Software Engineering
Week 7

Lydie du Bousquet

Lydie.du-bousquet@imag.fr

In collaboration with J.-M. Favre, I. Parissis, Ph. Lalanda, Y. Ledru

Schedule

- Development processes
  - What are the activities during the development?
  - How are they organized?
  - Which one should you choose?
- Software architecture

Development process

- Also known as
  - development methodology
  - software development life cycle,
  - software process
- Structure imposed on the development of a software product
  - organization of the tasks or activities that take place during the process
  - several models for such processes

Classical activities

- Requirements Analysis
- Specification
- Software architecture
- Design
- Implementation
- Testing
- Documentation
- Training and Support
- Maintenance

Classical organizations

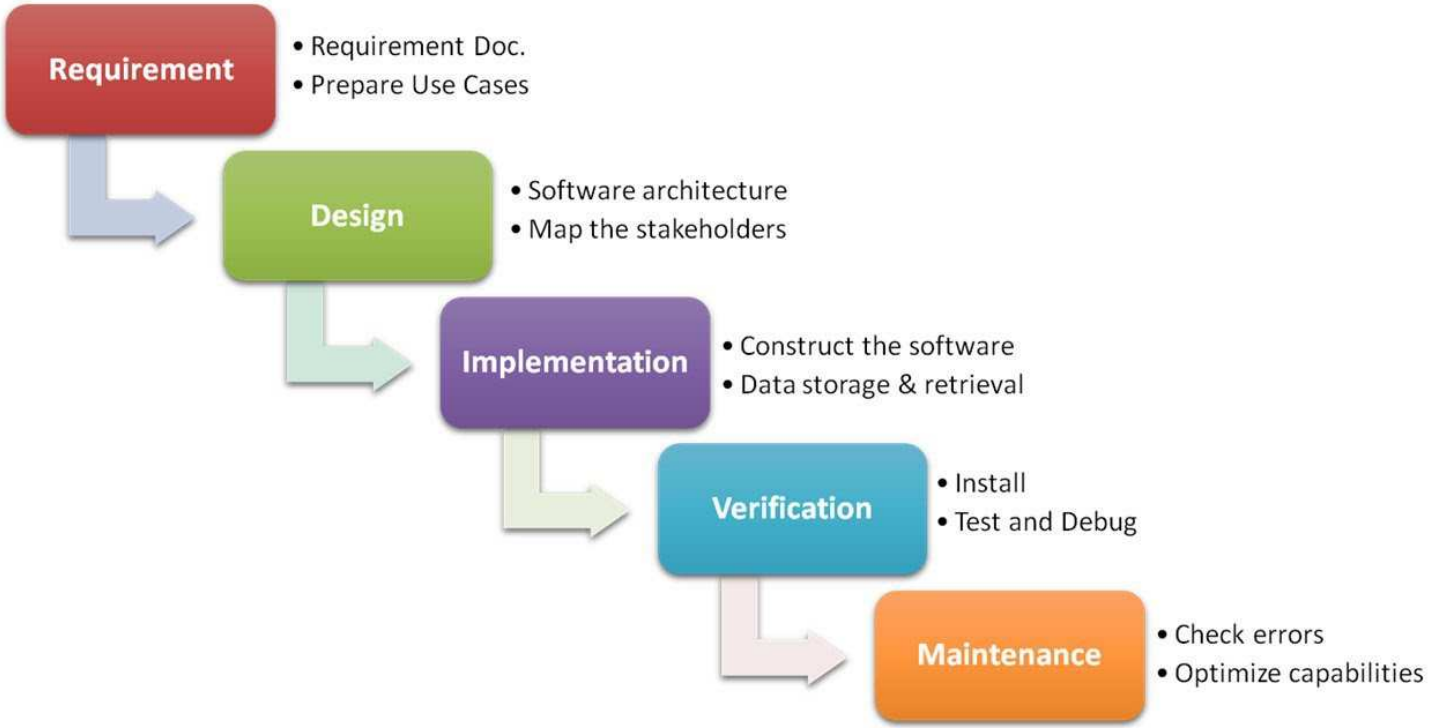
- Code and fix
- Waterfall development
- V-shaped Model
- Prototyping
- Incremental/iterative development
- Spiral
- Agile development

Code and fix

- Without much of a design in the way, programmers immediately begin producing code.
- At some point, testing begins (often late), unavoidable bugs must then be fixed before the product can be shipped.

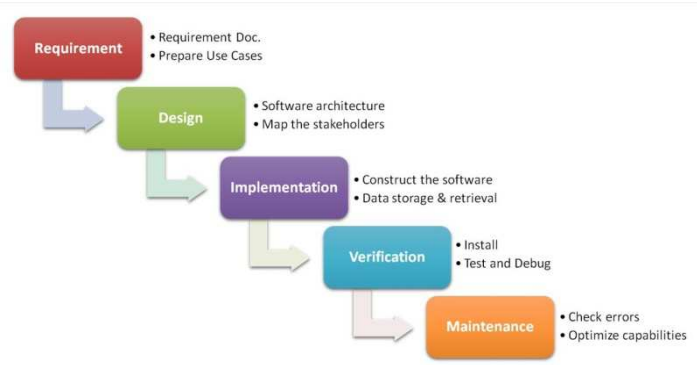
Waterfall development

- sequential development approach,
  in which development is seen as flowing steadily downwards through several phases



