Process Modelling

Natalia Sidorova

Process modelling (2II30)

- Lecturer: dr. Natalia Sidorova (n.sidorova@tue.nl)
- 4 study points = 112 study hours
  18 hours – lectures,
  94 hours – self-study + assignments
- Schedule:
  Lectures: Tuesday 5-6 h.,
  Office hours: Tuesday 15:30-16:30, HG 7.84
- Four assignments
  25, 25, 40 and 10% of the final grade resp.;
- Course website: www.win.tue.nl/∼sidorova/pm/
- studyweb!

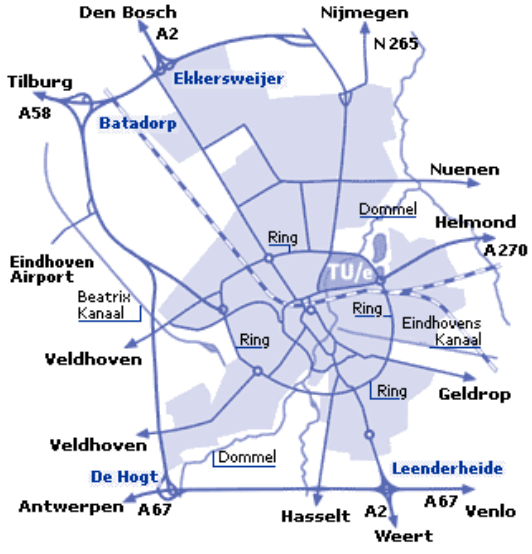
Why to model?

Why to model?

- To help understand, describe, or predict how things work in the real world by exploring a simplified representation of a particular entity or phenomenon.

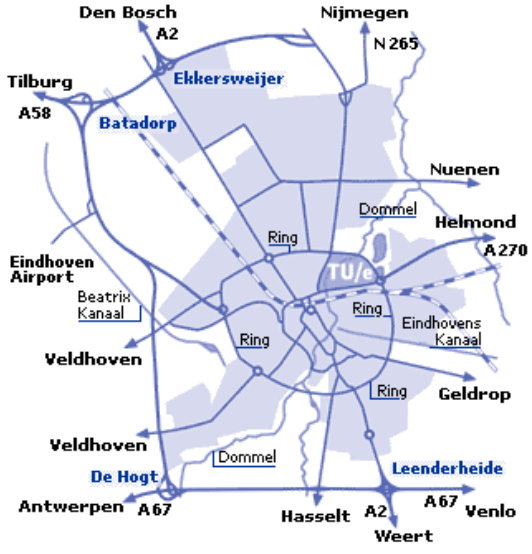
Why to model?

- To help understand, describe, or predict how things work in the real world by exploring a simplified representation of a particular entity or phenomenon.
- A model is useful when …



Why to model?

- To help understand, describe, or predict how things work in the real world by exploring a simplified representation of a particular entity or phenomenon.
- A model is useful when it answers a question!



What is a model?

What is a model?

- A model is a perception of (maybe imaginary) reality.
- A model is a representation of some aspect of reality.
- A model is a simplification of reality.
- A model is a formalized and simplified representation of reality which can be studied “easily”.
- A model hides uninteresting detail, highlights important facts and promotes understanding of the whole.
- The most difficult in modelling is to decide what needs to be left out and what needs to be included, i.e. reach a balance between simplicity and accuracy.

Types of models

Types of models

- Text-based models
- Physical models
- Mathematical models
  - Economic models
  - Sociological models
  - . . .
- ...

How does modelling work?

1. Get a clear picture of the problem.
2. Choose the most appropriate modelling medium to solve the problem and then make a model.
3. Solve the problem on your model.
4. Translate your answer back into the language of the original problem.
5. Check the solution. Is the answer good enough for your needs?

Importance of steps 1,2,4 and 5 is often underestimated!

What is a process?

What is a process?

- Any group of activities organized according to some purpose, is a process.
- These activities may be naturally occurring or designed.
- They are possibly taking up time, space or other resource.
- They may require specific inputs from one or several suppliers.
- A process normally produces some outcome to some customers.

What is there to model?

