Software Design

Models, Tools & Processes *

Lecture 1: Software Design and Software Development Process

Cecilia Mascolo

* Thanks to Alan Blackwell and Jim Arlow for letting me use some of their slides.

About Me

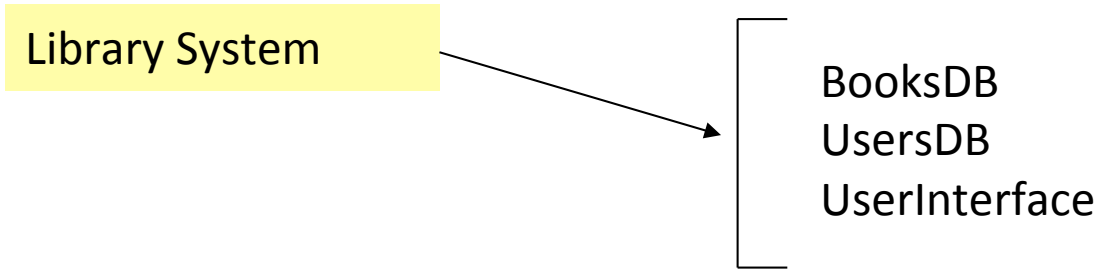
- Reader in Mobile Systems
  - Systems Research Group
- Research on Mobile, Social and Sensor Systems
- More specifically, mobility modelling
  - Instrumentation (sensing and mobile sensing)
  - Analysis (social and complex networks)
  - Exploitation (eg, recommender systems)

twitter

foursquare

Software Design

- Software Design is about modelling software systems
- “A system is an organised or complex whole: an assemblage or combination of things or parts forming a complex or unitary whole.” (Kast & Rosenzweig)
- “A system is a set of interrelated elements” (Ackoff)



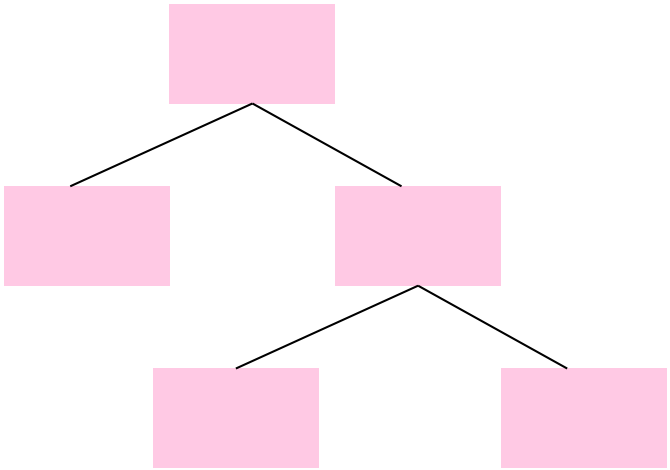
Everyday Words

"it is in the system", "the system failed", "rage against the system", "you can't buck the system", "the system is down", "the economic system", "in-car stereo system", "biological system", "paperwork system", "the financial environment", "closed system", "open system", "dynamic system", "in equilibrium"

We use these "system words" a lot. What do they mean

Organisation

- The predominant mode of organisation is hierarchical. Systems are composed of sub-systems, sub-systems are composed of sub-sub-systems and so on.



- In very complex cases we talk of “systems of systems”

Example: Robot and its components

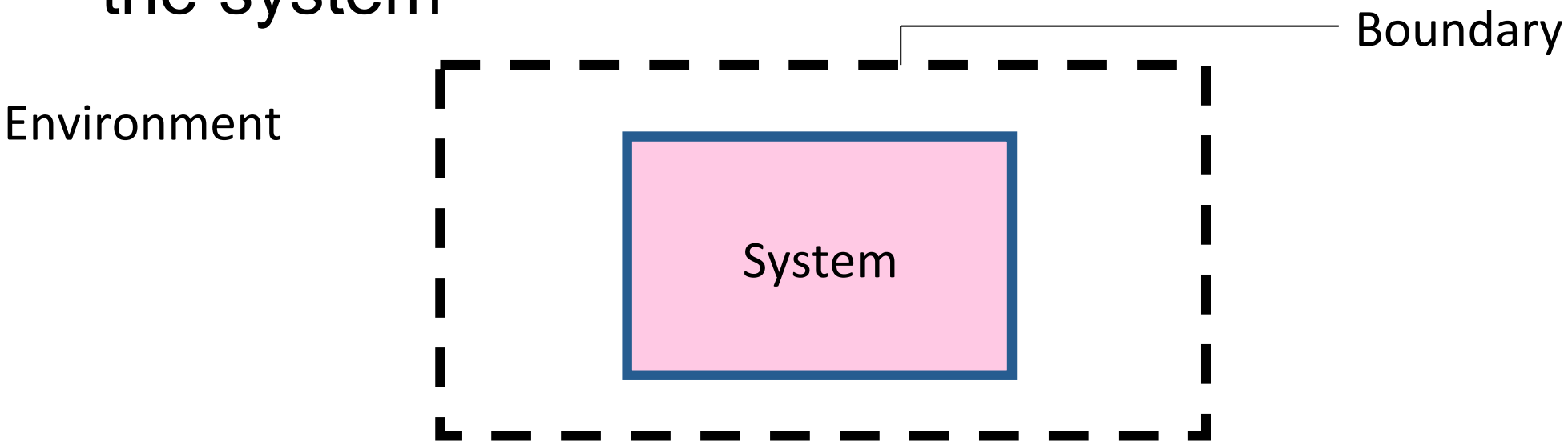
State

- The state of a system at a moment in time is the set of values of *relevant* properties which that system has at that time.
- Any system has an unlimited set of properties - only some of which are relevant for any particular set of purposes.

Examples: mass=10g, colour=red

Environment

- The environment of a system is the set of elements (and their relevant properties) which are NOT part of the system - but a change in any of which can produce a change in the state of the system



Environment

- The choice of the boundary is subjective. Different people may divide a domain of discourse into different systems and environments.