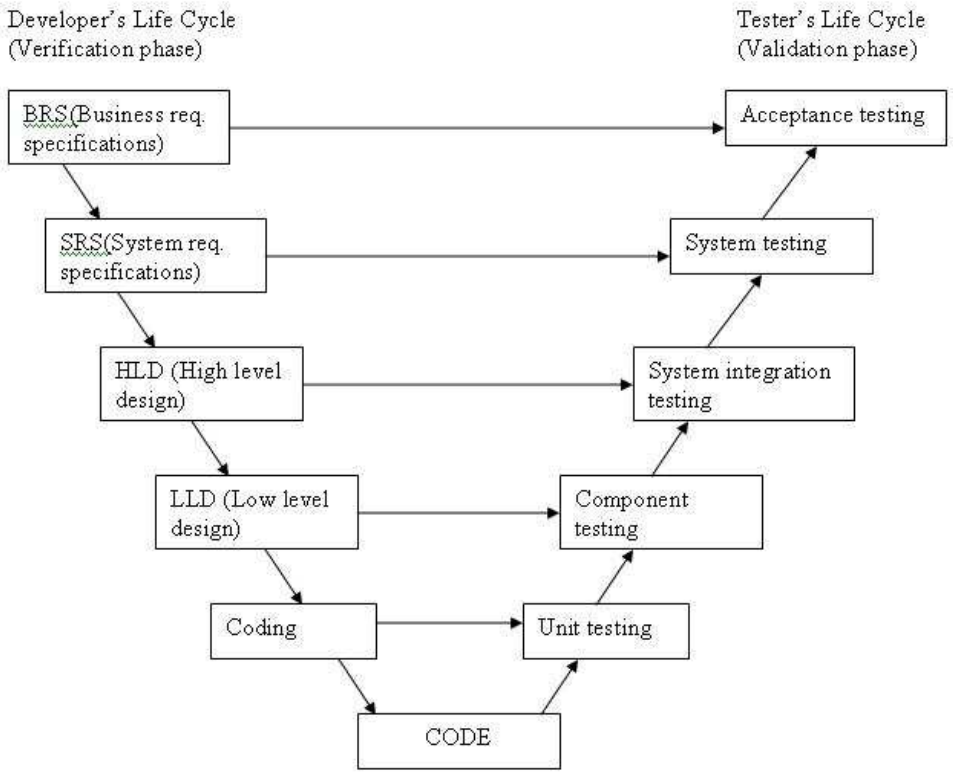
Waterfall development

- The idea behind:
  - **structured approach**: the current phase should be finished before starting a new one
  - identifiable **milestones**: each phase is documented and validated
- Criticisms
  - Clients may not know exactly what their requirements
  - Requirements may change
- Can be used
  - Small projects with well-known requirements



V-shaped Model

- Like the waterfall model, it is a **sequential** path of execution of processes
- **Testing** of the product is planned in parallel with a corresponding phase of development



V-shaped Model

- The **idea** behind
  - Focusing on validation (testing, most of the time)
- **Advantages** of V-model:
  - Simple and easy to use.
  - Testing happens well before/in parallel of coding.
- **Disadvantages** of V-model:
  - Very rigid and least flexible.
  - If any changes happen in midway, then the test documents along with requirement documents has to be updated.
- When to use the V-model:
  - Works well for small projects where requirements are easily understood
  - When validation is a key point

Prototyping

- Activity of creating **prototypes** of applications, i.e., **incomplete** versions of the product
  - simulates only a few aspects of the final product
- **Outline**
  - Identify basic requirements
  - Develop Initial Prototype
  - Review The customers, including end-users, examine the prototype and provide feedback on additions or changes.
  - Revise and Enhance the Specification / Prototype Using the feedback

Prototyping

- The **idea** behind
  - **Not** a standalone, complete development methodology
  - Attempts to reduce inherent project risk
  - User is involved throughout the development process
- **Advantages** of prototyping
  - Reduced time and costs
  - Improved and increased user involvement
- **Risks** using of prototyping
  - Insufficient analysis
  - Prototype vs finished system
  - Excessive development time of the prototype

Incremental/iterative development

- Methods combining linear and iterative methodologies,
- Objective: reduce risks by breaking a project into smaller parts and providing more ease-of-change during the development
- Different possibilities:
  - A series of mini-Waterfalls are performed. All phases of the Waterfall are completed for a small part, before proceeding to the next increment,
  - Overall requirements are defined before proceeding to evolutionary, mini-Waterfall development of individual increments of a system, or
  - Requirements analysis, and design of architecture and system core are defined via Waterfall, followed by iterative prototyping for different parts.

