Project

Barbara Re, Phd

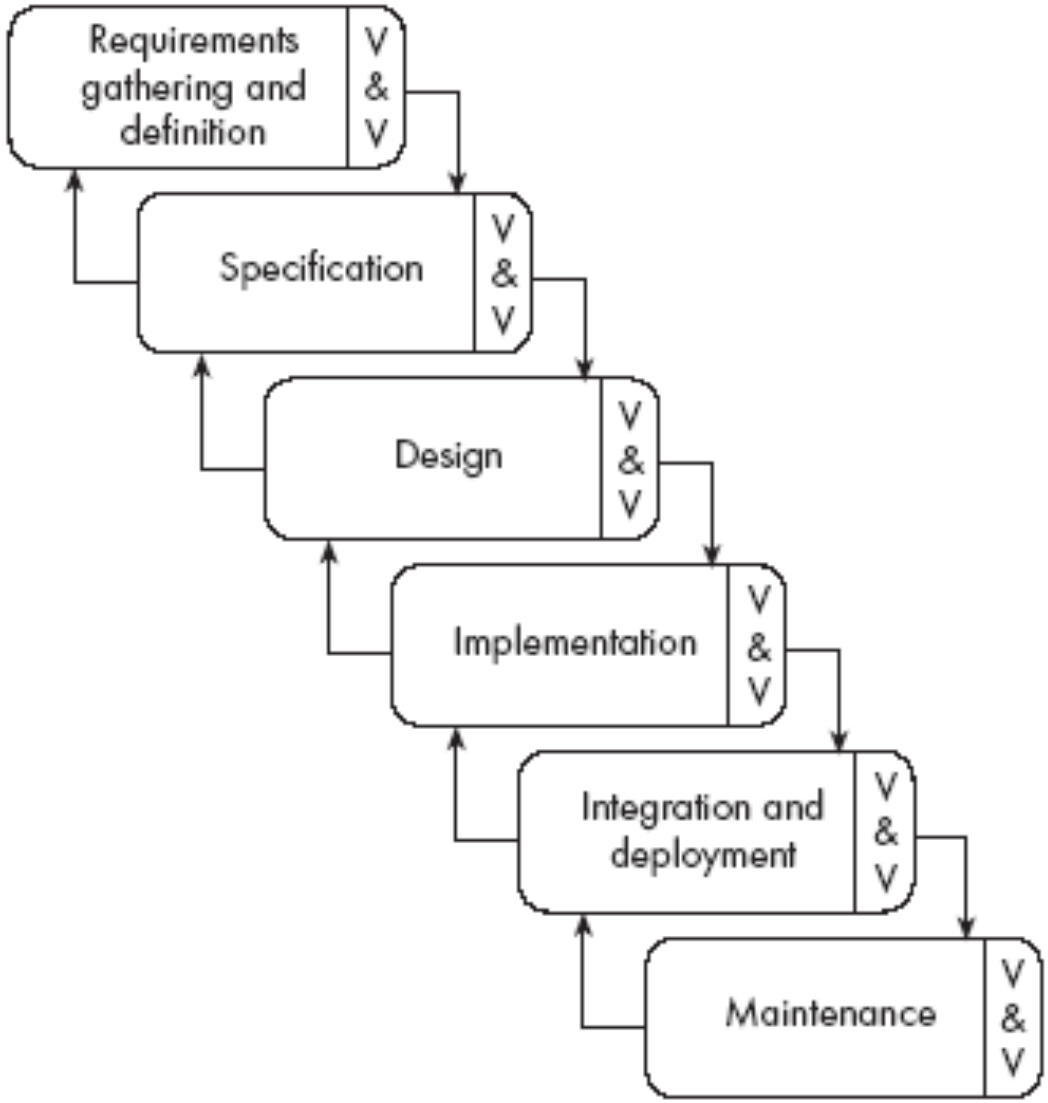
Steps …

- Motivation
- Objective
- Requirements Analysis
- Model and Abstraction
- Implementation
- Test and Validation