An architect views a house as the system comprised of mechanical, electrical, heating and water sub-systems. The electrical supply system is in the environment.
The electrician may view the electrical sub-system together with the electrical supply system as the system with the house as its environment.

Environment

- Setting boundaries is very important when analysing and designing a system. It limits your investigation and problem solving "space".

Example: Imagine you are designing a new electrical car. Are the repair shops, refuelling stations and parts supply part of the system you are designing or not? How much Do they affect the design of the car? Can you change them? How much would changes in them affect your design (robustness)?

Closed and Open

- Systems can be considered closed or open.
- Closed systems do not interact with their environment.
- Open systems have a dynamic relationship with their environment, receiving inputs, transforming these inputs and exporting outputs.

Inputs and Outputs

- A general view of a system

Modelling

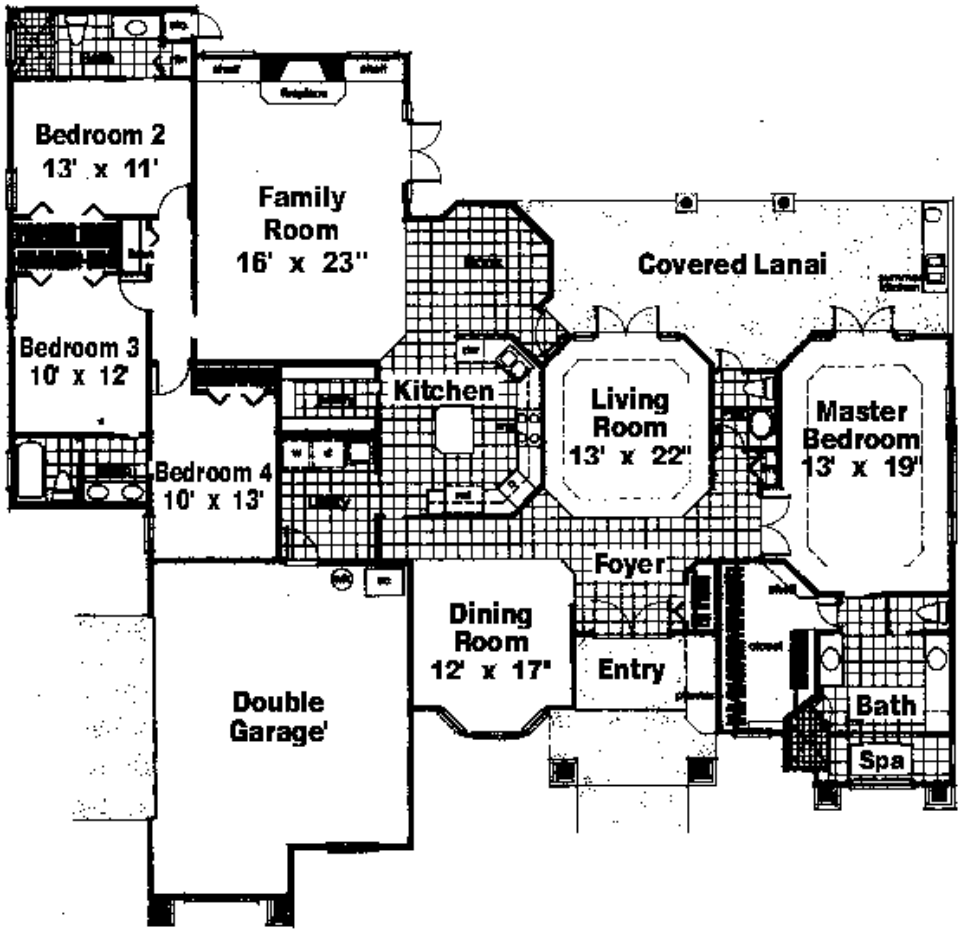
- Modelling a system means identifying its main characteristics, states and behaviour using a notation
- You “modelled” the Library System using Java
  - …a very detailed model…
- There are better techniques to build models

Model

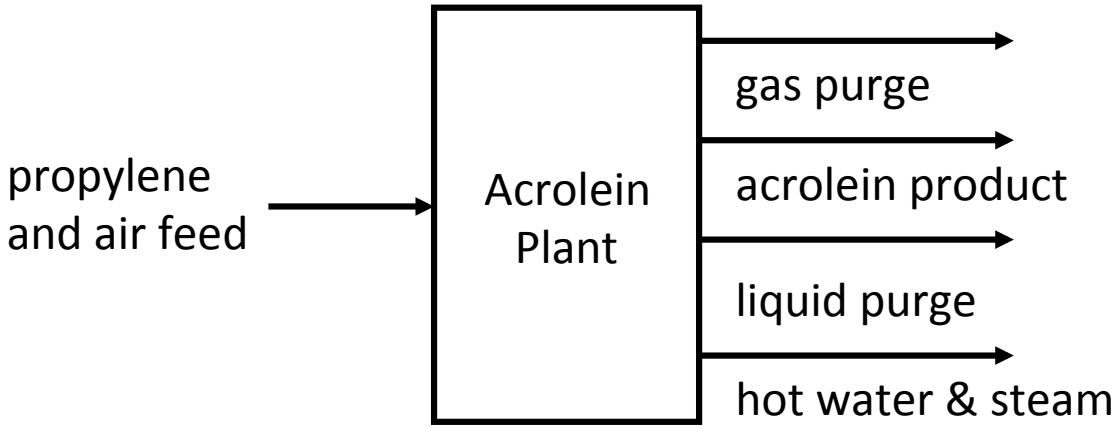
- A *model* is a description from which detail has been removed in a systematic manner and for a particular purpose.
- A simplification of reality intended to promote understanding.
- Models are the most important engineering tool, they allow us to understand and analyse large and complex problems.

