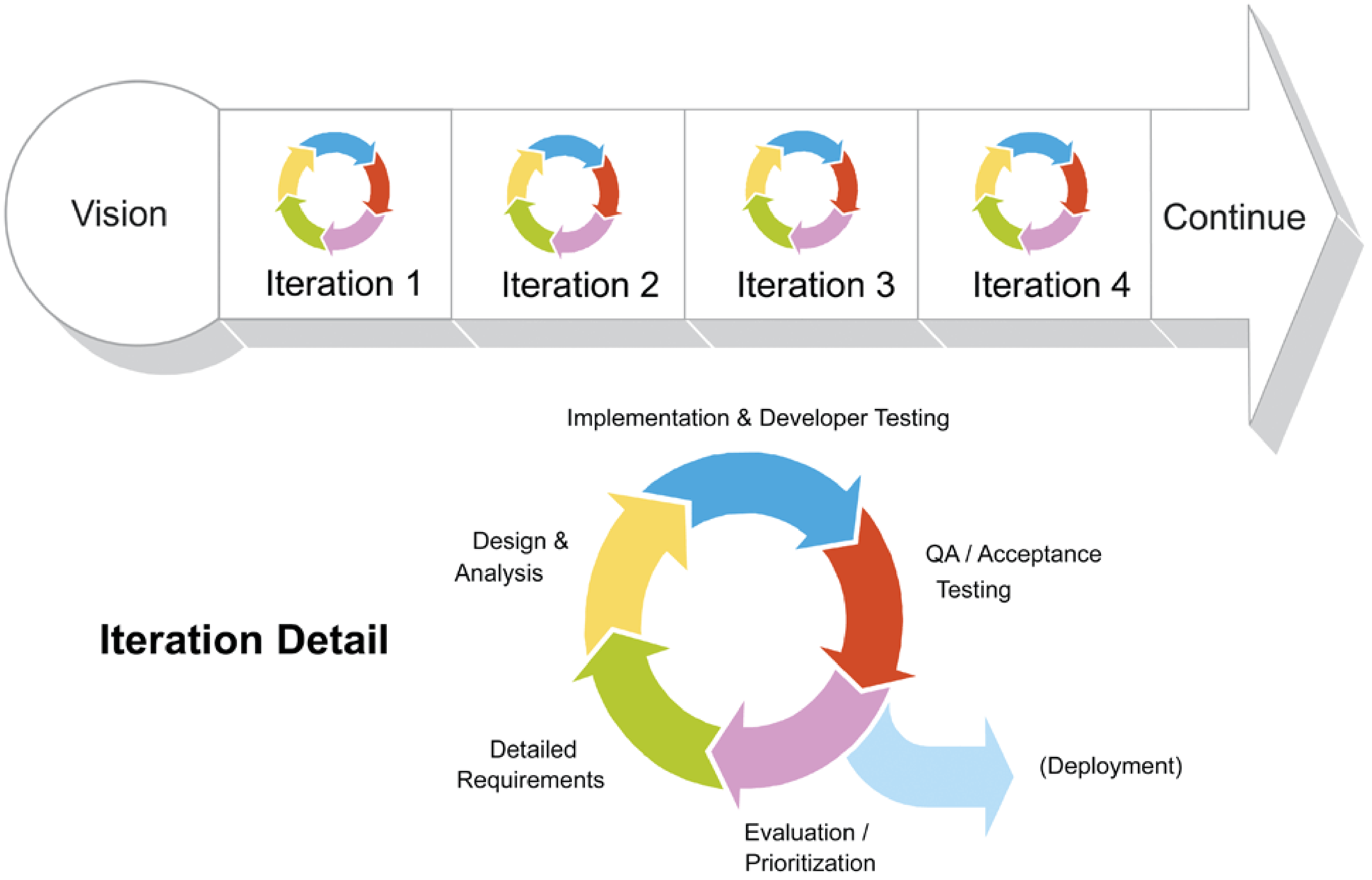
Stages of Incremental process



Incremental process
4 stage: Plan next iteration

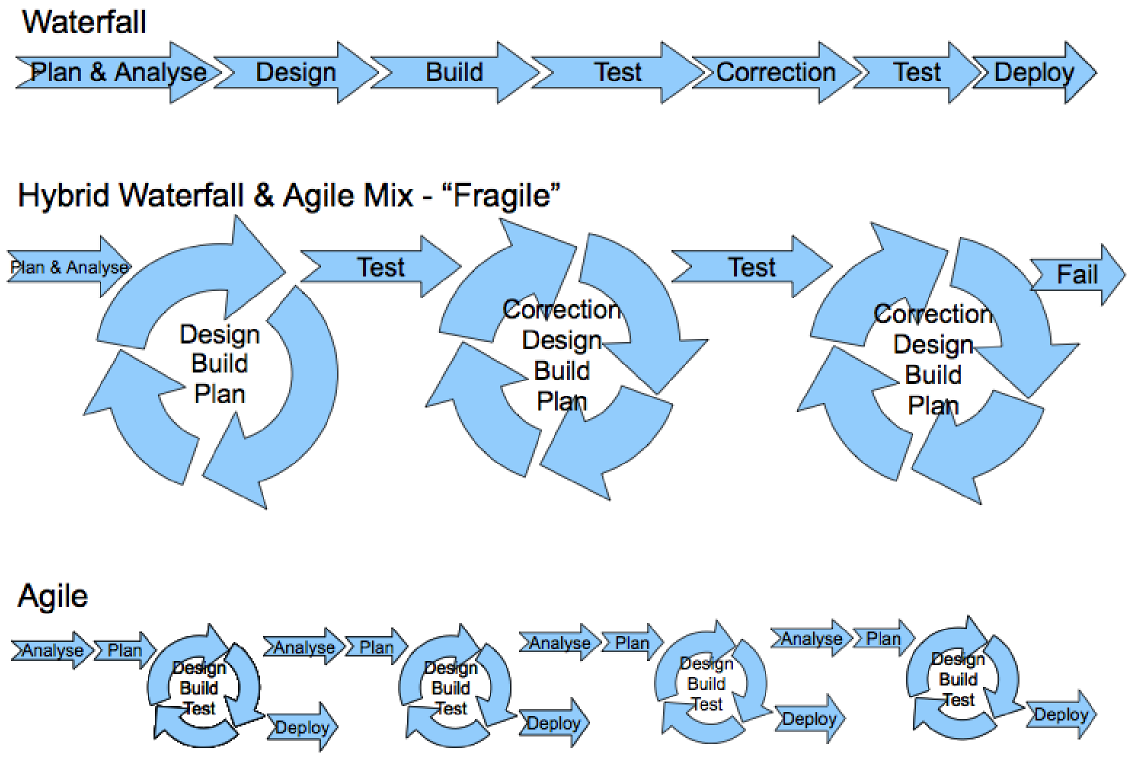
- Typical activities
  - Develop project plan
  - Develop configuration management plan
  - Develop a test plan
  - Develop an installation plan

Iterative development



Incremental vs Waterfall

- Incremental software development, which is a fundamental part of agile development methods, is better than a waterfall approach for systems whose requirements are likely to change during the development process.



Incremental process Strengths

- **Users see the system early** because of rapid prototyping tools
- **Critical high-risk functions are developed first**
- The design does not have to be perfect
- Users can be closely tied to all lifecycle steps
- **Early and frequent feedback from users**
- Cumulative costs assessed frequently

Incremental process Disadvantages

- **Time spent for evaluating risks too large** for small or low-risk projects
- Time spent planning, resetting objectives, doing risk analysis and prototyping may be excessive
- **The model is complex**
- Risk assessment expertise is required
- Developers must be reassigned during non-development phase activities
- May be **hard to define objective, verifiable milestones** that indicate readiness to proceed through the next iteration

When to use Incremental process

- When creation of a **prototype is appropriate**
- When costs and risk evaluation is important
- **For medium to high-risk projects**
- **Long-term project commitment unwise because of potential changes to economic priorities**
- Users are unsure of their needs
- Requirements are complex
- New product line
- **Significant changes are expected** (research and exploration)

Software process: Prototyping

- **Prototype - an experimental model, either functional or nonfunctional, of the system or part of the system**

- The main goal of prototypes is **to get a better understanding of a particular aspect** of the system under development

Software process: Prototyping

- A prototype is an early version of a software system that is used:
  - **to demonstrate concepts,**
  - **try out design options,**
  - **To find out more about the problem and its possible solutions**
- System prototypes allow potential users **to see how well the system supports their work.**