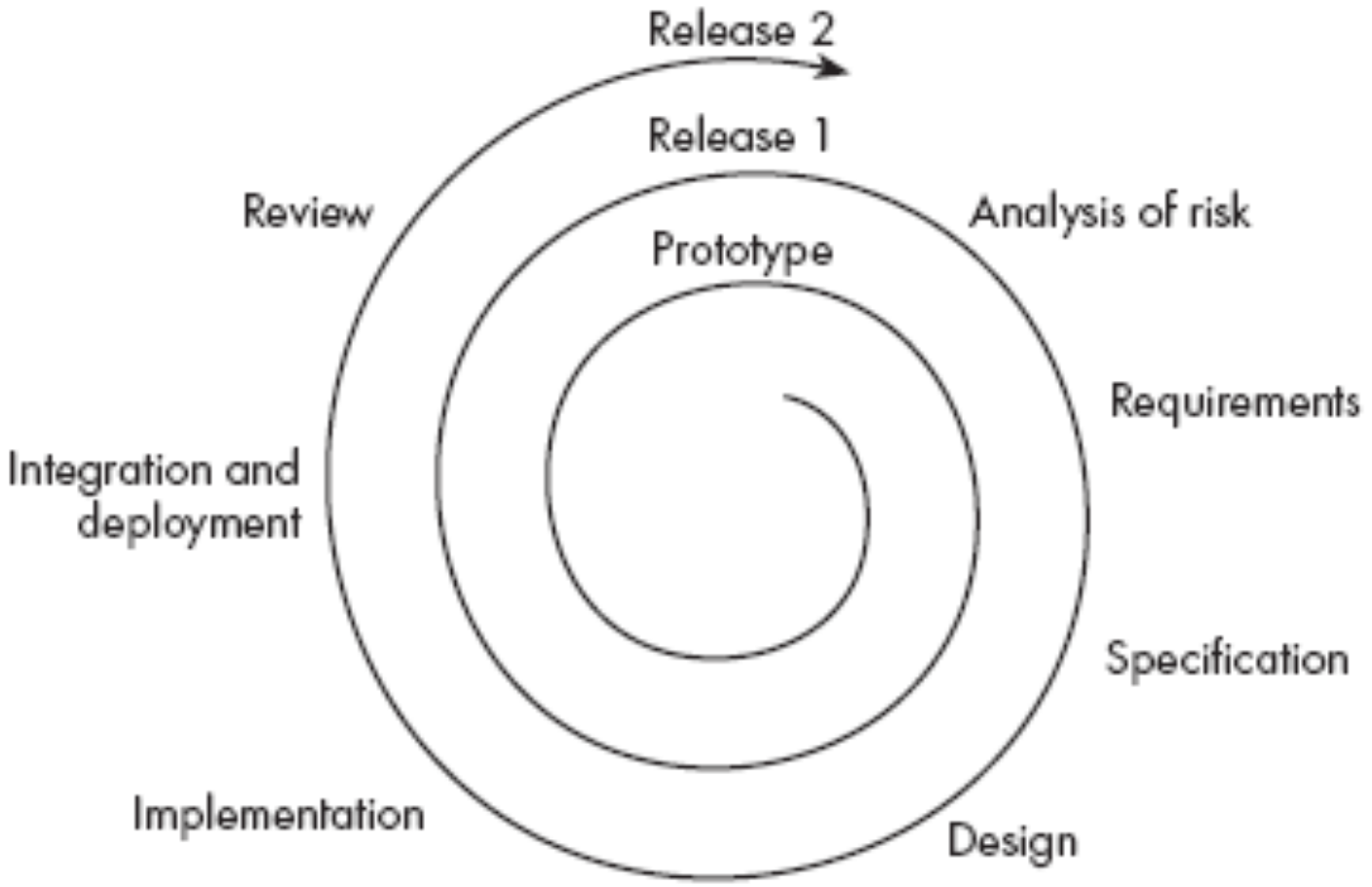
3P

- Effective SW project management focuses on 3 P's:
  - **People**
    - must be organized into effective teams
    - motivated to do high-quality work
    - coordinated to achieve effective communication and results
  - **Problem**
    - must be communicated from customer to developer
    - decomposed into its parts
    - positioned for work by SW team
  - **Process**
    - must be adapted to the people and problem

The waterfall model



The spiral model



Choosing and Appling

- From the waterfall model:
  - Incorporate the notion of stages.
- From the phased-release model:
  - Incorporate the notion of doing some initial high-level analysis, and then dividing the project into releases.
- From the spiral model:
  - Incorporate prototyping and risk analysis.
- From the evolutionary model:
  - Incorporate the notion of varying amounts of time and work, with overlapping releases.
- From concurrent engineering:
  - Incorporate the notion of breaking the system down into components and developing them in parallel.