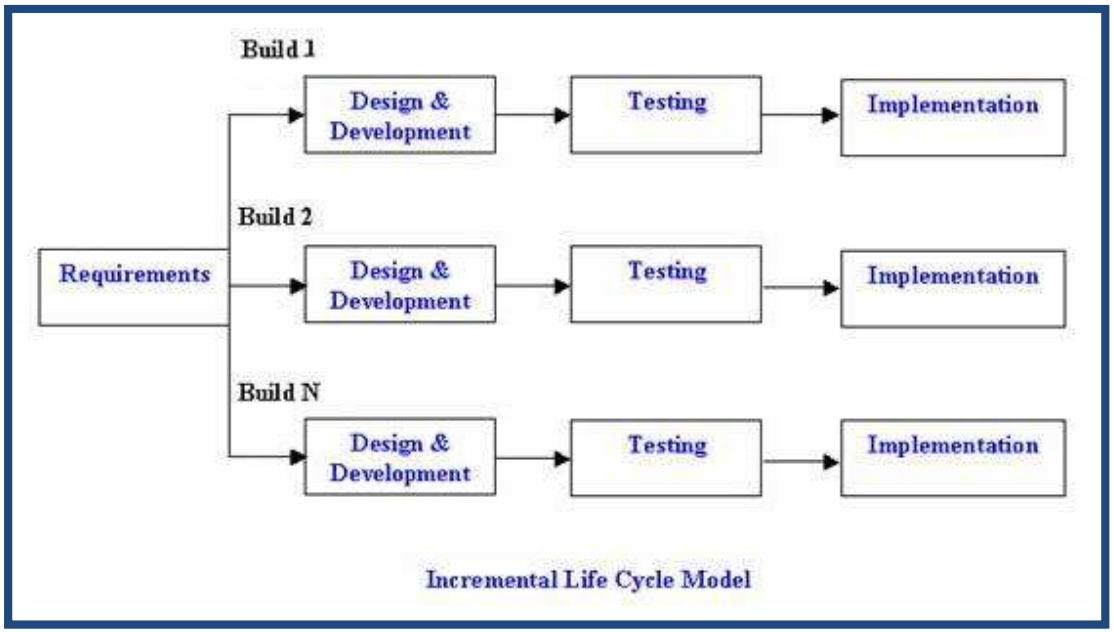
Incremental vs iterative development

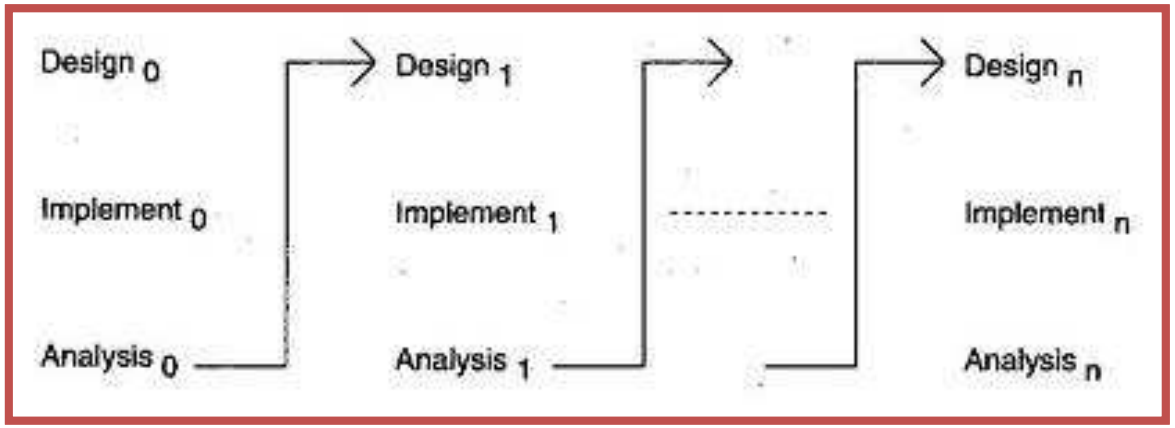
- **Iterative**: teams plan to revisit parts of the system to revise and improve them
- Disadvantages of **Iterative model**:
  - Each phase of an iteration is rigid with no overlaps
  - Costly system architecture or design issues may arise because not all requirements are gathered up front for the entire lifecycle
- When to use **Iterative model**
  - When the project is big
  - Major requirements must be defined; however, some details can evolve with time

Incremental vs **iterative** development

- **Incremental**: different parts are developed at various times and integrated based on their completion
- Disadvantages of **Incremental model**:
  - Needs good planning and design.
  - Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
  - Total cost is higher than waterfall
- When to use the **Incremental model**
  - when the requirements of the complete system are clearly defined and understood
  - major requirements must be defined; however, some details can evolve with time.

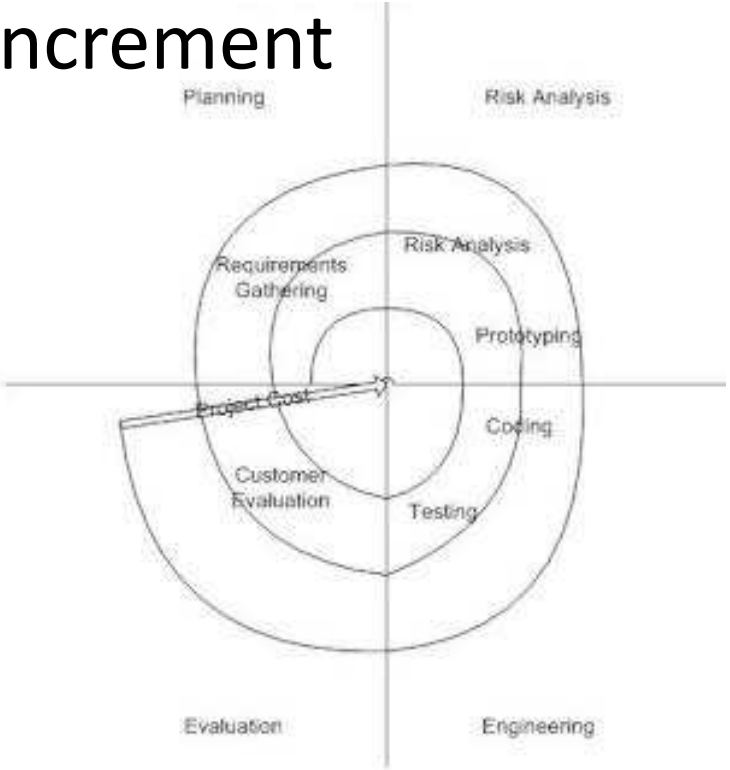
Incremental vs iterative development





Spiral development model

- Risk-driven process model
  - Similar to the incremental model,
  - with more emphasis placed on risk analysis.
- Each spiral corresponds to a increment
- Each spiral has four phases:
  - Planning,
  - Risk Analysis,
  - Engineering and
  - Evaluation



