[simula . research laboratory]

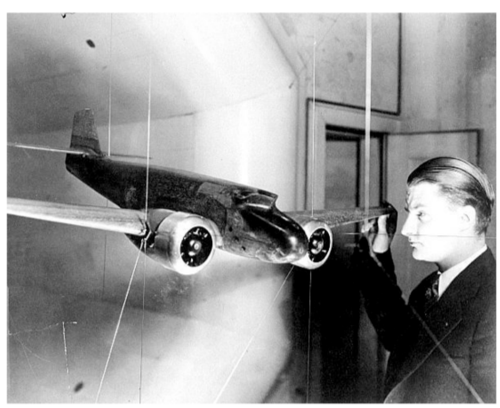
Introduction to Model-driven Software Development and Verification

Lionel C. Briand
Simula Research Laboratory
and
University of Oslo

[simula . research laboratory]

Models in Traditional Engineering

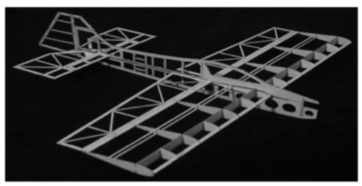
- As old as engineering (e.g., Vitruvius)
- Traditional means of reducing engineering risk

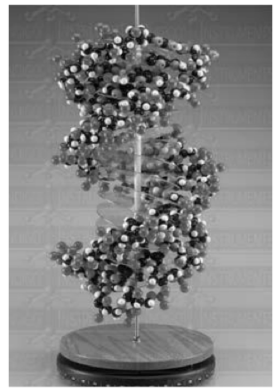


Slide provided by B. Selic

Models in Traditional Engineering

Models used in all branches of engineering



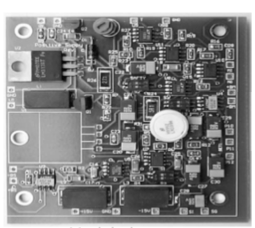


Slide provided by B. Selic

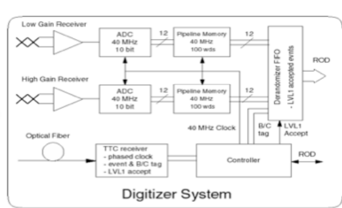
•Inspired from B. Selic presentation during Summer School MDD For DRES 2004 (Brest, September 2004)

What is a model

- Some definitions:
  - A simplified representation used to explain the workings of a real world system or event.
  - A reduced/abstract representation of some system that highlights the properties of interest from a given viewpoint. The viewpoint defines concern, scope and detail level of the model.



•Modeled system



•Functional model

Slide provided by B. Selic

[simula . research laboratory]

Characteristics of useful models

- Abstract
  - Emphasize important aspects while removing irrelevant ones
- Understandable
  - Expressed in a form that is readily understood by observers
- Accurate
  - Faithfully represents the modeled system
- Predictive
  - Can be used to answer questions about the modeled system
- Inexpensive
  - Much cheaper to construct and study than the modeled system

To be useful, engineering models must satisfy all of these characteristics!

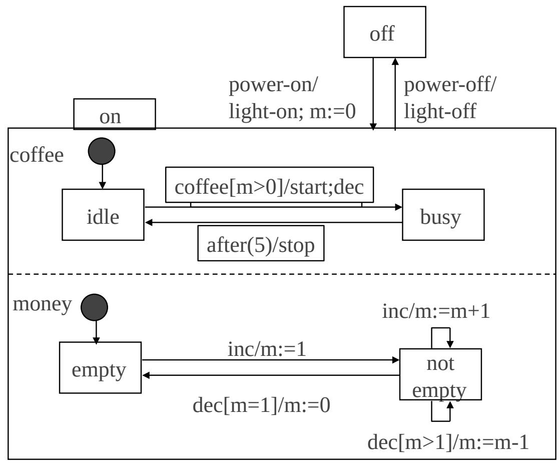
[simula . research laboratory]

Models in Software Engineering

- Not yet part of common practice
- “Software can be easily changed”
- A great deal of academic research, limited practice
- An emerging international standard (OMG): the Unified Modeling Language, Model Driven Architecture
- A quickly growing set of extensions and supporting technologies
- E.g., Automotive and aerospace industries