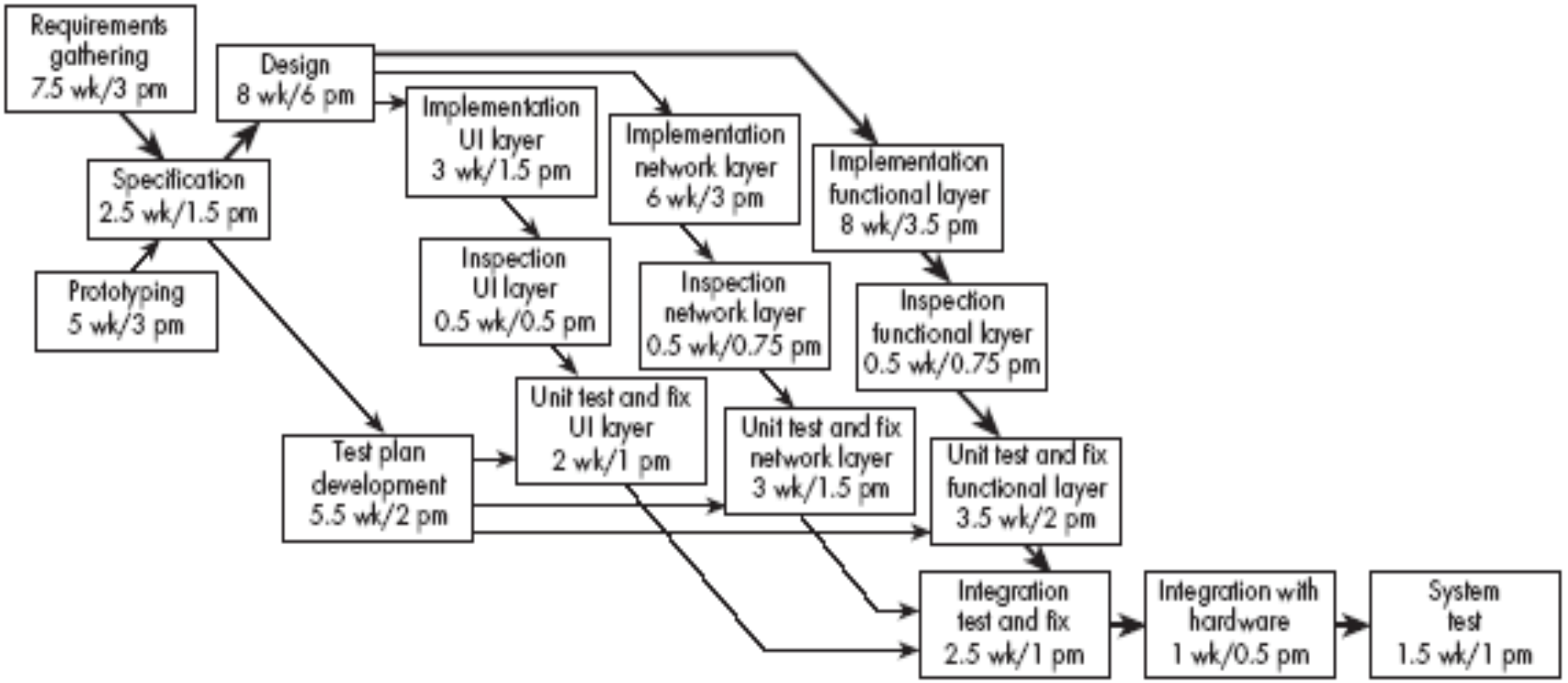
Project Scheduling and Tracking

- *Scheduling* is the process of deciding:
  - In what sequence a set of activities will be performed.
  - When they should start and be completed.
- *Tracking* is the process of determining how well you are sticking to the cost estimate and schedule.

PERT charts

- A PERT chart shows the sequence in which tasks must be completed.
  - In each node of a PERT chart, you typically show the elapsed time and effort estimates.
  - The *critical path* indicates the minimum time in which it is possible to complete the project.