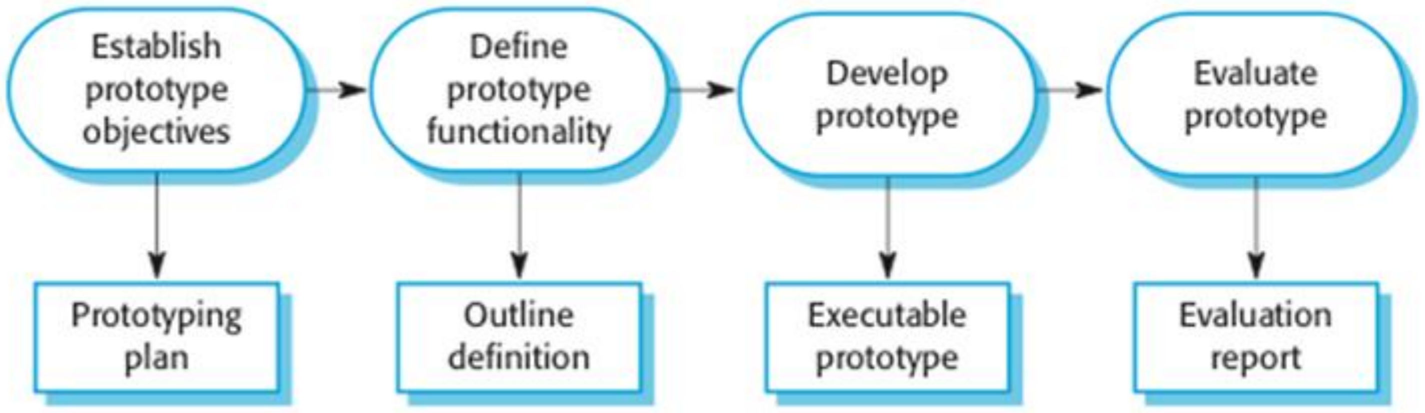
Exploratory prototypes

- An exploratory prototype is used in the initial phase of development, with the main purpose:
  - of **helping to identify the requirements** on the system to be developed.
  - It is above all a communication medium that helps to get a common **understanding about expected system properties between developers or analysts** on the one side, and stakeholders or users on the other.

Experimental prototypes

- An experimental prototype is used some what later, once at least an initial set of requirements has been identified.
- Its main purpose is **to experiment with the current solution** (initial requirements, design, technical solution, etc.) in order to validate it, e.g. checking the technical feasibility.

The process of prototype development



Why software prototype?

1. In the requirements engineering process, a **prototype can help with the elicitation and validation of system requirements.**

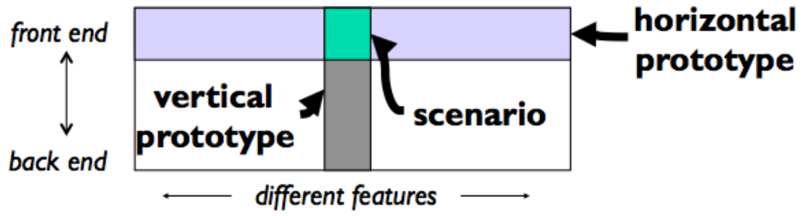
2. In the system design process, a **prototype can be used to explore software solutions** and in the development of a user interface for the system.

Basic Steps of Prototyping

1. Identify basic requirements
   - Including input and output info
   - Details (e.g., security) generally ignored
2. Develop initial prototype
   - UI first
3. Review
   - Customers/end –users review and give feedback
4. Revise and enhance the prototype & specs
   - Negotiation about scope of contract may be necessary

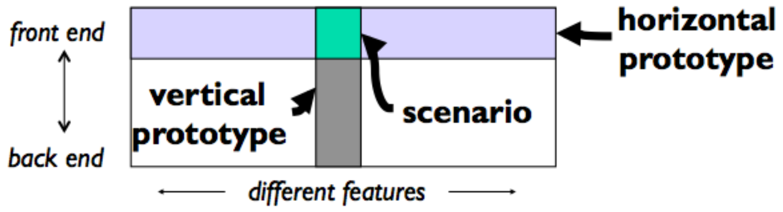
Dimensions of prototyping

- Horizontal prototype
  - Broad view of entire system/sub-system
  - Focus is on user interaction more than low-level system functionality (e.g. , database access)
  - Useful for:
    - Confirmation of UI requirements and system scope
    - Demonstration version of the system to obtain buy-in from business/customers
    - Develop preliminary estimates of development time, cost, effort