[simula . research laboratory]

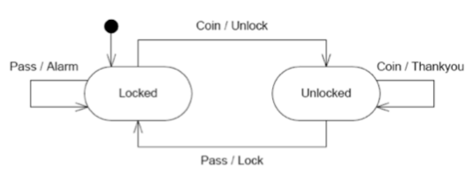
Example Model: State Machine



[simula . research laboratory]

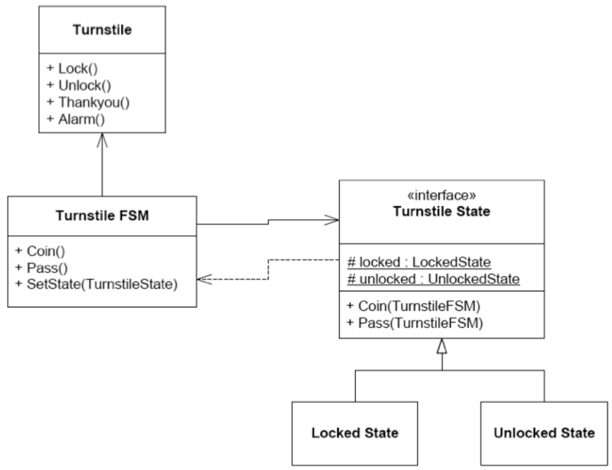
Abstraction?

```
enum State {Locked, Unlocked};
enum Event {Pass, Coin};
void Unlock();
void Lock();
void Thankyou();
void Alarm();
void Transition(Event e)
{
  static State s = Locked;
  switch(s)
  {
  case Locked:
    switch(e)
    {
    case Coin:
      s = Unlocked;
      Unlock();
    break;
    case Pass:
      Alarm();
    break;
}
  break;
  case Unlocked:
    switch(e)
    {
    case Coin:
      Thankyou();
    break;
    case Pass:
      s = Locked;
      Lock();
    break;
    }
  break;
  }
}
```

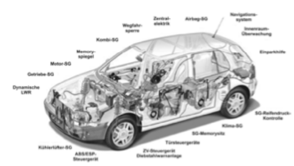


[simula . research laboratory]

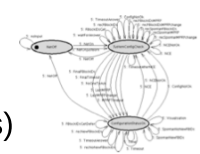
Generate Code with State Pattern



[simula . research laboratory]



Example Models



- States of a system
- Behavior of a system (events, actions)
- Structure of a system, at different levels of abstraction: components, dependencies
- Logical constraints and properties
- Non-functional properties, e.g., response time, safety

[simula . research laboratory]

Goals of Model Driven Software Development (1)

- Increase development speed
  - models are faster to develop and test, as they are at a higher level of abstraction than code
  - code is automatically generated from formal models using one or more well-defined transformation steps
- Enhance software quality
  - due to use of formally-defined modeling languages and automated transformations
  - however, the quality of the transformation has a strong impact on the quality of the final product
- Higher level of reusability
  - separates better reusable code from application-dependent code
  - reuses templates for generating application-dependent code

[simula . research laboratory]

Goals of MDSD (2)

- Improve manageability of complexity through abstraction
  - abstract the problem to focus on some particular points of interest
  - possible to have a set of nearly independent model views
  - iterative modeling may be expressed at different level of fidelity
- Minimize development risks
  - Through analysis and experimentation performed earlier in the design cycle
  - Enable to investigate and compare alternative solutions
- Improve communication between stakeholders:
  - foster information sharing and reuse
  - A model is often better suited than a long speech

© Lionel Briand

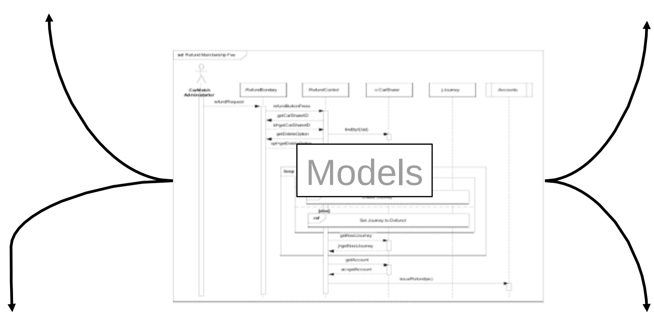
General challenges

- What should we model?
- At what level of detail to model?
- How to exploit/analyze such models?
- What is the cost-benefit of modeling?