Examples: an architectural plan,
a chemical plant diagram

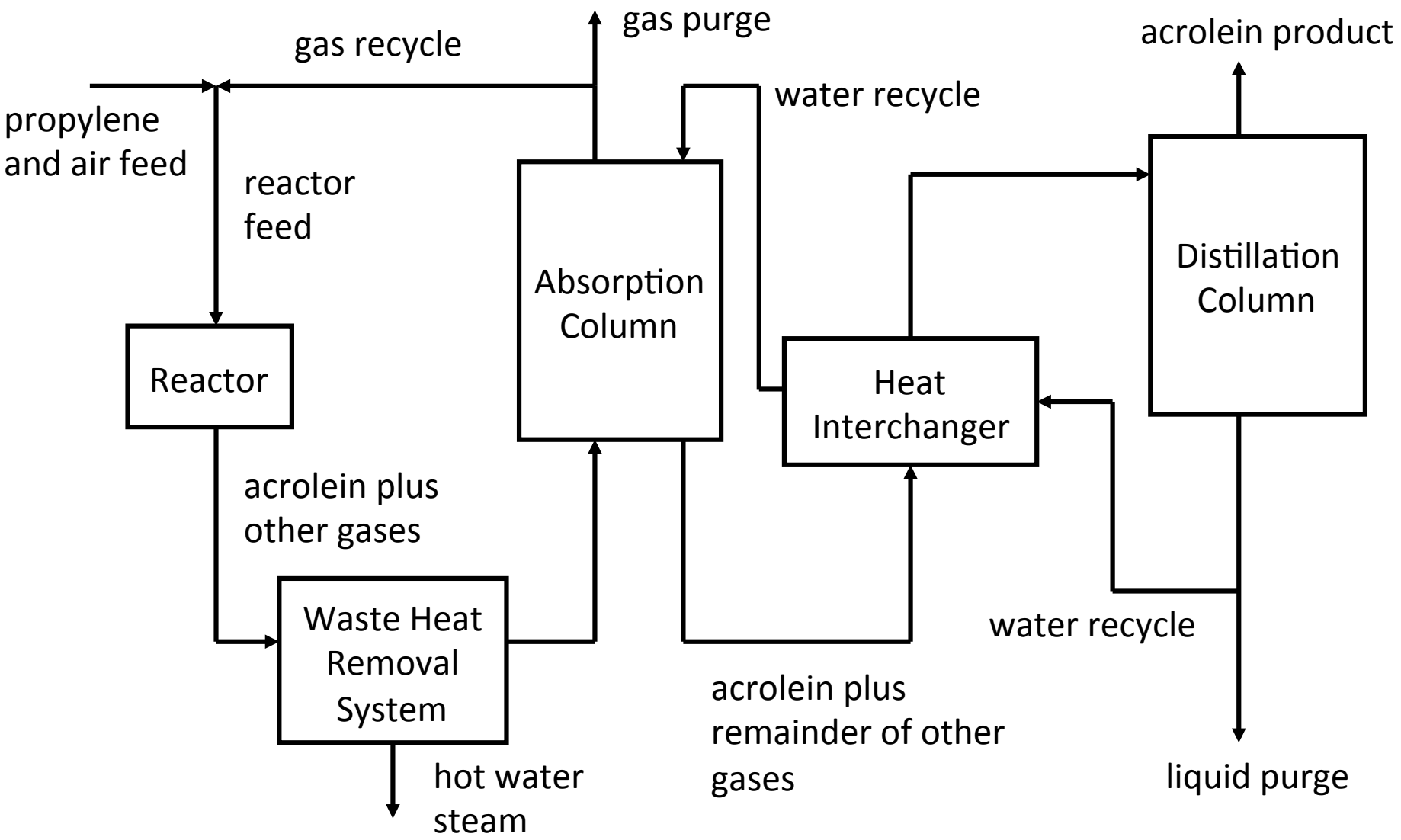
Model


Bedroom 2
13' x 11'
Family
Room
16' x 23"
Covered Lanai
Bedroom 3
10' x 12'
Kitchen
Living
Room
13' x 22"
Master
Bedroom
13' x 19'
Bedroom 4
10' x 13'
Foyer
Dining
Room
12' x 17"
Entry
Double
Garage'
Bath
Spa

Model


propylene
and air feed
Acrolein
Plant
gas purge
acrolein product
liquid purge
hot water & steam

Model


gas recycle
gas purge
acrolein product
water recycle
propylene
and air feed
reactor
feed
Absorption
Column
Distillation
Column
Reactor
Heat
Interchanger
acrolein plus
other gases
Waste Heat
Removal
System
water recycle
acrolein plus
remainder of other
gases
hot water
steam
liquid purge

Language

- Models are built in a language appropriate to the expression and analysis of properties of particular interest.

scale
projection
geometry

Architectural Plan

process

System Block Diagram

flow (material, energy, information)

Abstraction

- Abstraction is the process of removing detail from a model, of making the model more abstract.
- Reification is the opposite of abstraction, it is the process of adding detail to a model, of making the model more concrete.

Model Building

- Building a system can be seen as a process of reification. In other words moving from a very abstract statement of what is wanted to a concrete implementation.
- In doing this you move through a sequence of intermediate descriptions which become more and more concrete.
- These intermediate descriptions are models. The process of building a system can be seen as the process of building a series of progressively more detailed models.