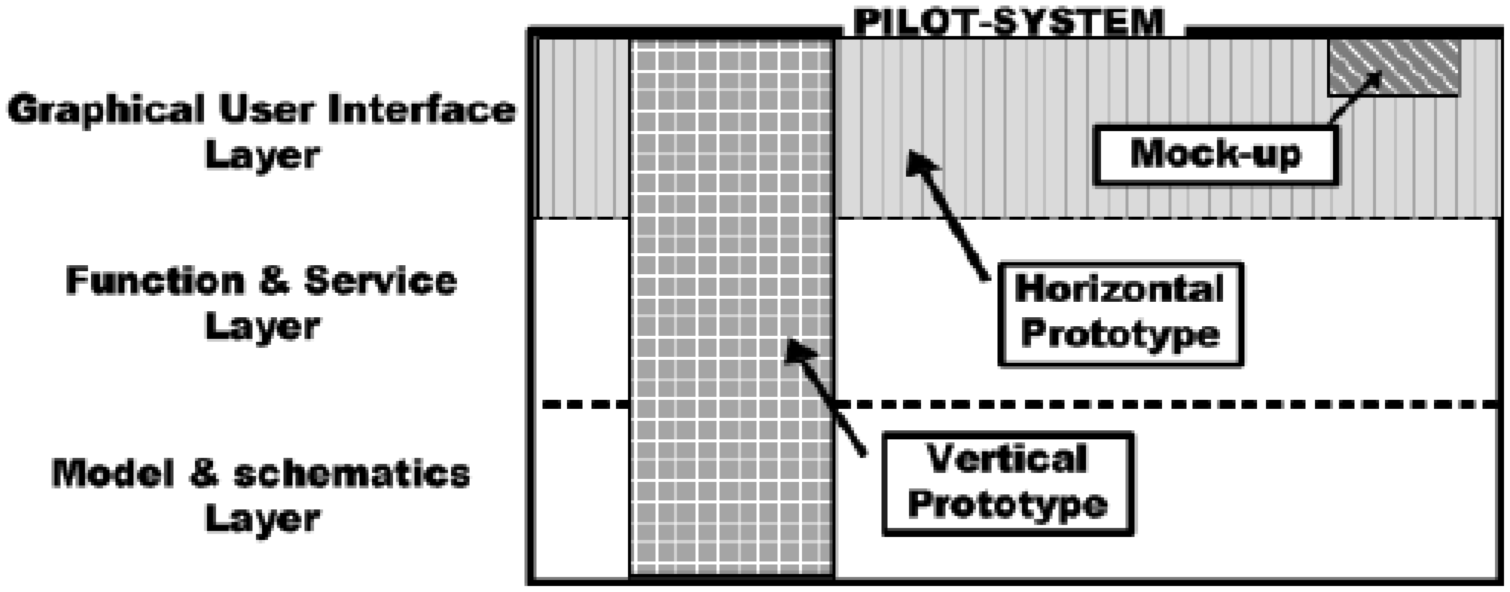


Dimensions of Prototyping

- Vertical prototype
  - More complete elaboration of a single sub-system or function
  - Useful for:
    - Obtaining detailed requirements for a given function
    - Refining database design
    - Obtaining info on system interface needs
    - Clarifying complex requirements by drilling down to actual system functionality



Horizontal vs. vertical prototyping



Horizontal vs. vertical prototyping

Horizontal prototype:

- broad coverage of features
- less detail for each feature
- less realistic evaluation

Vertical prototype:

- fewer features
- more detail for each feature
- more realistic evaluation

Prototyping advantages

- Reduced time and cost
  - Can improve the quality of requirements and specifications provided to developers
    - Early determination of what the user really wants can result in faster and less expensive software
- Improved/increased user involvement
  - User can see and interact with the prototype, allowing them to provide better/more complete feedback and specs
  - Misunderstandings/miscommunications revealed
  - Final product more likely to satisfy their desired look/feel/performance

Disadvantages of prototyping 1

- Insufficient analysis
  - Focus on limited prototype can distract developers from analyzing complete project
  - May overlook better solutions
  - Conversion of limited prototypes into poorly engineered final projects that are hard to maintain
  - Limited functionality may not scale well if used as the basis of a final deliverable
    - May not be noticed if developers too focused on building prototype as a model