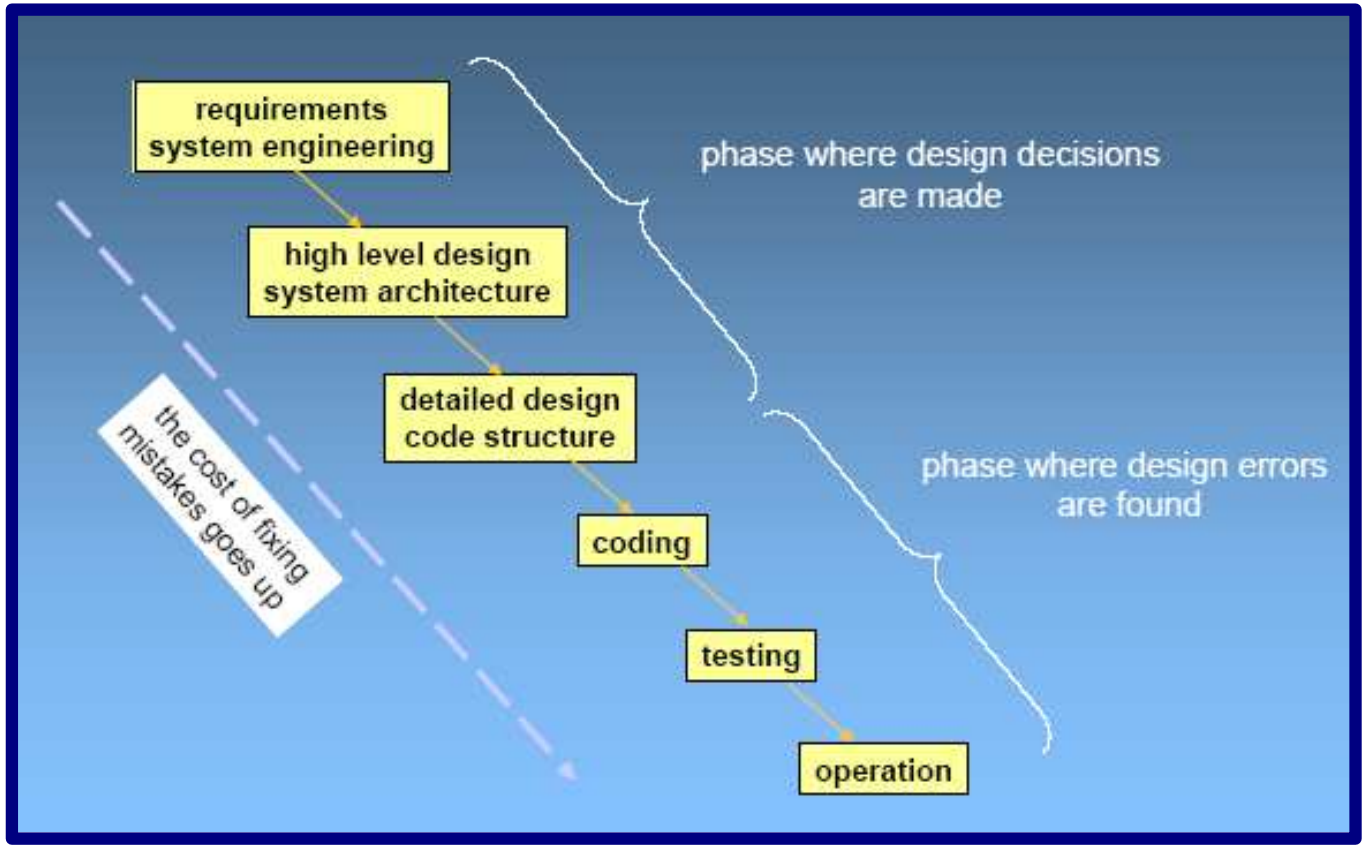
What is there to model?

- What is going to be done?
- Who is going to do it?
- How and why will it be done?
- Who is dependent on it’s being done?
- Duration, costs, probabilities, etc.

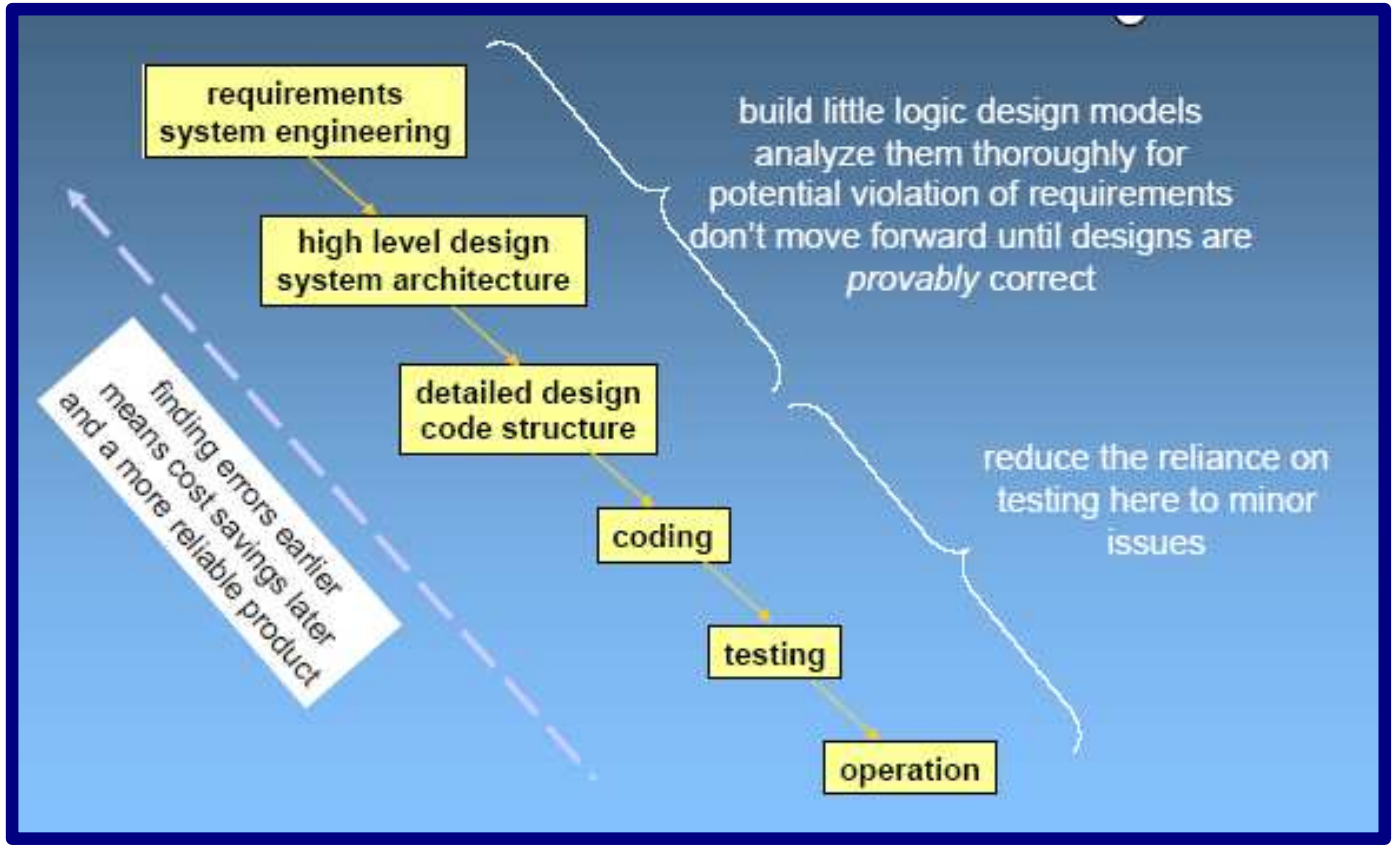
What is here to model?