Exercise

- Build a system block diagram model of central heating system
  - First do a high level diagram with a single block showing inputs and outputs
  - Then break down the system into sub-systems and look at the flows between them
  - Next select *one* of these sub-systems and break it down into sub-sub-systems

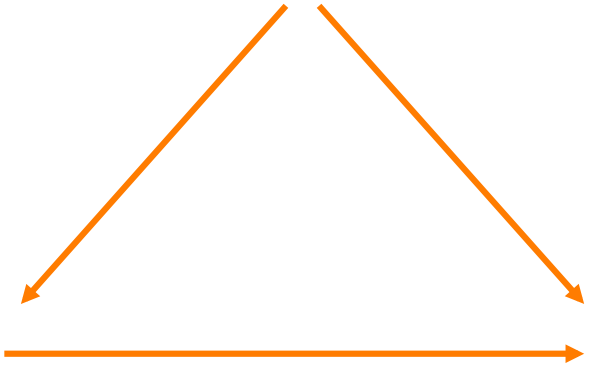
Some Questions to Ask Yourself

- Do you understand how central heating systems work? Has building the model helped?
- If given the model by somebody else would you understand what a central heating system was and how it operated?
- Have you set the “right” boundary?
- Have you used the block diagram language correctly?

Modelling

to experiment
to clarify
to understand
to analyse
to evaluate

Reason for modelling



What to model

structure
transformations
inputs and outputs
state

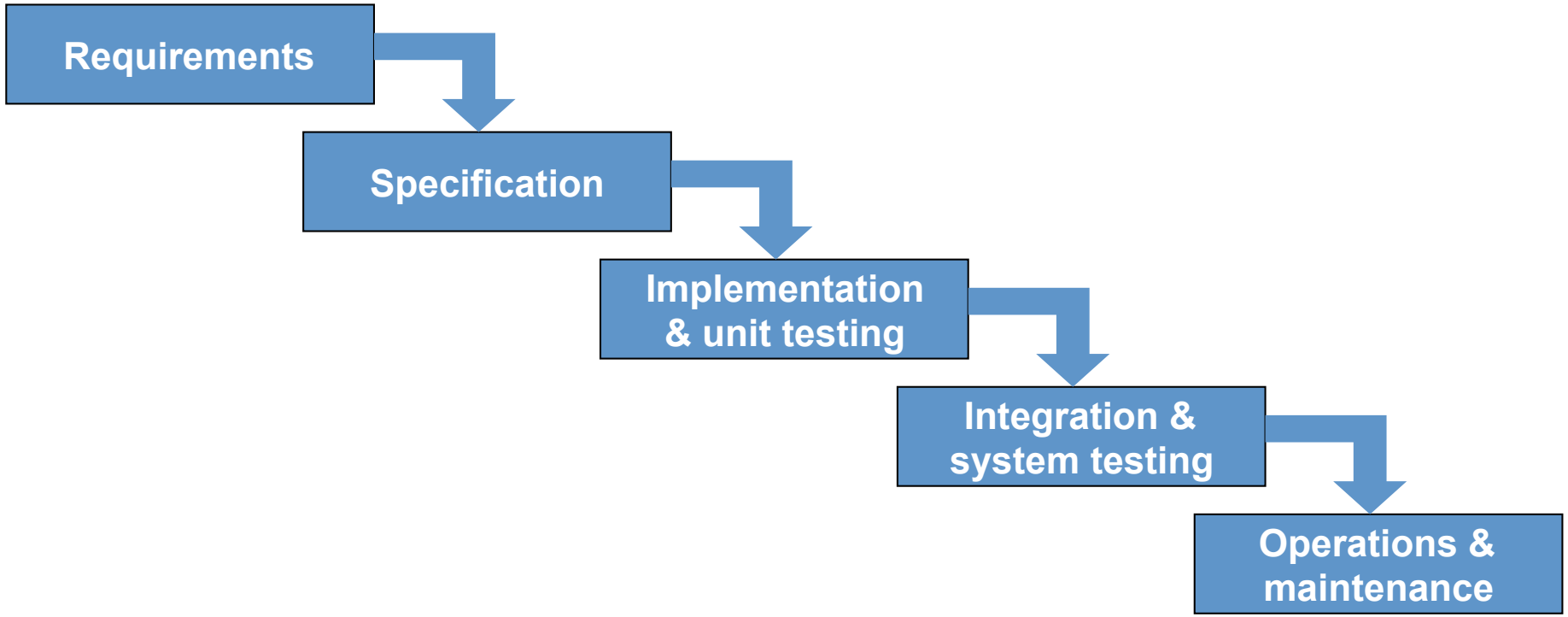
How to model

textual
graphical
mathematical

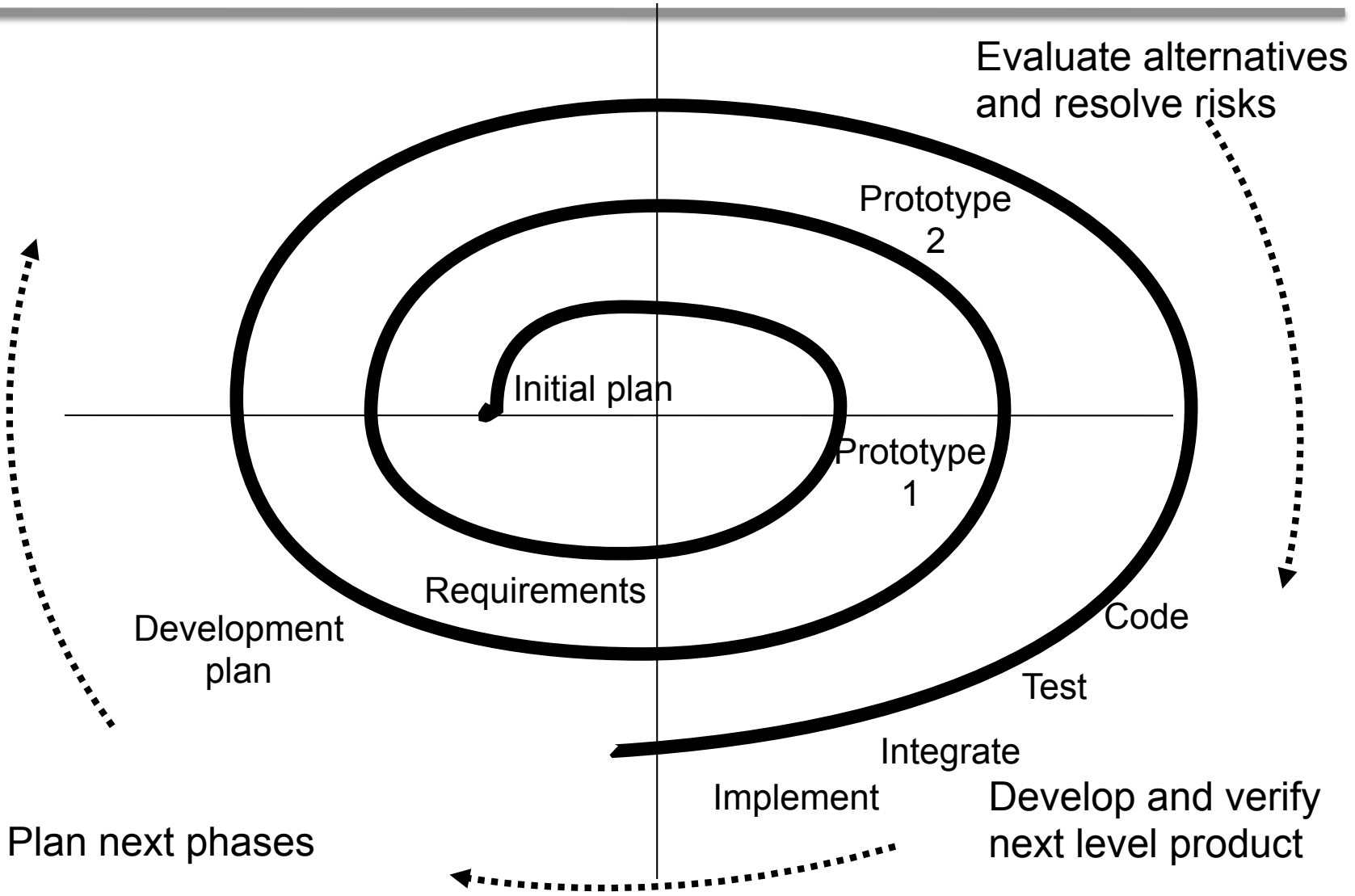
Design and process

- Design is a process, not a set of known facts
  - process of learning about a problem
  - process of describing a solution
  - at first with many gaps …
  - eventually in sufficient detail to build the solution

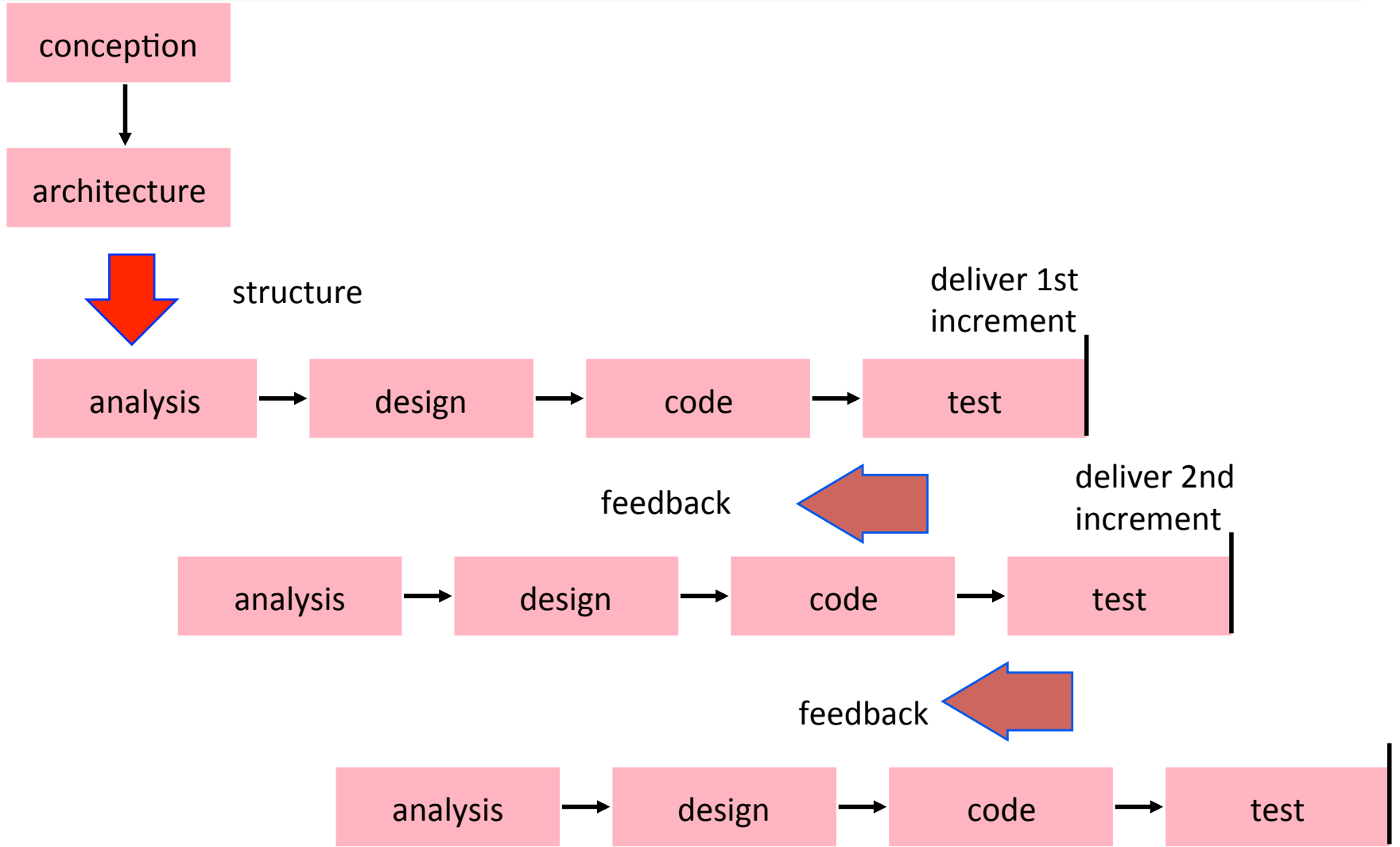
Older terminology: the “waterfall”