Spiral development model

- **Advantages** of Spiral model
  - High amount of risk analysis hence, avoidance of Risk is enhanced
  - Strong approval and documentation control
  - Additional Functionality can be added at a later date
- **Disadvantages** of Spiral model
  - Can be a costly model to use.
  - Risk analysis requires highly specific expertise.
  - Project's success is highly dependent on the risk analysis
  - Doesn't work well for smaller projects.

Spiral development model

- **When** to use Spiral model
  - When costs and risk evaluation is important
  - For medium to high-risk projects
  - Users are unsure of their needs
  - Requirements are complex
  - Significant changes are expected (research and exploration)

Agile development

- Incremental development processes
  - Rapid cycles (1 to 3 weeks)
  - Small release
- Example of agile methods
  - XP (Extreme Programming)
  - Scrum
  - …

Agile development: 12 principles

- **Customer satisfaction** by early and continuous delivery of useful software
- Welcome **changing requirements**, even in late development
- Working software is delivered frequently (weeks rather than months)
- Close, **daily cooperation** between business people and developers
- Projects are built around motivated individuals, who should be trusted
- **Face-to-face conversation** is the best form of communication (co-location)
- **Working software** is the principal measure of progress
- Sustainable development, able to maintain a constant pace
- Continuous attention to **technical excellence** and good design
- **Simplicity** (maximizing the amount of work not done) is essential
- Self-organizing teams
- Regular **adaptation** to changing circumstance

Agile development

- **Advantages** of Agile processes:
  - Customer satisfaction by rapid, continuous delivery of useful software
  - People and interactions are emphasized rather than process and tools
  - Customers, developers and testers constantly interact with each other
  - Working software is delivered frequently (weeks rather than months)
  - Face-to-face conversation is the best form of communication
  - Close, daily cooperation between business people and developers
  - Continuous attention to technical excellence and good design
  - Regular adaptation to changing circumstances
  - Even late changes in requirements are welcomed
- **Disadvantages** of Agile processes:
  - Difficult to assess the effort required at the beginning of the software development life cycle.
  - There is lack of emphasis on necessary designing and documentation.
  - The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- When to use Agile processes:
  - When new changes are needed to be implemented.
  - Agile assumes that the end users' needs are ever changing in a dynamic business and IT world

Choice of a development process

- Learn the about the development processes
- Assess the needs of Stakeholders
- Define the criteria (see after)

Factors	Waterfall	V-Shaped	Prototyping	Iterative and Incremental	Agile Methodologies
Unclear User Requirement	Poor	Poor	Good	Good	Excellent
Unfamiliar Technology	Poor	Poor	Excellent	Good	Poor
Complex System	Good	Good	Excellent	Good	Poor
Reliable system	Good	Good	Poor	Good	Good
Short Time Schedule	Poor	Poor	Good	Excellent	Excellent
Strong Project Management	Excellent	Excellent	Excellent	Excellent	Excellent
Cost limitation	Poor	Poor	Poor	Excellent	Excellent
Visibility of Stakeholders	Good	Good	Excellent	Good	Excellent
Skills limitation	Good	Good	Poor	Good	Poor
Documentations	Excellent	Excellent	Good	Excellent	Poor
Component reusability	Excellent	Excellent	Poor	Excellent	Poor

Exercise
Which development process?

- System for student management in university (this system replace an existing system without any functional evolutions)
- An new interactive system for travelers to have schedules on their smartphones
- A system to control subway without drivers
- A very large system for Air traffic management
- A very new 3D-system for software maintenance

And now ?

- You have collected the requirements
- You have chosen the development process
- What should be done now ?

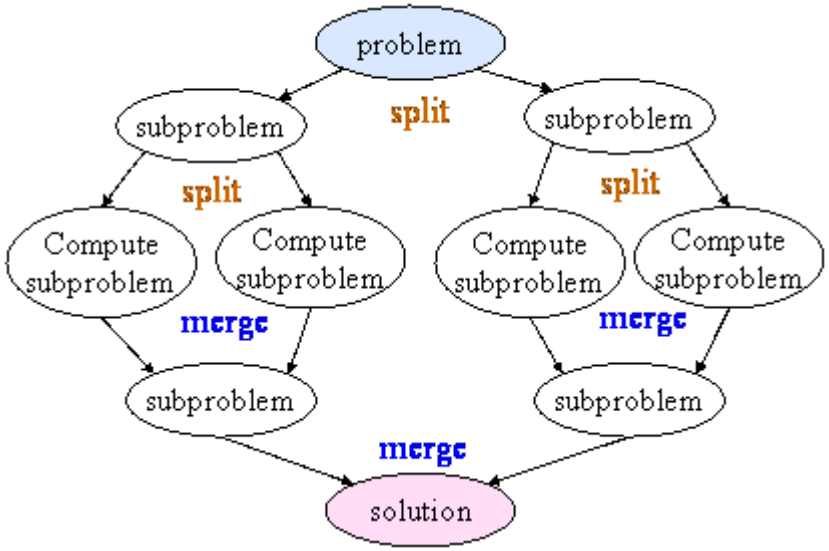
And now ?