- Think which components of the phone are relevant,
- which characteristics of those components we care about,
- which assumptions about the environment of the system we can make,
- . . .

Place of modelling in SE



Can it be any better?



Some figures from 1986 (and still valid)

"Observed ranges of defect removal efficiency for programming defect removal methods."

	Efficiency (%)		
Removal Step	Lowest	Modal	Highest
Personal checking of design or docs	15	35	70
Informal group design reviews	30	40	60
Formal design inspections	35	55	75
Formal code inspections	30	60	70
Modelling or prototyping	35	65	80
Desk checking of code	20	40	60
Unit testing (single modules)	10	25	50
Function testing (related modules)	20	35	55
Integration testing (full system)	25	45	60
Field testing (live data)	35	50	65
Cumulative efficiency	93	99	99

Capers Jones, Programming Productivity, 1986, Table 3-25, p. 179

Failing software costs money

- Thousands of dollars for each minute of factory down-time
- Huge losses of monetary and intellectual investment
  - Rocket boost failure (e.g., Arianne 5)
- Business failures associated with buggy software (Ashton-Tate dBase)
- Think what the losses can be due to a failure of e-business applications

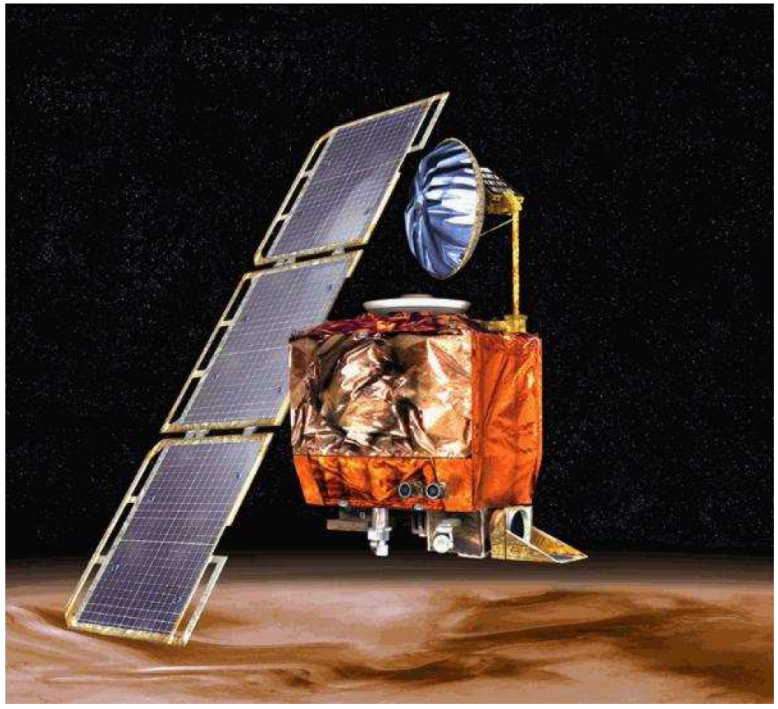
Ariane 5 crash (1996)

The launcher began to disintegrate at about 39 seconds because of high aerodynamic loads resulting from an angle of attack of more than 20 degrees.