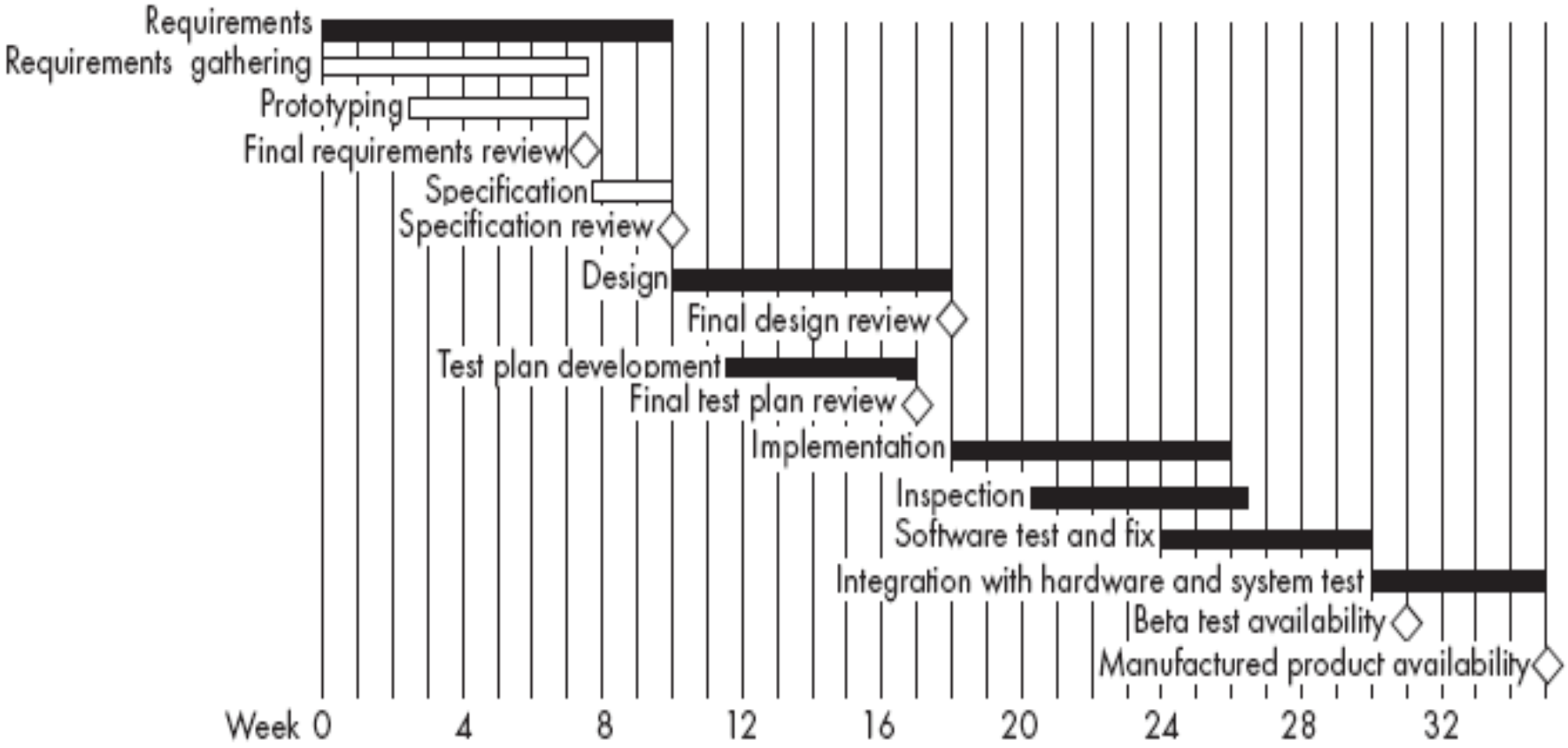
Example of a PERT chart



Gantt charts

- A Gantt chart is used to graphically present the start and end dates of each software engineering task
  - One axis shows time.
  - The other axis shows the activities that will be performed.
  - The black bars are the top-level tasks.
  - The white bars are subtasks
  - The diamonds are *milestones:*
    - Important deadline dates, at which specific events may occur

Example of a Gantt chart



Focus Group Vs. Individual Interview

- **A focus group** is a small group discussion guided by a trained leader, used to learn more about opinions on a designated topic, and then guide future action.

- Individual Interview

How to Conduct a Focus Group:

Before the meeting:

- Decide on the meeting particulars.
- Prepare your questions.
- Recruit your members.
- Review the arrangements.

How to Conduct a Focus Group:

When the group meets:

- Thank people for coming.
- Review the group's purpose and goals.
- Explain how the meeting will proceed and how members can contribute.
- Set the tone by asking an opening question and making sure all opinions on that question are heard.

How to Conduct a Focus Group:

When the group meets:

- Ask further questions in the same general manner.
- When all your questions have been asked, ask if anyone has any other comments to make.
- Tell the group about any next steps that will occur and what they can expect to happen now.
- Thank the group for coming!

How to Conduct a Focus Group:

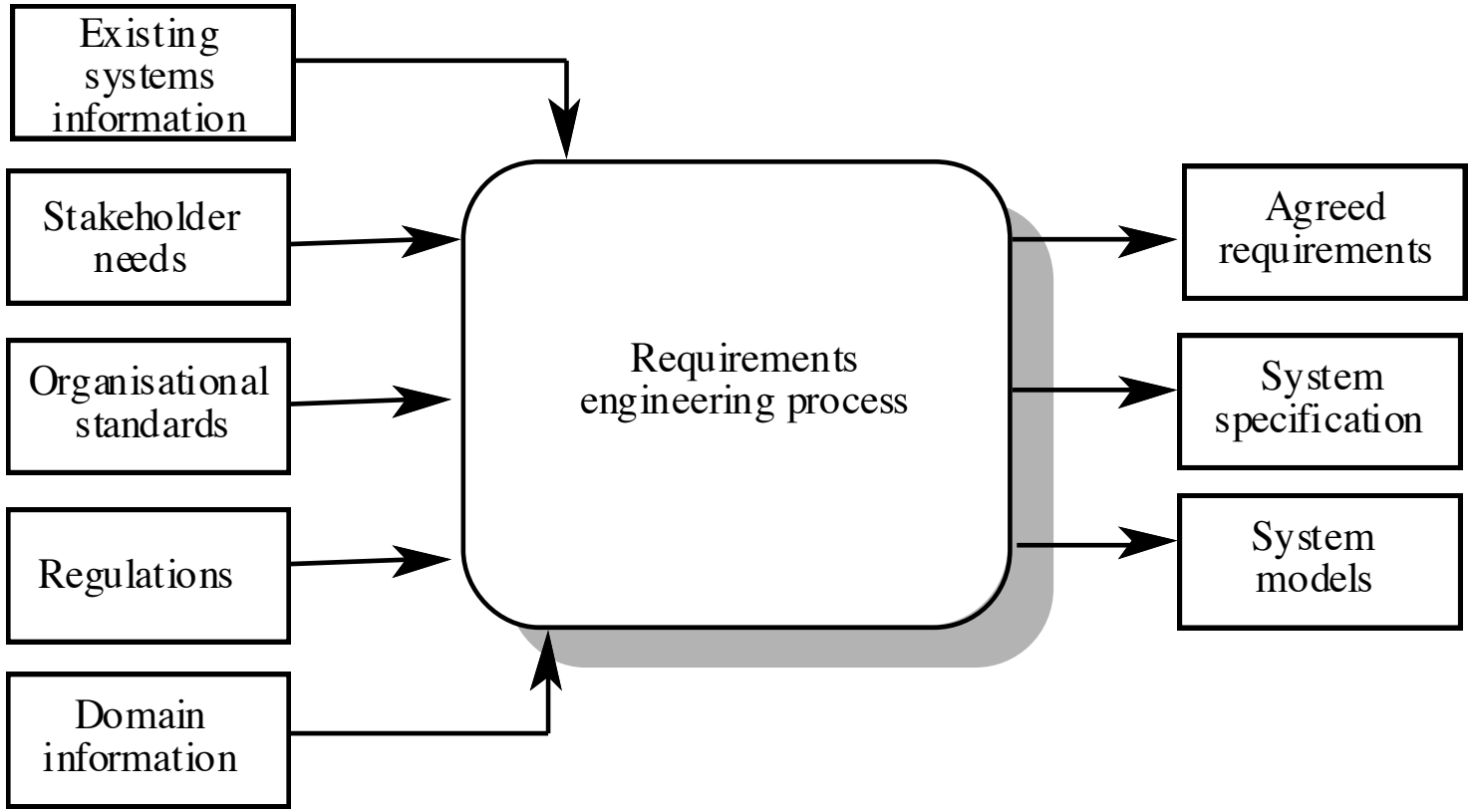
After the meeting meets:

- Make a transcript or written summary of the meeting.
- Examine the data for patterns, themes, new questions, and conclusions.
- Share the results with the group.
- Use the results.

Requirements Analysis [1]

- What is it?
  - The process by which customer needs are understood and documented.
  - Expresses "what" is to be built and NOT "how" it is to be built.
- Example 1:
  - The system shall allow users to withdraw cash. [*What?*]
- Example 2:
  - A sale item's name and other attributes will be stored in a hash table and updated each time any attribute changes. [*How?*]

Requirements Engineering Processes



Requirements Analysis [2]

- C- and D-Requirements
  - C-: Customer wants and needs; expressed in language understood by the customer.
  - D-: For the developers; may be more formal.

Requirements Analysis [3]

- Roadmap:
  - Identify the customer.
  - Interview customer representatives.
  - Write C-requirements, review with customer, and update when necessary.
  - Write D-requirements; check to make sure that there is no inconsistency between the C- and the D-requirements.

Requirements Analysis [4]

- C-requirements:
  - *Use cases* expressed individually and with a use case diagram. A use case specifies a collection of scenarios.
    - Sample use case: *Process sale*.
  - *Data flow diagram*:
    - Explains the flow of data items across various functions. Useful for explaining system functions. [Example on the next slide.]
  - *State transition diagram*:
    - Explains the change of system state in response to one or more operations. [Example two slides later.]
  - *User interface*: Generally not a part of requirements analysis though may be included. [Read section 3.5 from Braude.]

Requirements Analysis [5]

1. Organize the D-requirements.
   (a) Functional requirements

   The blood pressure monitor will measure the blood pressure and display it on the in-built screen

   (b) Non-functional requirements

   (i) Performance

   The blood pressure monitor will complete a reading within 10 seconds.

   (i) Reliability

   The blood pressure monitor must have a failure probability of less than 0.01 during the first 500 readings.

Requirements Analysis [6]

(c) Interface requirements: interaction with the users and other applications

The blood pressure monitor will have a display screen and push buttons. The display screen will….

(d) Constraints: timing, accuracy

The blood pressure monitor will take readings with an error less than 2%.