- You have collected the requirements
- You have chosen the development process
- What should be done now ?

Start the design
(w.r.t. the development process)

Problem decomposition

- The only way to solve a complex problem is to **decompose** it into smaller ones
  - « divide to conquer »



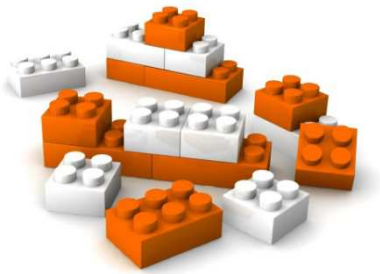
- Different forms of decomposition
  - Modularity and abstraction
  - Separation of concerns

Schedule

- Development processes
- Software architecture
  - What is it?
  - What is it for?

Software architecture

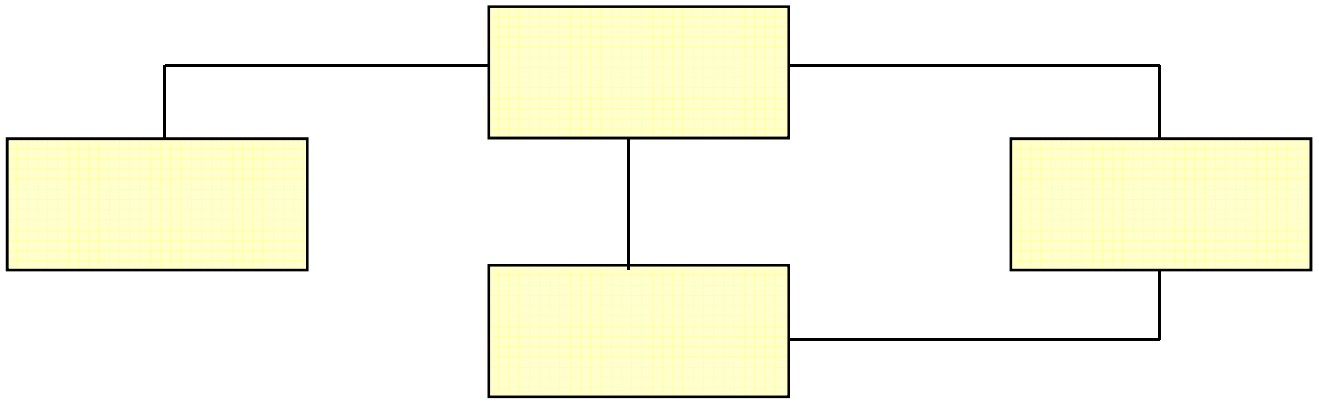
- Refers to the high level structures of a software system
- Discipline of
  - creating such structures, and
  - documenting of these structures



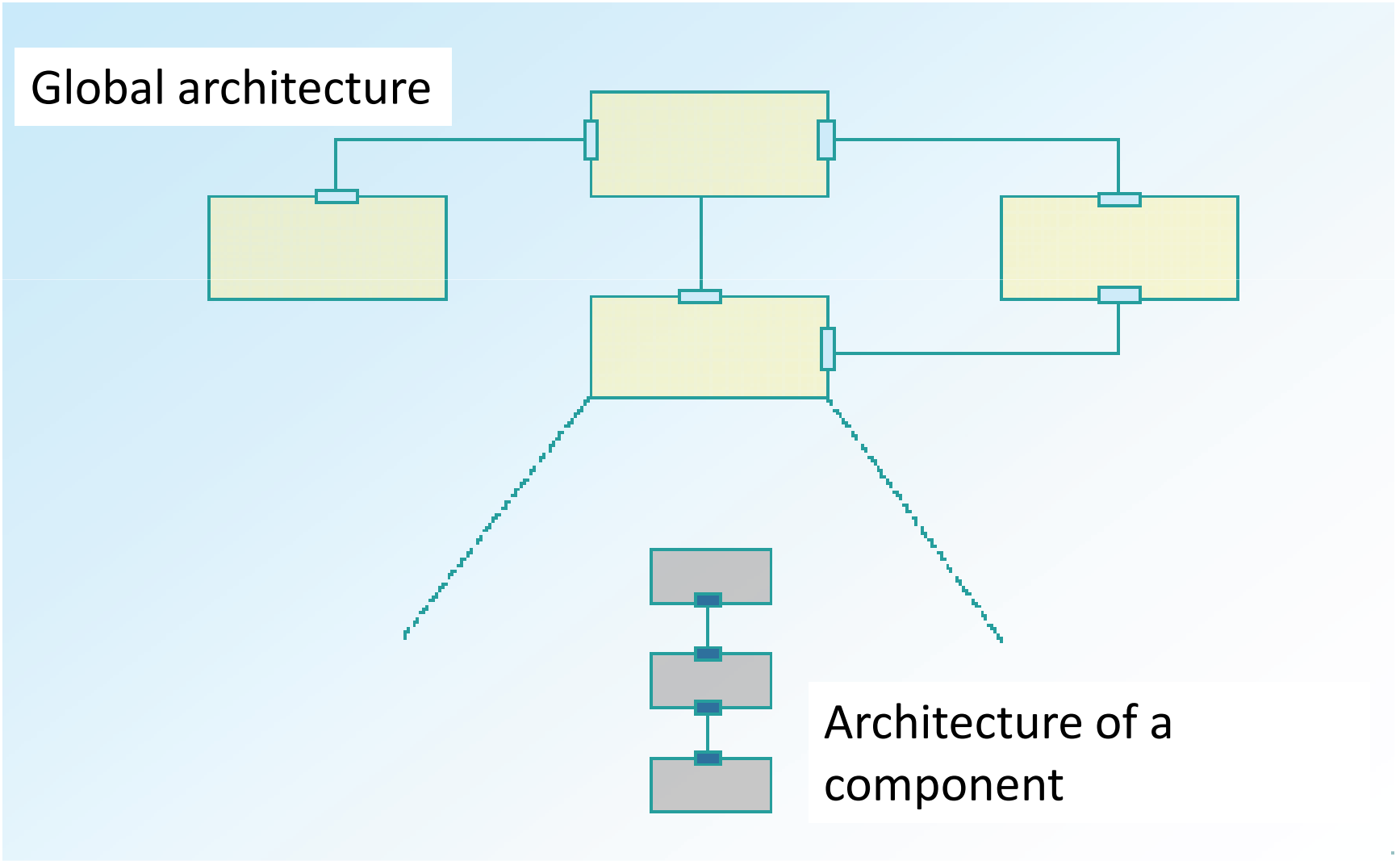
- Advantages
  - facilitates communication between stakeholders,
  - captures early decisions about the high-level design,
  - allows reuse of design components between projects