Modern alternative: the “spiral”



Incremental Model



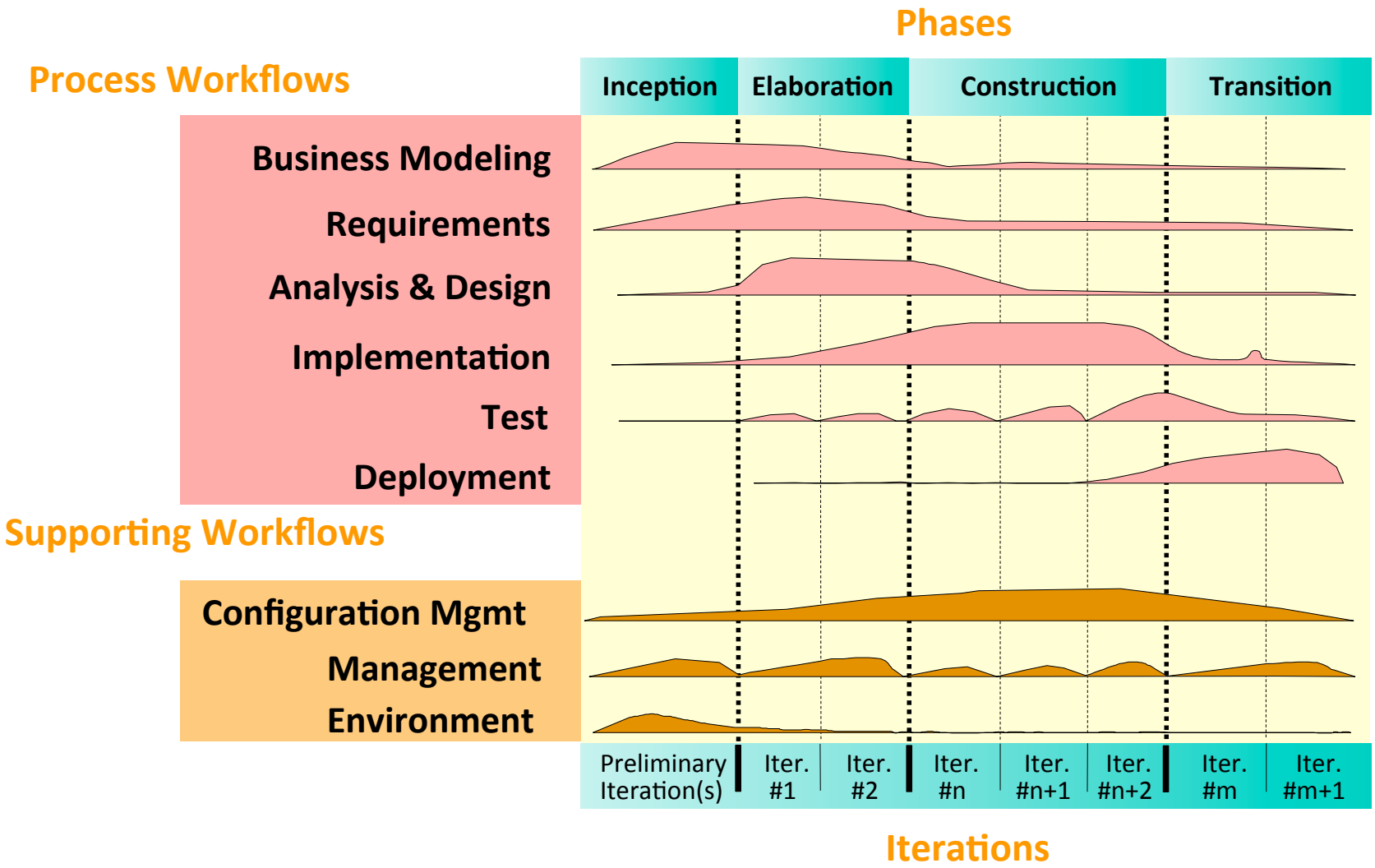
Unified Software Development Process (USDP)

- USDP is the development process associated to UML (Unified Modelling Language described later)
- USDP is based on Incremental Process
- Each iteration is like a mini-project that delivers a part of the system
  - It is use case driven
  - Architecture centric
  - Iterative and incremental

USDP basics

- Iterative & incremental
  - Iterations & baselines
  - Phases & milestones
  - Workflows
- Architecture-centric
- Use-case driven & risk confronting