Disadvantages of prototyping 2

- User confusion of prototype and finished system
  - Users can think that a prototype (intended to be thrown away) is actually a final system that needs to be polished
    - Unaware of the scope of programming needed to give prototype robust functionality
  - Users can become attached to features included in prototype for consideration and then removed from final specification

Disadvantages of prototyping 3

- Developer attachment to prototype
  - If spend a great deal of time/effort to produce, may become attached
  - Might try to attempt to convert a limited prototype into a final system
    - Bad if the prototype does not have an appropriate underlying architecture

Disadvantages of prototyping 4

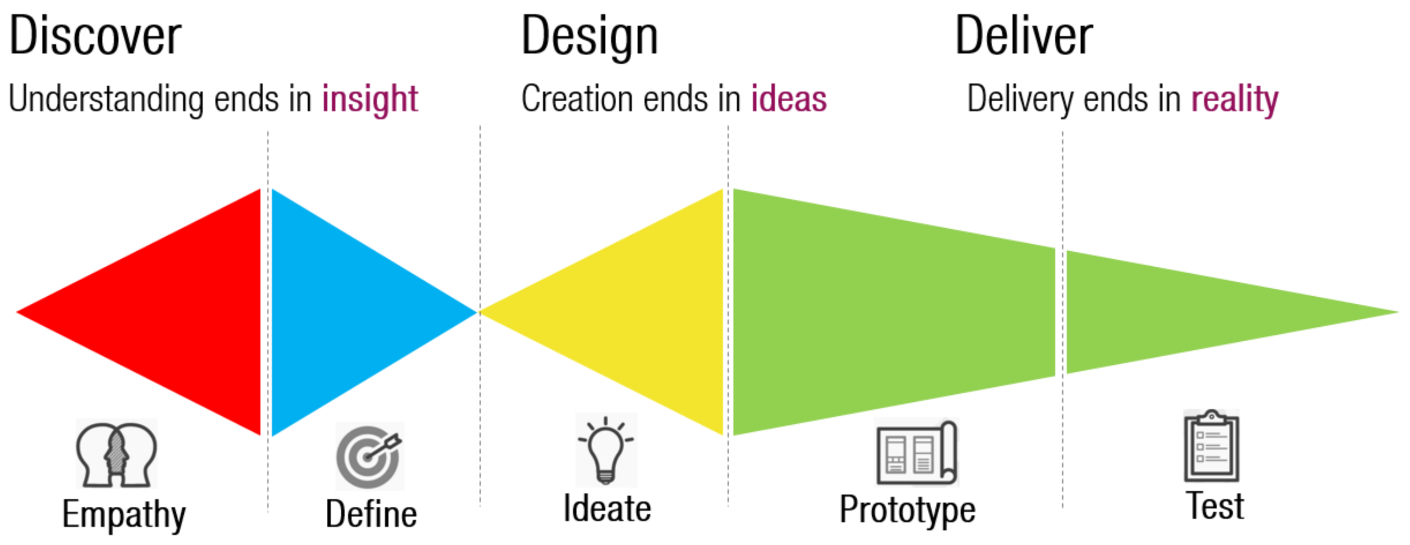
- Excessive development time of the prototype
  - Prototyping supposed to be done quickly
  - If developers lose sight of this, can try to build a prototype that is too complex
  - For throw away prototypes, the benefits realized from the prototype (precise requirements) may not offset the time spent in developing the prototype – expected productivity reduced
  - Users can be stuck in debates over prototype details and hold up development process

Disadvantages of prototyping 5

- Expense of implementing prototyping
  - Start up costs of prototyping may be high
  - Expensive to change development methodologies in place (re-training, re-tooling)
  - Slow development if proper training not in place
    - High expectations for productivity unrealistic if insufficient recognition of the learning curve
  - Lower productivity can result if overlook the need to develop corporate and project specific underlying structure to support the technology