The cause of the crash

- Ariane 5 software reused old code from Ariane 4 that was not respecified and retested in new environment
- Ariane 5 (being more powerful than Ariane 4) caused unanticipated floating-point exception (which would have never occurred on Ariane 4), causing an exception to be thrown which was not caught
- The operand error occurred because of an unexpected high value of the horizontal velocity sensed by the platform: The early part of the trajectory of Ariane 5 differs from that of Ariane 4 and results in higher horizontal velocity values.
- Direct cost 500.000.000 EUR, indirect cost 2.000.000.000 EUR

Loss of Mars Climate Orbiter (1999)



Cause: unchecked type mismatch of metric and imperial units

Power shutdown of USS Yorktown

Cause: A sailor mistakenly typed 0 in a field of the kitchen inventory application. Subsequent division by this field cause an arithmetic exception, which propagated through the system, crashed all LAN consoles and remote terminal units, and lead to power shutdown for about 3 hours.

To find about other bugs. . .

http://wwwzenger.informatik.tu-muenchen.de/persons/huckle/bugse.html

In the future ...

the danger is only growing.

Software is becoming the dominant component of society's infrastructure.

- Everything will be monitored/controlled
  - networked watches, devices, . . .
  - autonomous vehicles, intelligent highways,. . .
  - virtual X rather than physical X
- These systems may not have manual backup
- Failures will be very costly and dangerous
- Your job may depend on your ability to produce reliable systems

A success story: Paris metro

Line 14 (Meteor) of the Paris metro:
The traffic is entirely controlled by software.
Both driverless and conventional trains are supported.

A success story: Paris metro

- The safety-critical software components (on board, along the track, on the ground) are developed by Matra Transport using the B abstract machine method [Abrial96].
- Abstract models of components are refined to concrete models, which are then automatically translated to ADA code (87 000 lines).
- The refinement was entirely validated by formal proofs (automatic, or interactive).
- Errors were found and corrected at the modelling phase.
- No single error was found during the conventional testing of the system.

Process modelling in Comp. Sci.

- Reactive systems: Infinite processes responding to their environment.
- Main categories of process modelling reflect the objectives to:
  - facilitate human understanding and communication which requires that a group is able to share a common representation format;
    - models of processes in the system to be built
    - models of (e.g. business-) processes that the software should support
  - support process improvement which requires a basis for defining and analysing processes;
  - reason upon the process correctness.

Correctness(1)

- A system is correct when it meets its requirements. “A design without requirements cannot be right or wrong, it can only be surprising.”
- Verification always starts with the identification of the relevant correctness requirements.
- - It is best to assume that the system is incorrect until the opposite can be shown.
  - Reason: common sense can be misleading, especially in distributed systems design.

Correctness(2)

- You cannot prove a system correct in any absolute sense.
- You can only prove that a system (or a model of a system) does or does not have certain properties.
- It is human judgment to conclude whether having or not having those properties constitutes “correctness” of the design.
- Getting the properties (requirements) right is as important as getting the (model of the) system right.