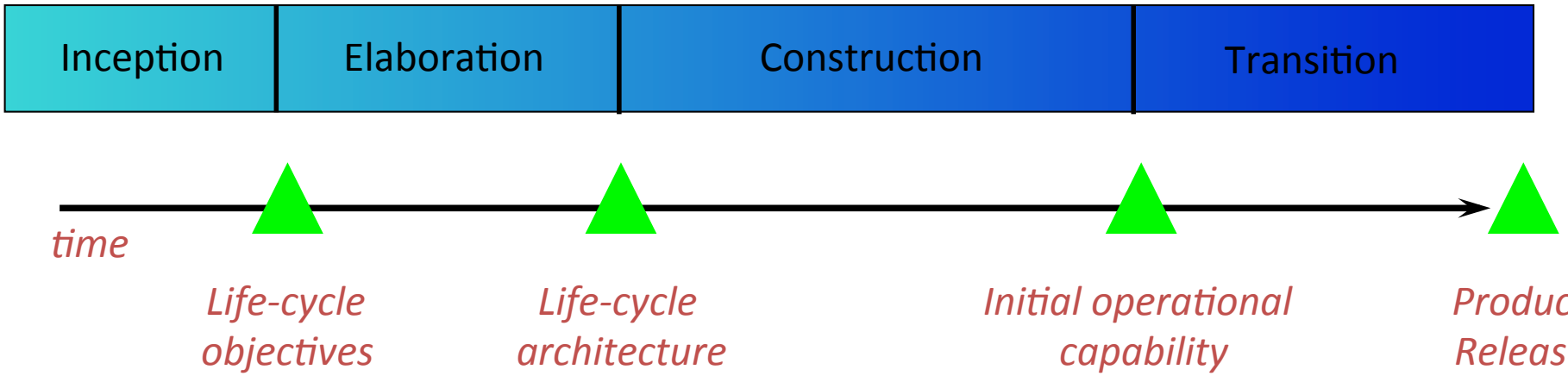
Overall structure of the USDP lifecycle



Adapted from [Jacobson 1999]

RATIONAL SOFTWARE

Lifecycle phases & milestones



- **Inception** Define scope of project & develop business case
- **Elaboration** Plan project, specify features & baseline architecture
- **Construction** Build product
- **Transition** Transition product to its users

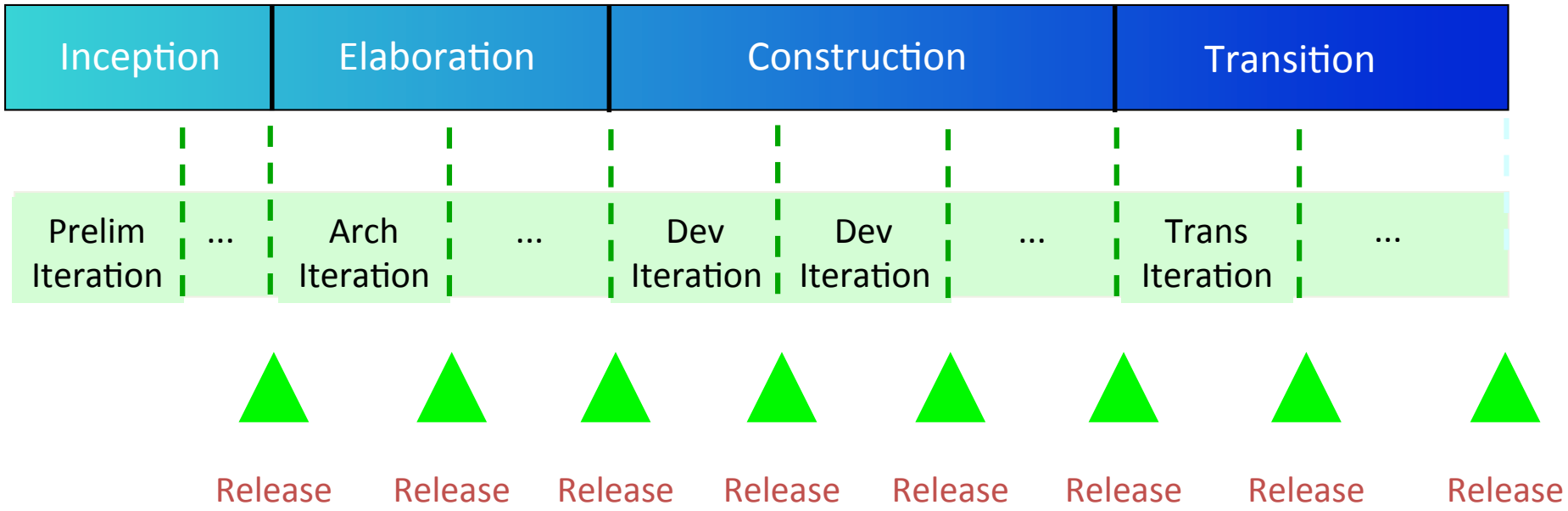
Adapted from [Booch 1999]

RATIONAL SOFTWARE

Milestone acceptance criteria

- **Lifecycle objectives** - system scope, key requirements, outline architecture, risk assessment, business case, feasibility, agreed project objectives with stakeholders
- **Lifecycle architecture** - executable architectural baseline, updated risk assessment, project plan to support bidding process, business case verified against plan, continued stakeholder agreement
- **Initial operational capability** - product ready for beta test in user environment
- **Product release** - completed beta & acceptance tests, defects fixed & in the user community

Phases & iterations



An iteration is a sequence of activities with an established plan & evaluation criteria, resulting in an executable release

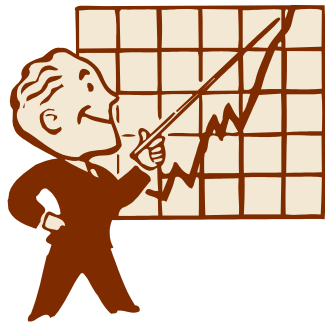
Adapted from [Booch 1999]