Application of Modeling

Communication and knowledge sharing

System architecture and design analysis



Source code generation and system evolution

System verification and validation

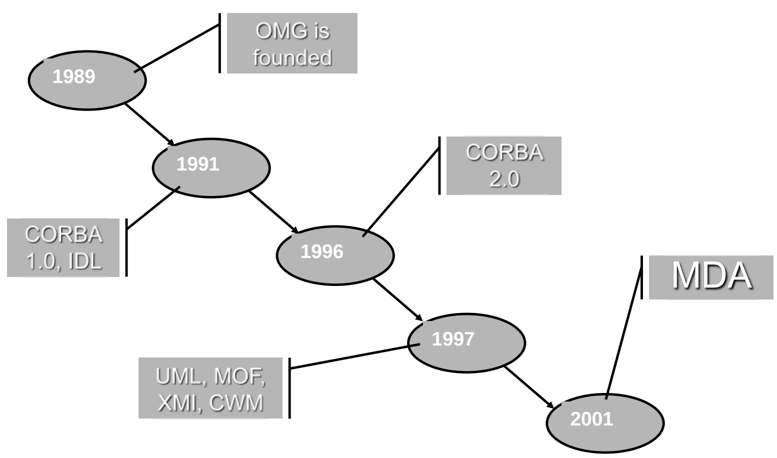
[simula . research laboratory]

The Object Management Group (OMG)

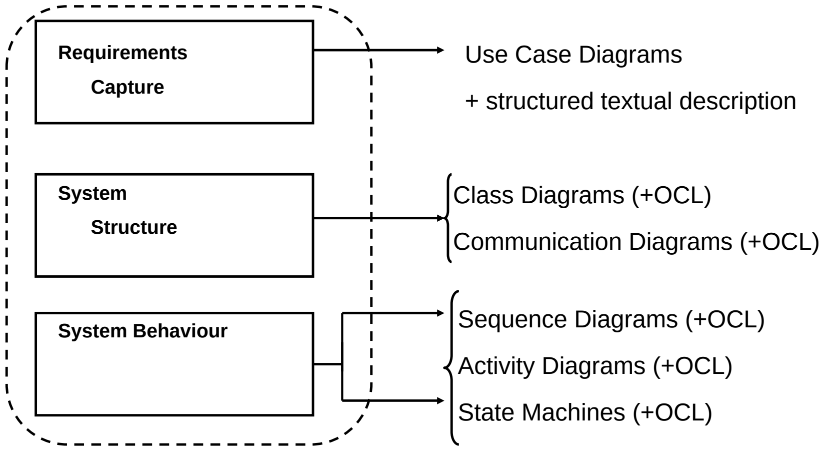
- An open membership and non-profit consortium
- Produces and maintains computer industry specifications for interoperable enterprise applications

[simula . research laboratory]

OMG’s Milestones



OMG’s Unified Modeling Language (UML)