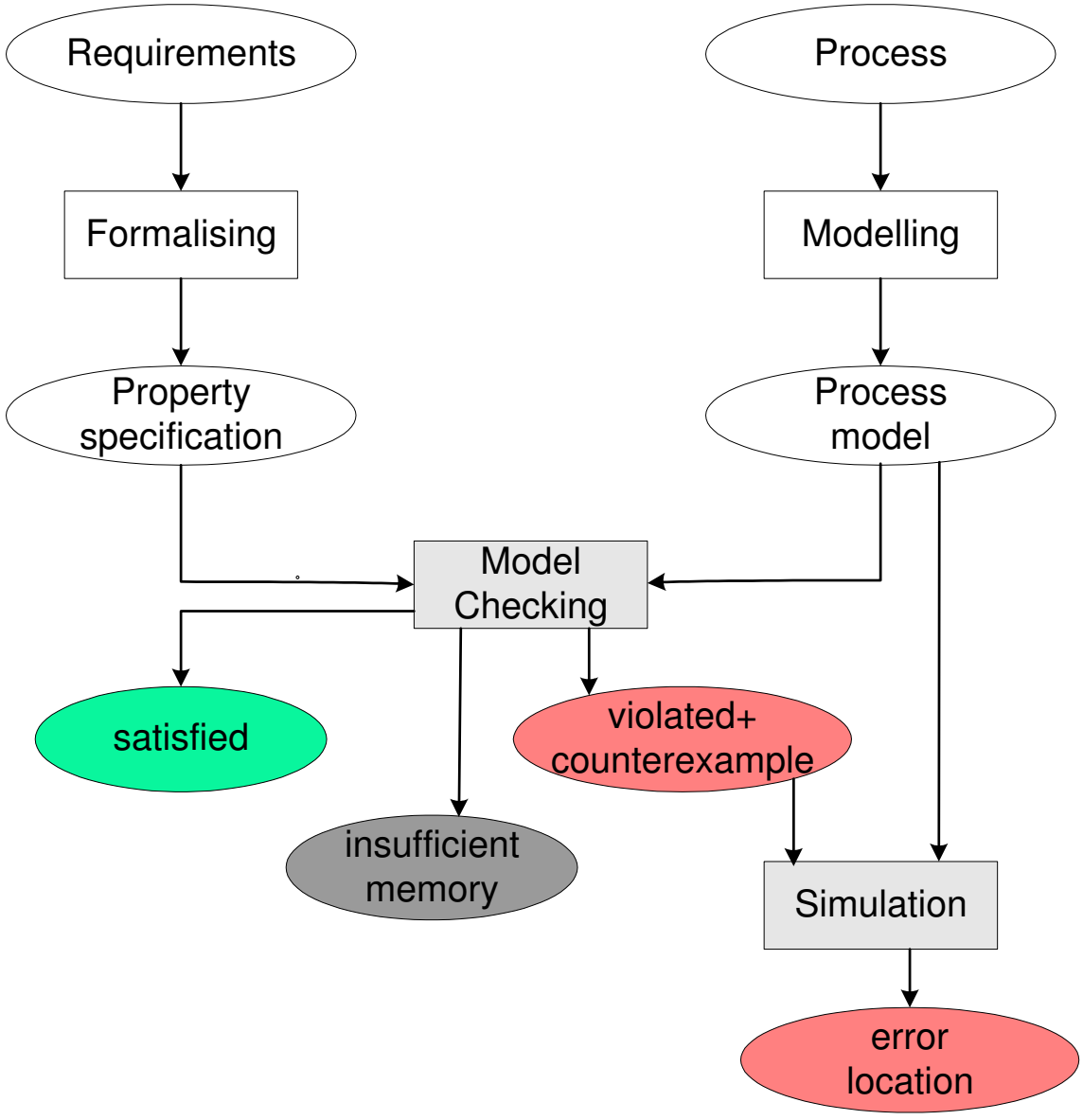
Validation vs. verification

- Model validation = check if the model is true or not, i.e. compare with reality
- Model verification = check that the model does what we think it does

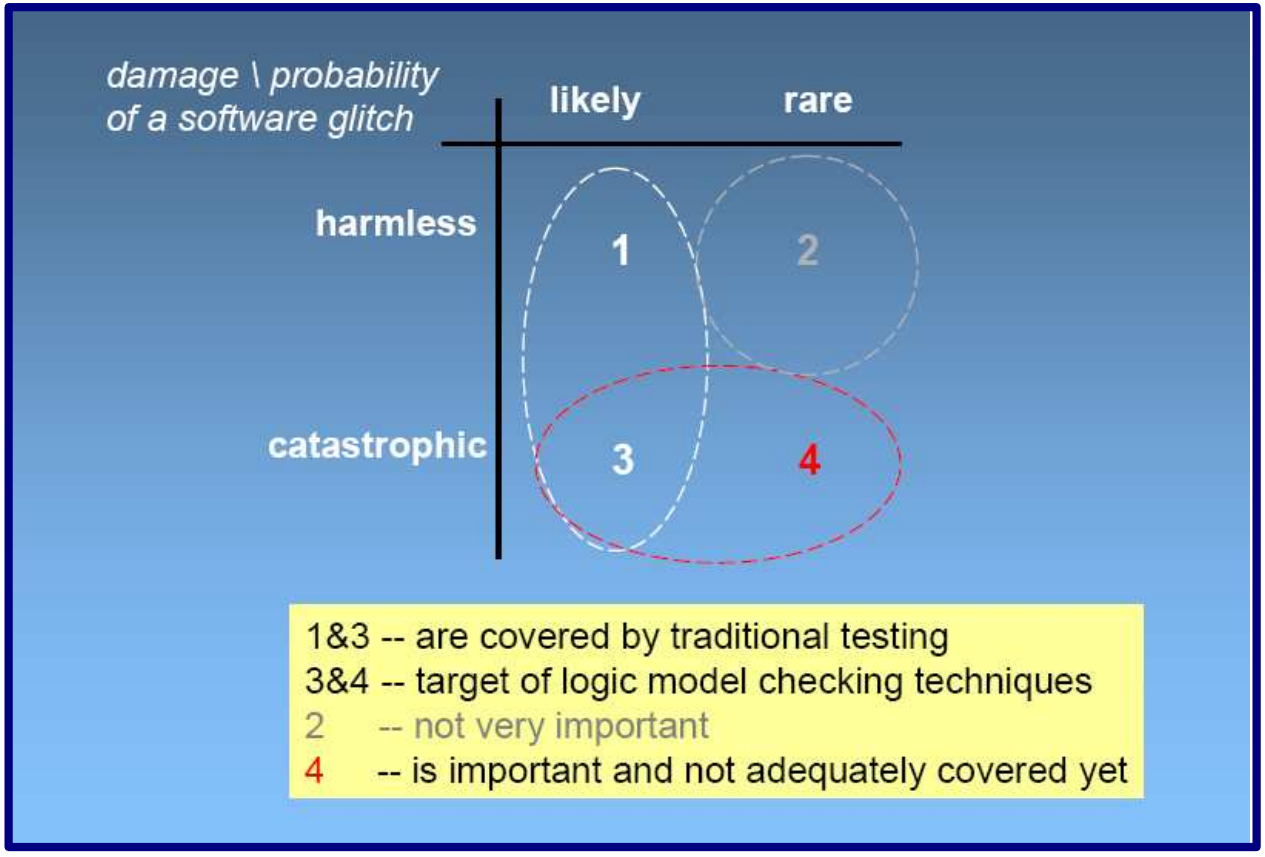
Requirements

- Some requirements are standard:
  – a system should not deadlock
  – no process starvation
  – no explicitly stated assertion should fail
- others are application specific:
  – required system invariants, process assertions
  – effective progress requirements
  – proper termination states (other than the default)
  – general causal and temporal relations on states, e.g., when a request is issued eventually a reply is returned