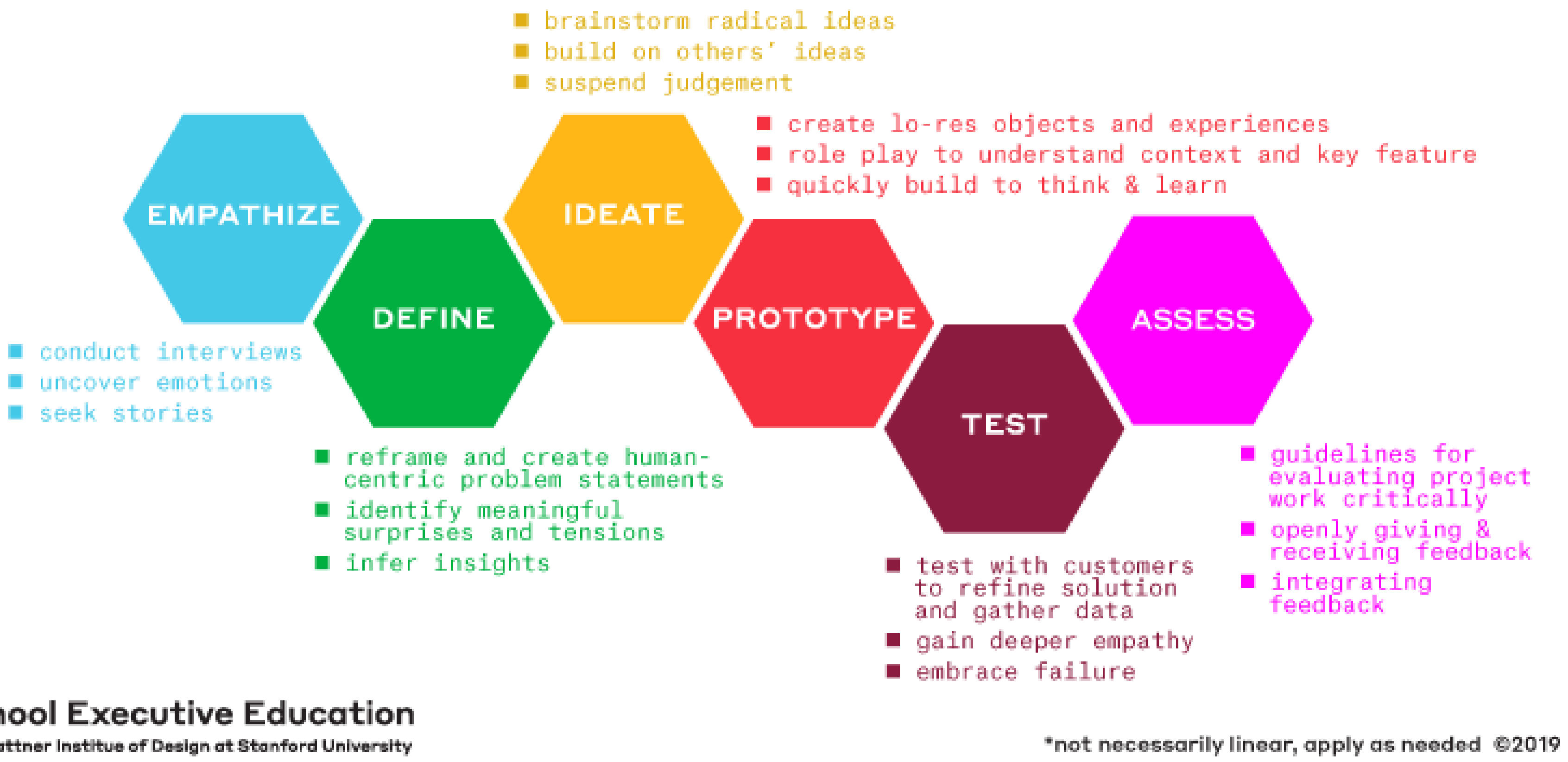
Design thinking

- The main purpose of design thinking is to get an understanding of **what users really need**, and to provide innovative solutions to that need.



https://dschool-old.stanford.edu/sandbox/groups/designresources/wiki/36873/attachments/74b3d/ModeGuideBOOTCAMP2010L.pdf

Design Thinking Process Diagram*



brainstorm radical ideas
build on others' ideas
suspend judgement
create lo-res objects and experiences
role play to understand context and key feature
quickly build to think & learn
EMPATHIZE
IDEATE
DEFINE
PROTOTYPE
ASSESS
TEST
conduct interviews
uncover emotions
seek stories
reframe and create human-centric problem statements
identify meaningful surprises and tensions
infer insights
test with customers to refine solution and gather data
gain deeper empathy
embrace failure
guidelines for evaluating project work critically
openly giving & receiving feedback
integrating feedback
d.school Executive Education
Hasso Plattner Institue of Design at Stanford University
*not necessarily linear, apply as needed ©2019

https://www.interaction-design.org/literature/topics/design-thinking

https://voltagecontrol.co/a-step-by-step-guide-to-the-design-thinking-process-d0a95a28b9db