Software architecture definition

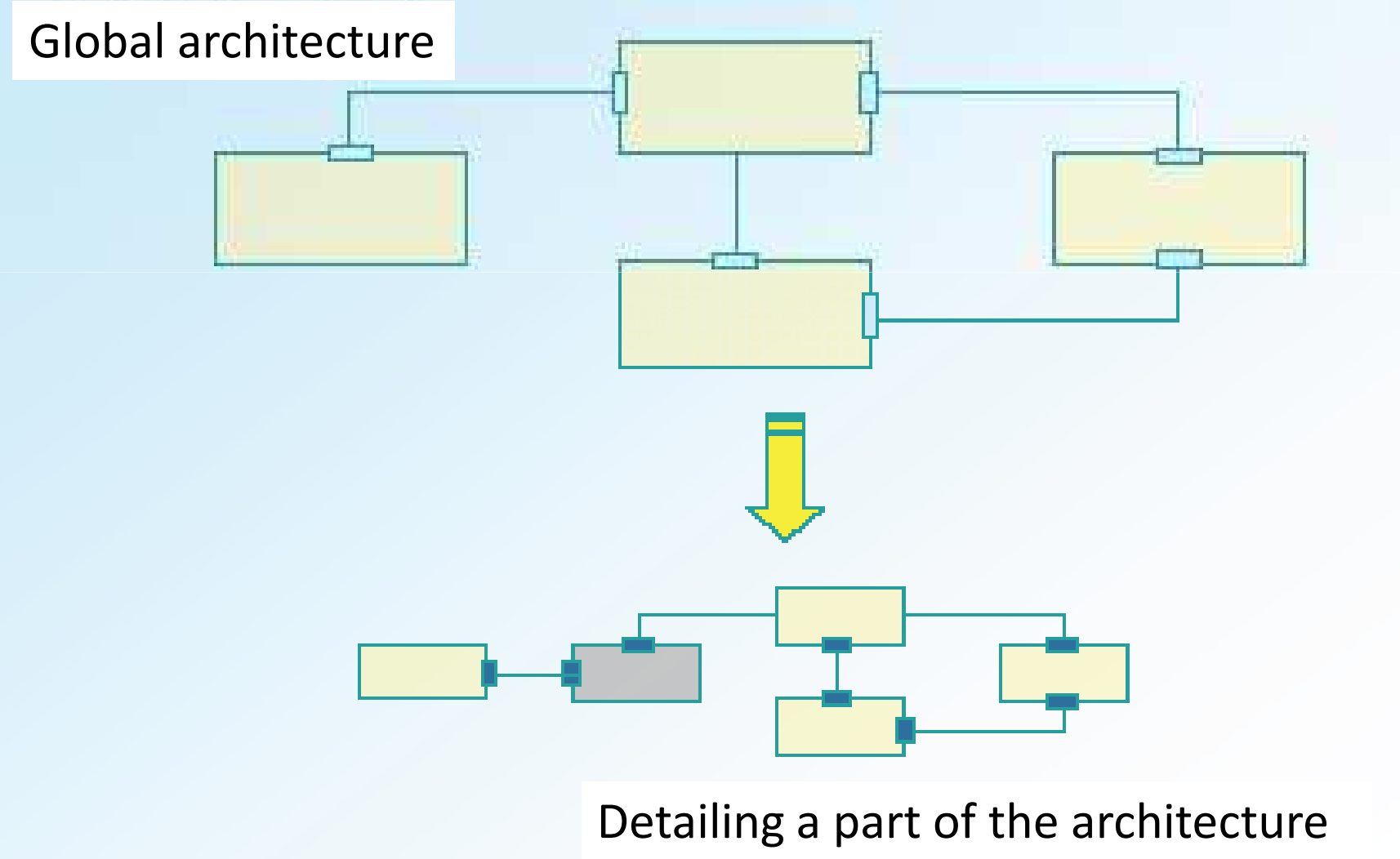
- Abstract representation of the final system as components and connectors
  - First decomposition of the problem/solution