Model and Abstraction from semiformal to formal

Model

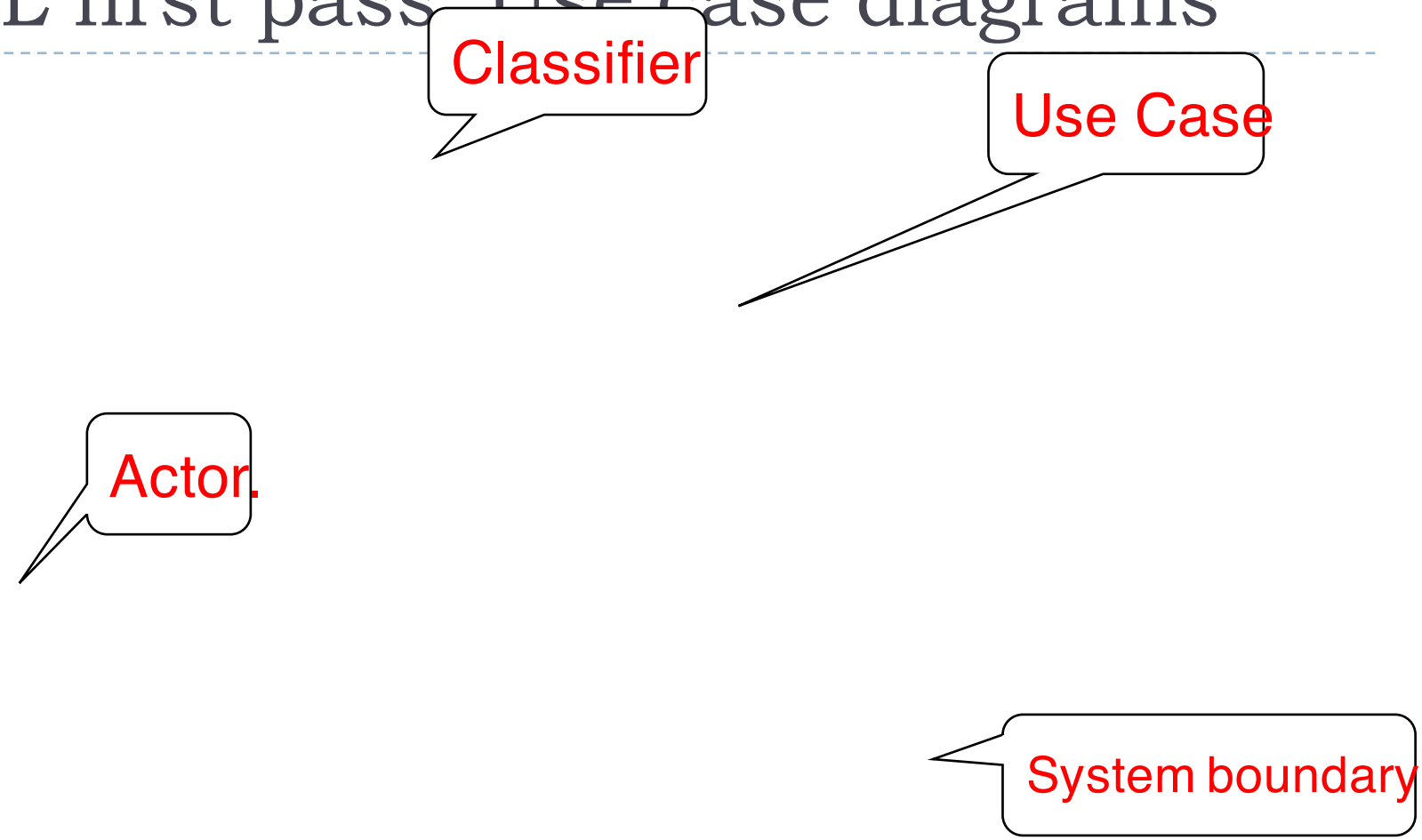
- A model is an abstraction of a system
  - A system that no longer exists
  - An existing system
  - A future system to be built.

We use Models to describe Software Systems

- Object model: What is the structure of the system?
- Functional model: What are the functions of the system?
- Dynamic model: How does the system react to external events?