Steps of Design thinking

1. **Empathise with users to understand them**, their needs, and the problems to be solved
2. **Define the needs of the users** and the resulting problem to be solved
3. **Ideate is about generating ideas** to solve the problems, deliberately taking a very wide view of possible solutions and typically using some form of brainstorming techniques
4. **Prototype the best problem solutions found**
5. **Test the prototype solution** to refine the prototype solution iteratively.

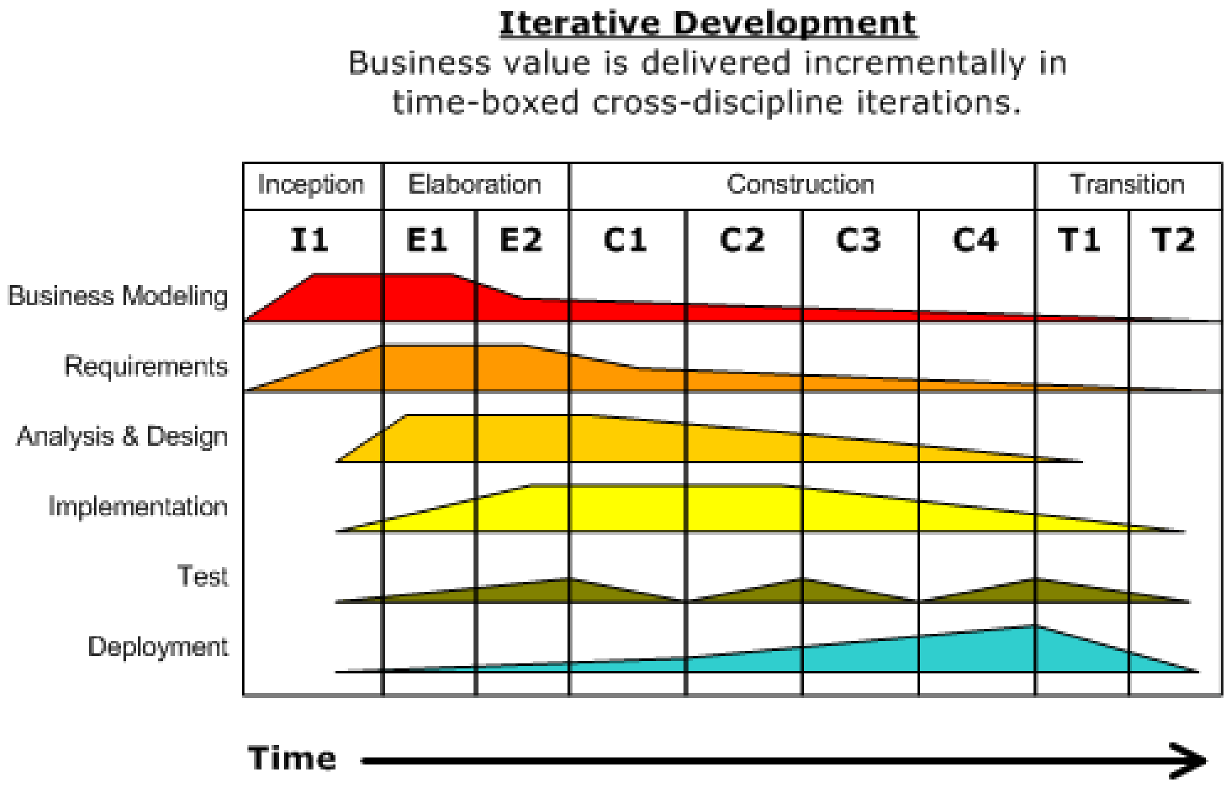
Map–Sketch–Decide–Prototype–Test

The Rational Unified Process

- This is Combined Software Life Cycle and Process Models
- With the growing importance **of object-oriented development**, a number of different methods for object-oriented analysis, design and programming were published in the early 1990s by different authors.
- RUP is a **component-based model**