Iterations

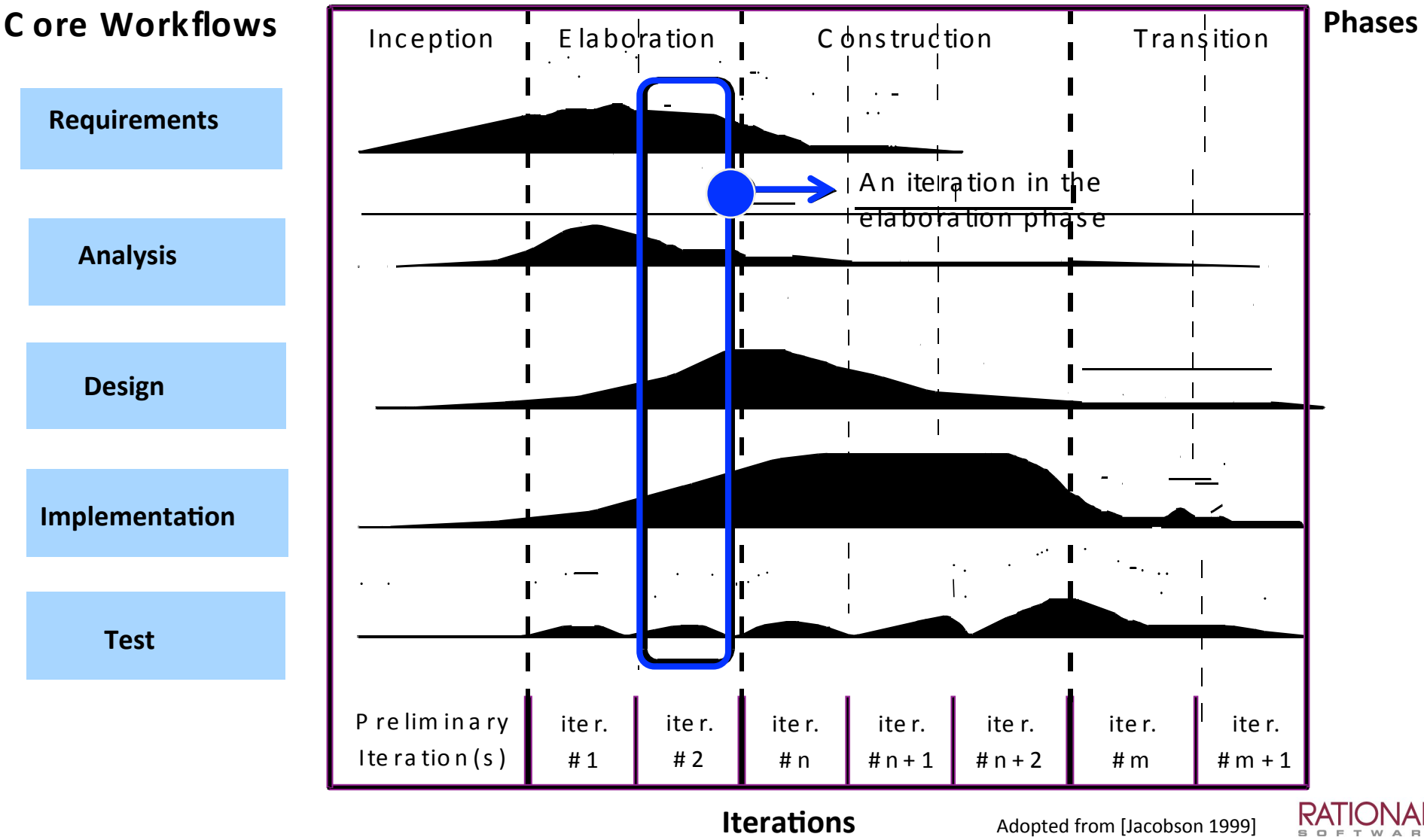
- *Iteration* is key to USDP
- Each iteration is like a mini-project
  - Planning; analysis & design; integration & test; release
  - Results in an *increment*
- 5 core workflows during each iteration
  - Requirements; analysis; design; implementation; test
- Final product release may follow a sequence of iterations (which may even overlap!)

Increments

- Each iteration results in the release of various artefacts - this is called a *baseline*
- Baselines assist with review & approvals procedures
- An *increment* is actually the difference between 2 successive baselines



Phases, iterations & workflows



Adopted from [Jacobson 1999]

RATIONAL SOFTWARE

Learning by building models

- The software design process involves gaining knowledge about a problem, and about its technical solution.
- We describe both the problem and the solution in a series of *design models*.
- Testing, manipulating and transforming those models helps us gather more knowledge.
- One of the most detailed models is written in a programming language.
  - Getting a working program is almost a side-effect of describing it!

Outline for the rest of the course

- Roughly follows stages of the (UML-related) *Rational Unified Process*
  - Inception
    - structured description of what system must do
  - Elaboration
    - defining classes, data and system structure
  - Construction
    - object interaction, behaviour and state
  - Transition
    - testing and optimisation
- Plus allowance for *iteration*
  - at every stage, and through all stages

Unified Modeling Language

- **Use Case** diagrams - interactions with / interfaces to the system.
- **Class** diagrams - type structure of the system.
- **Collaboration** diagrams - interaction between instances
- **Sequence** diagrams - temporal structure of interaction
- **Activity** diagrams - ordering of operations
- **Statechart** diagrams - behaviour of individual objects
- **Component** and **Deployment** diagrams - system organisation

Books

UML Distilled: A brief guide to the standard object modeling language
Martin Fowler, Addison-Wesley 2003 (3rd edition)

Some concepts from here:

UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design. Jim Arlow, Ila Neustadt. Addison-Wesley. 2005.

Exam questions

- This syllabus appeared *under this name* for the first time in 2006
  - See relevant questions 2006-2009
- But syllabus was previously introduced as:
  - Software Engineering II 2005, Paper 2, Q8
- Some components had previously been taught elsewhere in the Tripos:
  - Programming in Java 2004, Paper 1, Q10
  - Software Engineering and Design 2003 Paper 10, Q12 and 2004 Paper 11, Q11
  - Additional Topics 2000, Paper 7, Q13

Supervision exercises

- Use design briefs from Part 1b Group Design Projects
  - http://www.cl.cam.ac.uk/teaching/group-projects/design-briefs.html
- Choose a specific project to work on
- Carry out initial design phases, up to the point where you could start writing source code
  - Supervision 1: Inception phase + early elaboration
  - Supervision 2: Iterate and refine elaboration phase

Summary

- Systems provides a framework of concepts for thinking and talking about complex technical and social phenomena.
- Software is an important part of many large and complex real-world systems.
- Modelling requires disciplined simplification and the careful application of a modelling language.
- It is not enough to think about what you want to model you need to think about how you are going to use that model.
- Development Processes help structuring the activity of building software systems.