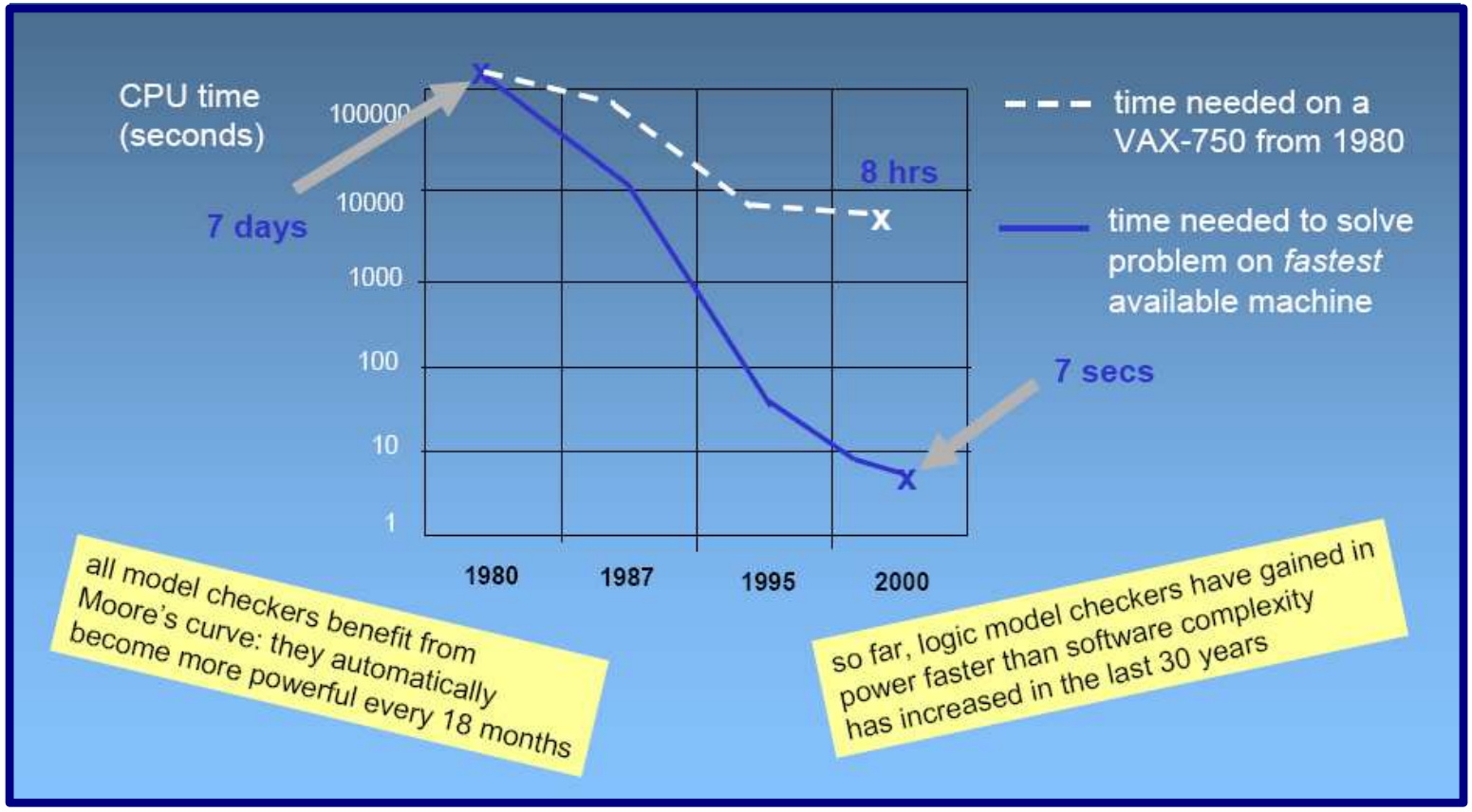
The model-checking approach



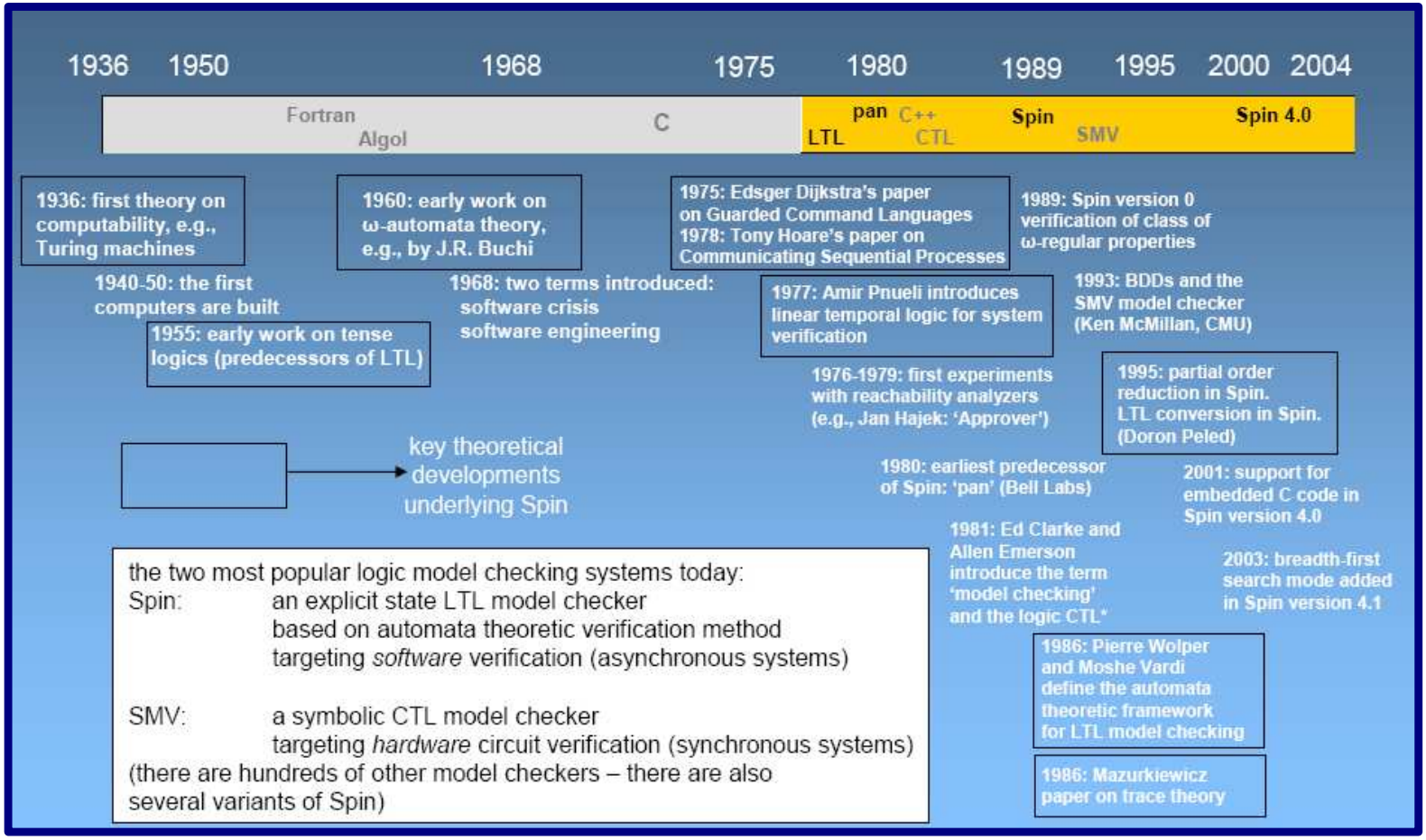
Potentially added value



Is it feasible? (some trends)



Some history



Most of the necessary theory was developed in the 60s and 70s but not fully exploited until fairly recently

Describing models and requirements

- Requirements: Temporal logics
- Models: automata, shared variable automata, communicating automata, Petri nets, process algebra, UML activity diagrams, etc.
  - graphical of textual
  - synchronous or asynchronous communication
  - timed or untimed
  - . . .

Variations in modelling style

- State-based: a condition or mode of being
  - phone is off the hook