A sequence of abstractions

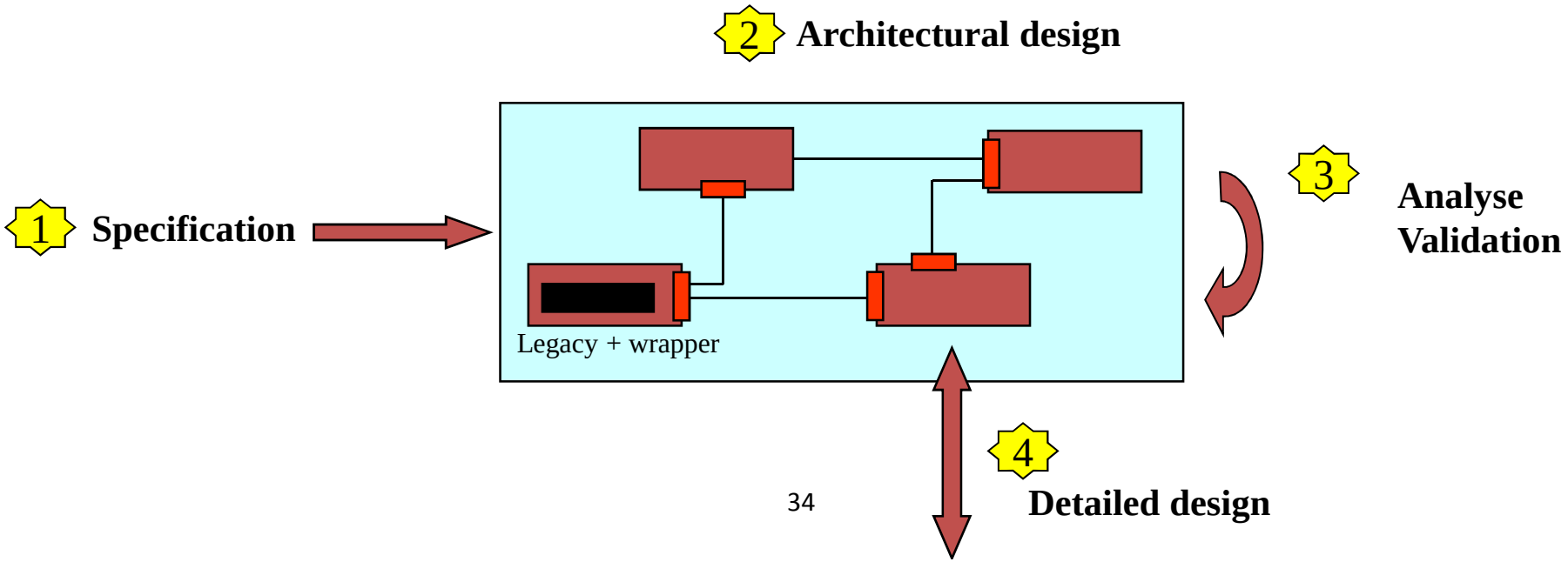


A sequence of abstractions



Position

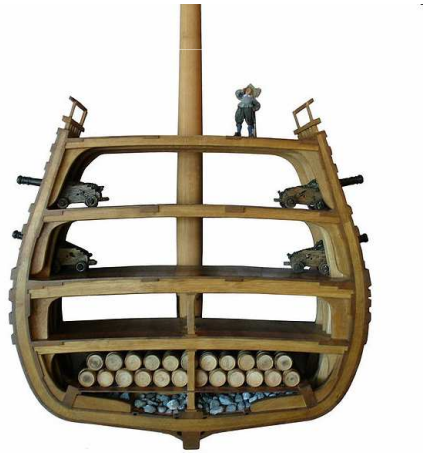
- Architecture = first step(s) of conception



Why architecture step ?

Vasa story

- Swedish warship
  - Built between 1626 and 1628
- Requirements
  - 2 gun decks
  - large and a small ship
- Sank after sailing about 1,300 m
  - into her maiden voyage
- Why?
  - inexperience
  - Requirement were not managed



Images from Wikipedia

Why architecture step ?

Vasa story

- Requirement were not incompatible
  - A second ship was built
    - 1,5 meter more in width
    - Did not sink
- The architect did not his job
  - Requirement given by the king
  - Inexperience

Why architecture step ?