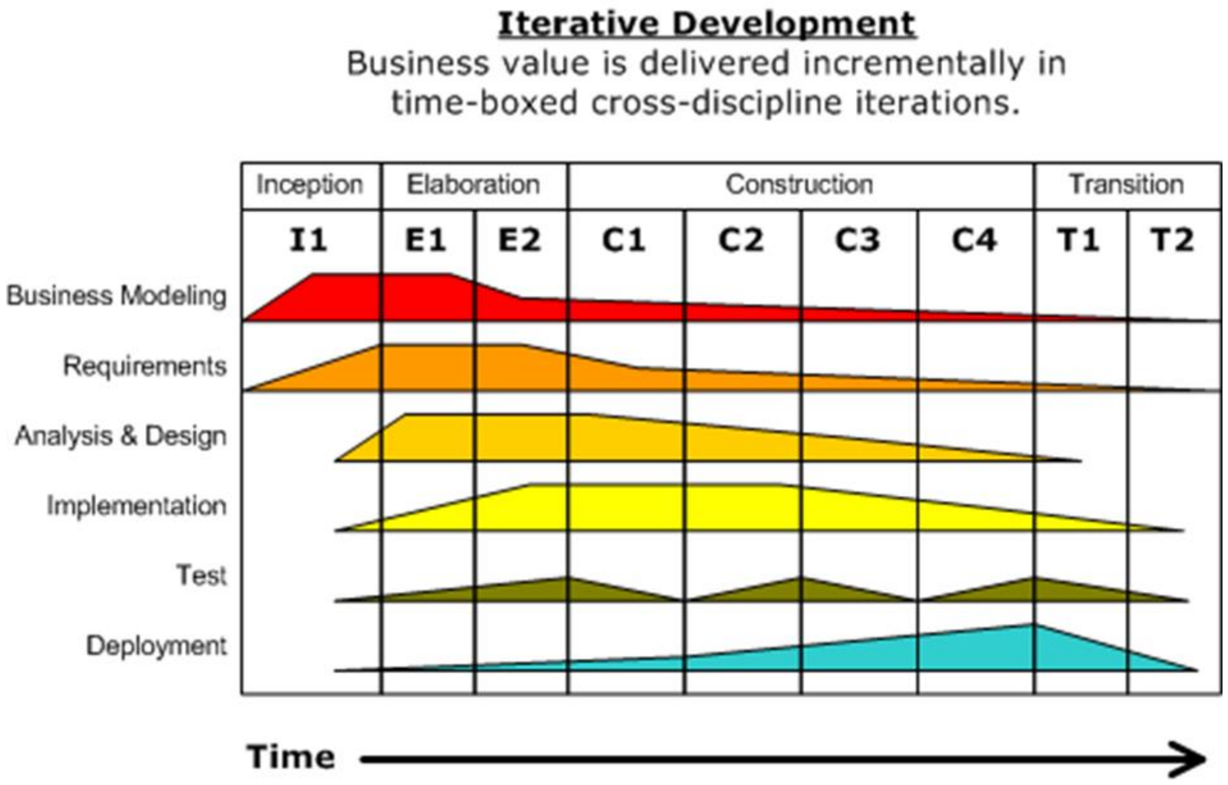
The Rational Unified Process

- RUP describes an iterative, incremental approach to software development,
- Is not an agile approach



The Rational Unified Process

- The final version 7.0 was published in 2005.
- Today, RUP is still fairly well-known but no longer supported by Rational or its parent company IBM.



Other Software Process Models

- Formal System Development:
  - Transforms a mathematical based specification through different representation → executable program.
  - Program correctness is easy to demonstrate, as the transformations preserve the correctness.
- Reuse-Oriented Development:
  - Concentrates on integrating new system with existing components/systems.
  - Growing rapidly as development cost increase.

Why are there so many models?

- The choice of a **model depends on the project circumstances and requirements**
- A good choice of a model can result in a vastly more productive environment than a bad choice
- **A cocktail of models is frequently used in practice to get the best of all worlds** – models are often combined or tailored to environment

- “Models” are as often descriptive as they are prescriptive

Parnas and Clements. A rational design process: How and why to fake it. *IEEE Trans. Software Eng. 12*, 2 (Feb 1986), 251-257.

The “best” model depends on…

- The task at hand
- Risk management
- Quality / cost control
- Predictability
- Visibility of progress
- Customer involvement and feedback
- Team experience
- …