  - call is connected
- Event-based: something that happens at a given place and time
  - lifting a phone
  - dialing number 112

Variations in modelling style

- State-based: a condition or mode of being
  - phone is off the hook
  - call is connected
- Event-based: something that happens at a given place and time
  - lifting a phone
  - dialing number 112
- Changes in state are caused by events
- Not all events cause a change in state

Events

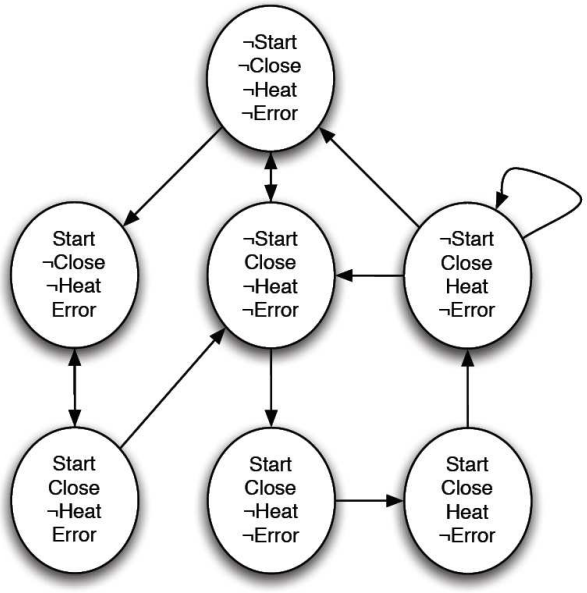
- Event: an occurrence at a specific time and place, that can be described and is worth remembering
- Analyse what events will occur that the system will have to respond to.
- Advantages:

Events

- Event: an occurrence at a specific time and place, that can be described and is worth remembering
- Analyse what events will occur that the system will have to respond to.
- Advantages:
  - Allows you to focus on external environment to keep you at high level view of system (not the functioning of it)
  - End users can easily describe system needs in terms of events that affect their work, so useful when working with users

Automata / transition systems

Microwave oven:



- The machine evolves from one state into another by performing actions given by transitions.
- Both states and transitions may be labelled.
- We may designate initial and final states.

Introducing state variables

- Assignments: a transition may modify the value of variables
- Guards: a transition may be guarded by a condition on the variables.

Interprocess communication

Concurrent processes: collection of two or more sequential processes in operation executing concurrently.

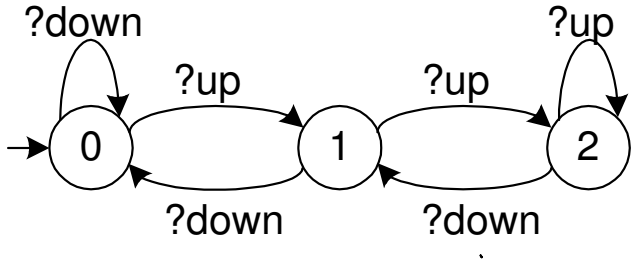
Communication via:

- Shared variables
- Message passing

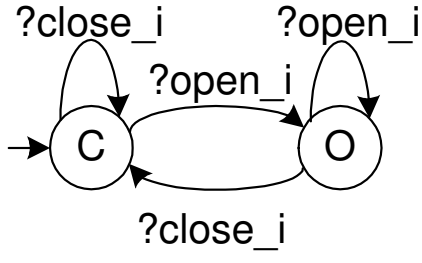
Synchronization by message passing

A smallish elevator:

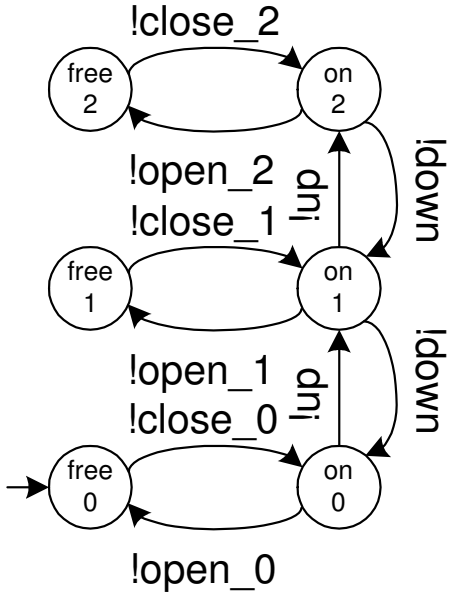
The cabin



The ith door



The controller



Types of message passing (1)

- Asynchronous: using buffers with unbounded capacity.
  - sender may race ahead an unbounded number of steps;
  - sender never blocks;
  - receiver blocks on empty queues.
- Synchronous: no buffering between sender and receiver.
  - sender blocks until receiver is ready to receive;
  - receiver blocks until sender is ready to send.

Types of message passing (2)

- Buffered: using buffers with bounded capacity.
  - sender may race ahead a finite, bounded number of steps;
  - sender blocks on full buffer;
  - receiver blocks on empty buffer.
- Communication channels may be lossy.

The Spin model checker

- Proving correctness of process interactions specified using buffered channels, shared variables, or a combination
- Focus — asynchronous control in software systems
- Spin has “program-like” notation for specifying design choices (Promela)
- Interactive and random simulation options
- Powerful notation for expressing general correctness requirements (LTL)
- Methodology for establishing logical consistency of the design choices against correctness requirements

Homework for this week (part 1)

- Read the information about Spin at http://spinroot.com/spin/whatispin.html, http://spinroot.com/spin/Man/Quick.html and http://spinroot.com/spin/Man/Manual.html
- Download a copy of Spin; downloading and installation instructions: http://wwwis.win.tue.nl/ rpost/spin/
- Start experimenting with Spin a little (just simulation and checks for deadlocks); follow the instructions from http://spinroot.com/spin/Man/GettingStarted.html Take (at least) examples 2 and 5 from http://spinroot.com/spin/Man/Exercises.html Make a model of a simple elevator (see the description on the website)

Homework for this week (part 2)

- read J. Ludewig "Models in software engineering"
- read J.M. Wing "Hints to Specifiers"
- Write down 3 well-reasoned statements for each paper.
  That could be something you especially liked/unliked, something you believe to be very important in process modelling, . . .

Next week:

How to specify requirements formally?