- System Model: Object model + functional model + dynamic model

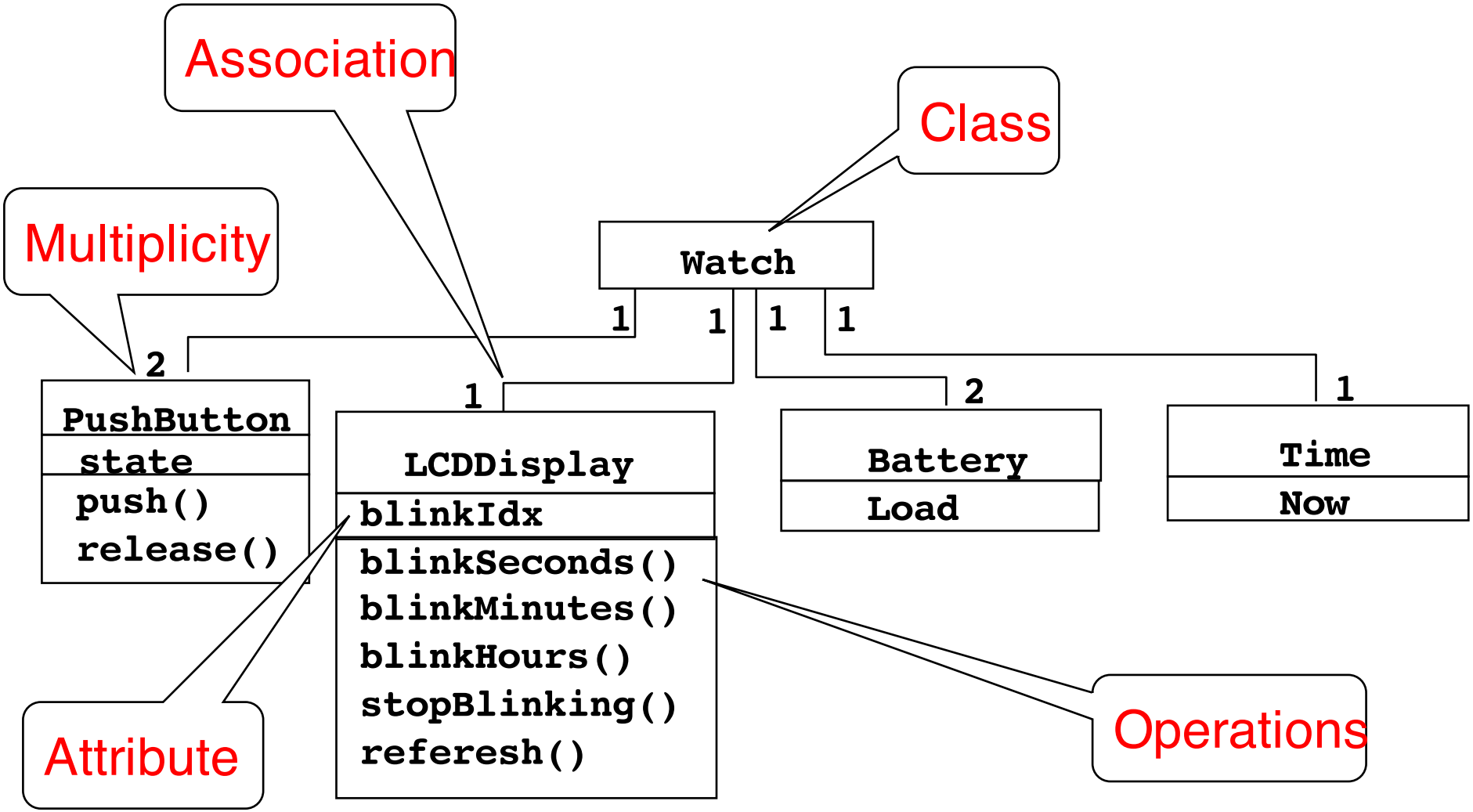
UML first pass: Use case diagrams



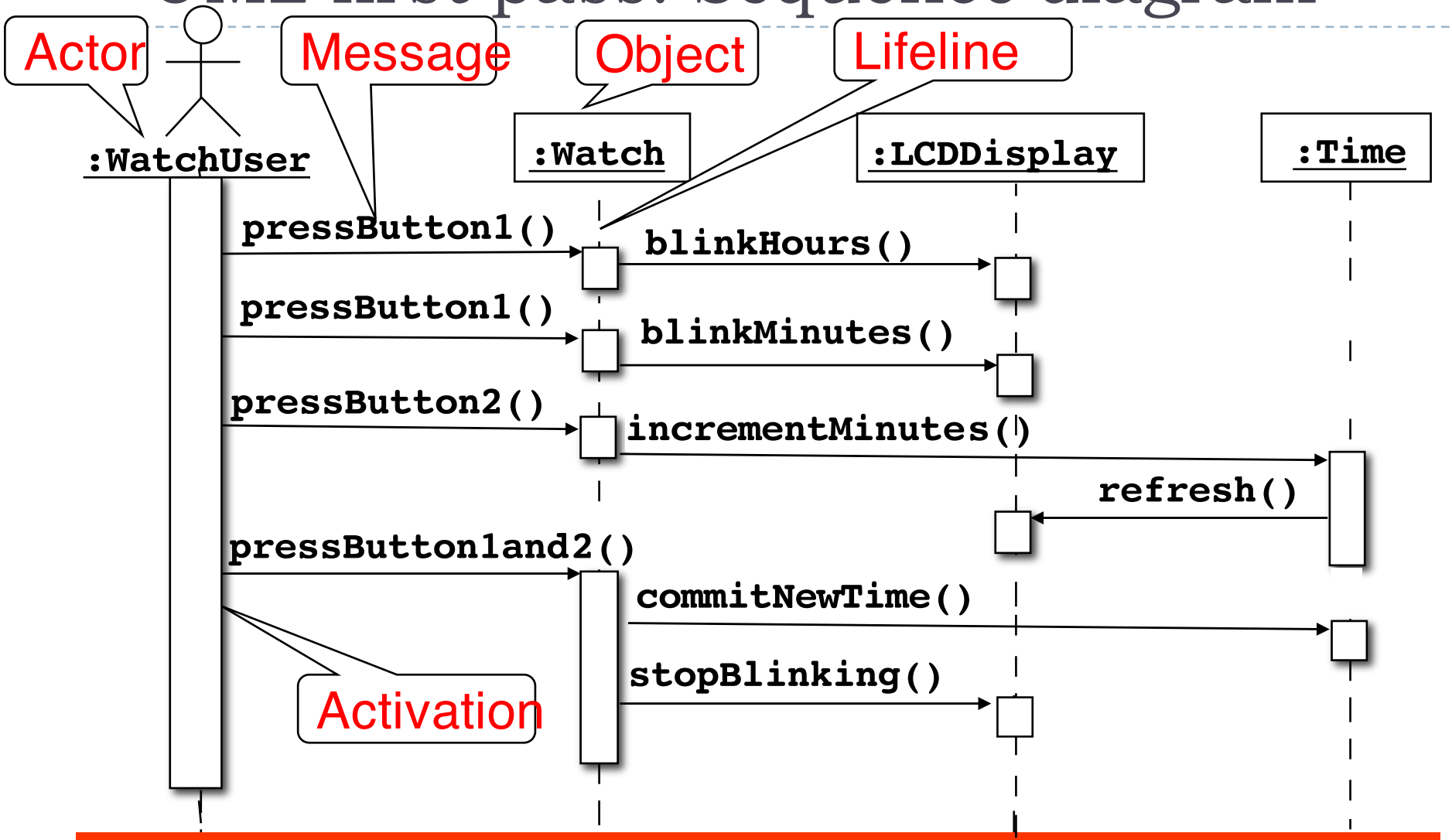
Use case diagrams represent the functionality of the system from user's point of view

UML first pass: Class diagrams

Class diagrams represent the structure of the system



UML first pass: Sequence diagram



Sequence diagrams represent the behavior of a system as messages ("interactions") between *different objects*

What are Formal Methods?