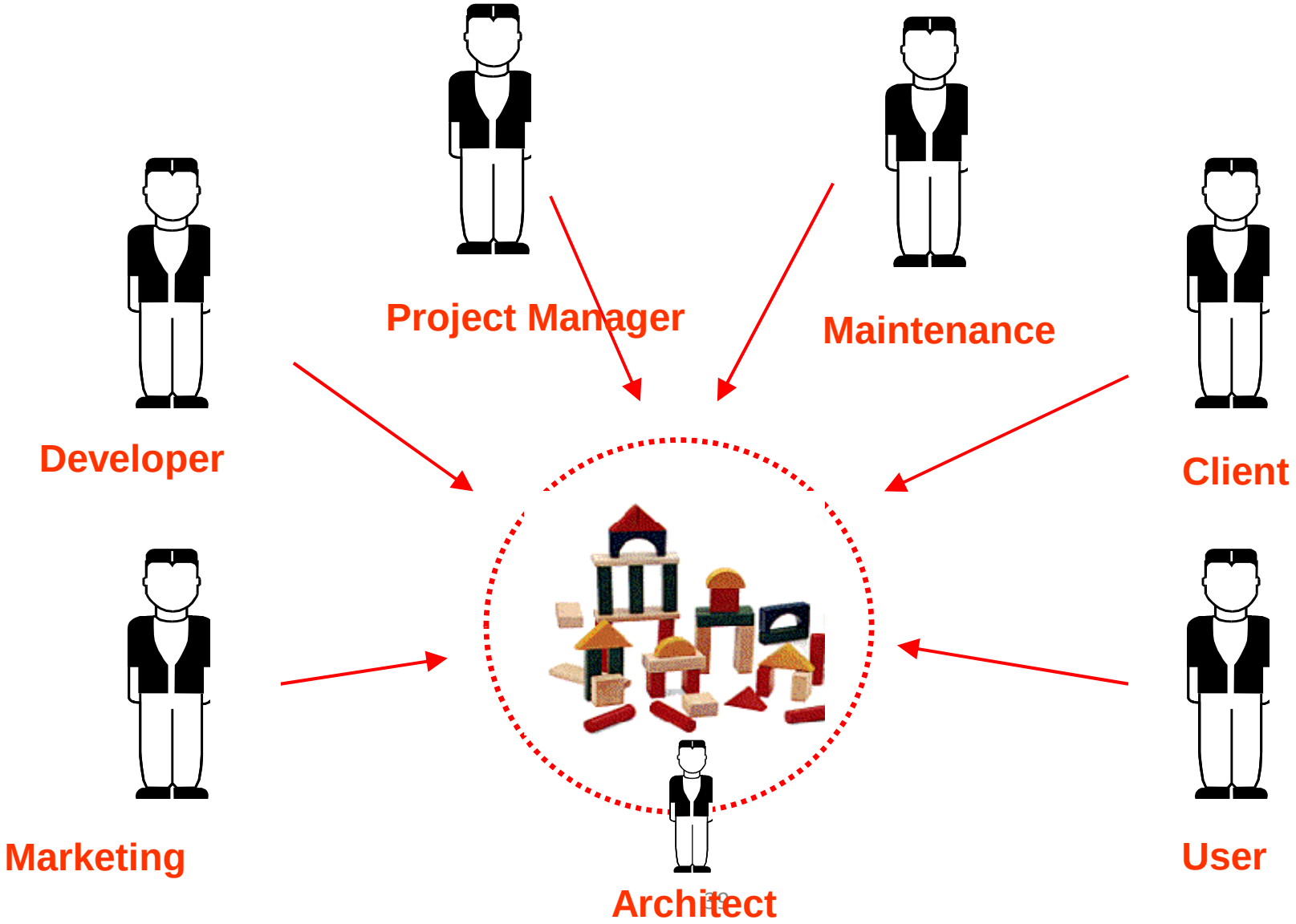
Vasa story: lessons

- Clients always enforce requirements
- Some of them are not compatible
- Architecture step in the development process aims at indentifying consequences of choices
- If there are incompatibilities, tradeoff have to be made, by the architect
- In case of problem, it is the responsibility of the architect, not of the king!

Architecture and Software qualities

- Architecture has an **influence** on the final properties of a system
- The architectural structure **favors or penalize** the non functional properties such as:
  - Performance
  - Security
  - Safety
  - Availability
  - Maintainability
  - etc.
- It is a first level of tradeoff.
  And it will be impossible to come back

Architecture: a meeting point



Architecture

- How to represent an architecture?
- How to design an architecture?

  – Exercise:

  Let XXX an old-system to be modified and extended

  No documents are available but code

  No one can remember how the system work

  What do you need to be able to do the job?

Exercise

- The given code is for a commercial web-site to sell drugs (medicine)
- Know, you want to
  - Sell toys
  - Add security
- You have the code

What do you need to be able to do the job?

Which views to represent the architecture?

- Structure of the application
  - Components and the way they are linked
  - **Logical View(s)**
- Behavior
  - Interaction among the components
  - **Dynamical View(s)**
- Hardware
  - Which hardware is required/used
  - **Physical view(s)**

Which views to represent the architecture?

But also

- Some elements to identify the context
  - Stakeholders, limits of the system
  - **Context diagram**
- **Functionalities** of the system
  - Defined in the specification
  - Use case diagram

References

- Software architecture in practice - second edition
  Len Bass, Paul Clements, Rick Kazman
  Addison Wesley, 2003

- Pattern-oriented software architecture
  Buschmann, Meunier, Rohnert, Sommerlad, Stal
  Wiley, 1996

- Applied software architecture
  Hofmeister, Nord, Soni
  Addison Wesley, 2000

- Design and use of software architectures
  Jan Bosch
  Addison Wesley, 2000

For the final evaluation

- You should know
  - Advantages and limits of the different life-cycle
  - Challenges and issues of software architecture
- You should be able to
  - Choose an appropriate life-cycle
  - Reformulate a simple textual description with the appropriate UML diagram