- Formal methods can
  - Be a foundation for designing safety critical systems
  - Be a foundation for describing complex systems
  - Provide support for program development

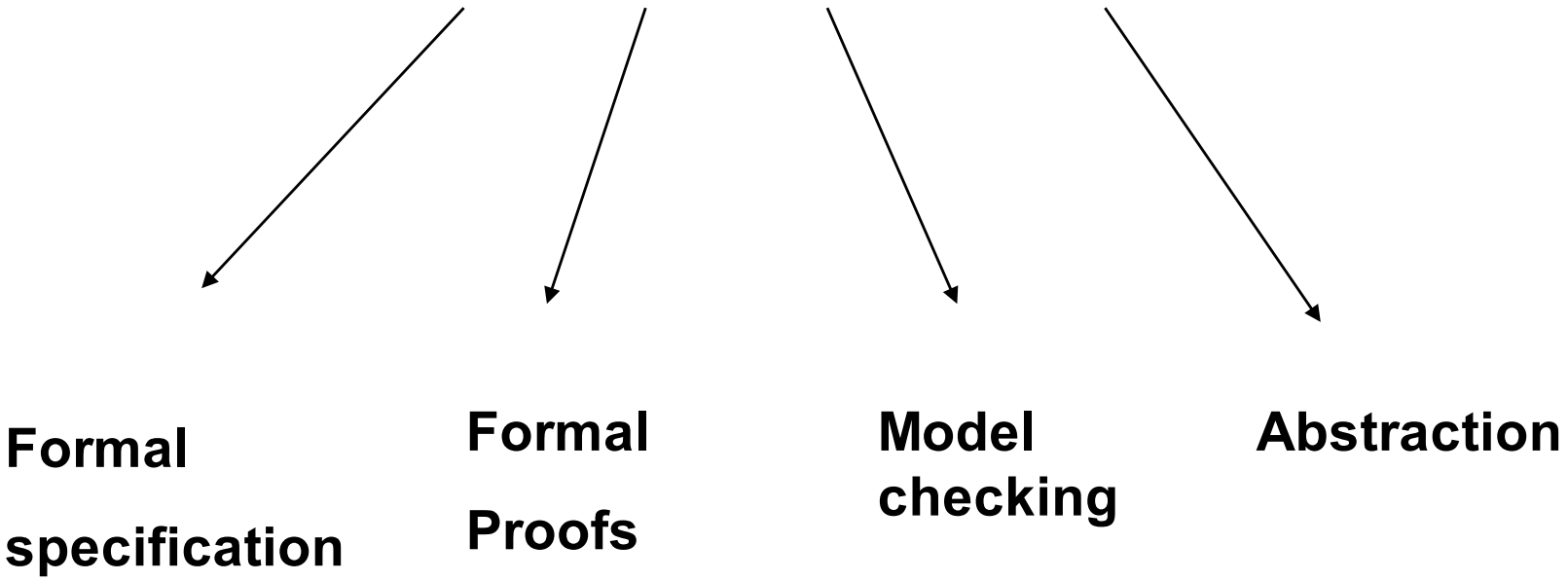
- Techniques and tools based on mathematics and formal logic
- Can assume various forms and levels of rigor
  - Informal
  - Low
  - Medium
  - High

Why Consider Formal Methods?

- The development of a formal specification provides insights and an understanding of the software requirements and software design
  - Clarify customers' requirements
  - Reveal and remove ambiguity, inconsistency and incompleteness
  - Facilitate communication of requirement or design
  - Provides a basis for an elegant software design
  - Traceability
    - System-level requirements should be traceable to subsystems or components

Formal Methods Concepts

Formal Specification Methods



Formal Specification

- The translation of non-mathematical description (diagrams, table, natural language) into a formal specification language
- It represents a concise description of high-level behavior and properties of a system
- Well-defined language semantics support formal deduction about the specification

Type of Formal Specifications

- Model Oriented: Construct a model of the system behavior using mathematical objects like sets, sequences etc.
  - Statecharts, SCR, VDM, Z
  - Petri Nets, CCS, CSP, Automata theoretic models
- Property Oriented: Use a set of necessary properties to describe system behavior, such as axioms, rules etc.
  - Algebraic semantics
  - Temporal logic models.

Formal Proofs

- Proof is an essential part of specification
- Proofs are constructed as a series of small steps, each of which is justified using a small set of rules
- Proofs can be done manually, but usually constructed with some automated assistance

Model Checking

- A technique relies on building a finite model of a system and checking that a desired property holds in that model
- Two general approaches
  - temporal model checking
  - automaton model checking
- Use model checkers
  - SMV

Abstraction

- Representation of the program using a smaller model
- Allows you to focus on the most important central properties and characteristics
- Getting the right level of abstraction is very important in a specification.

Validation

- In the early validation we decided to perform an assessment aiming at examining the overall status of the platform
- **Used Methodology**
  - The quantitative validation
  - Focus Group
- The platform validation involved people with different expertise considering different scenarios
- **Validation Scenarios**