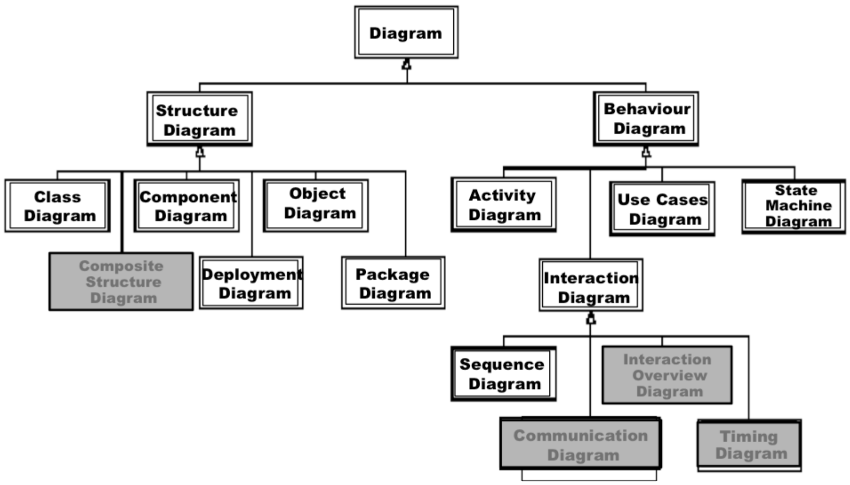


Status: UML 2.0, SysML, profiles

- Compared to UML 1.x, UML 2.0 adds useful features for systems engineering, large scale modeling
- Many “profiles” specialize the UML for specific purposes, e.g., SPT/MARTE for real-time, concurrent systems.
- SysML: Address system engineering requirements (e.g., mechatronics)
- Many commercial modeling tools (e.g., Rational Software Architect, Magicdraw)
- Even an open source UML tools (e.g., Papyrus, TopCased)

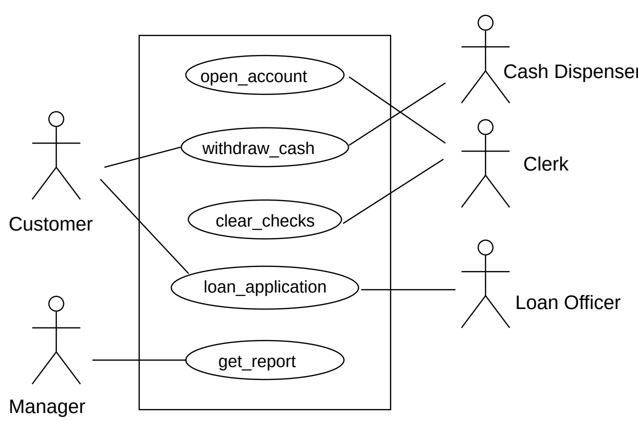
[simula . research laboratory]

UML 2.0

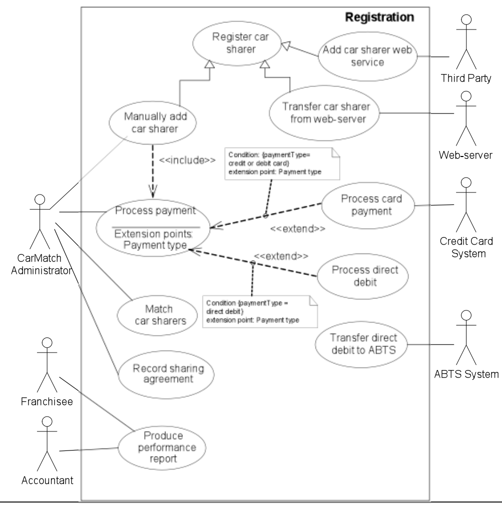


[simula . research laboratory]

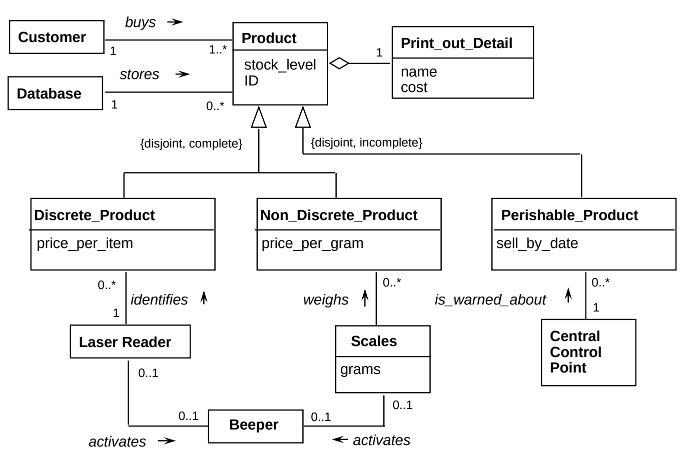
Use Case Diagram



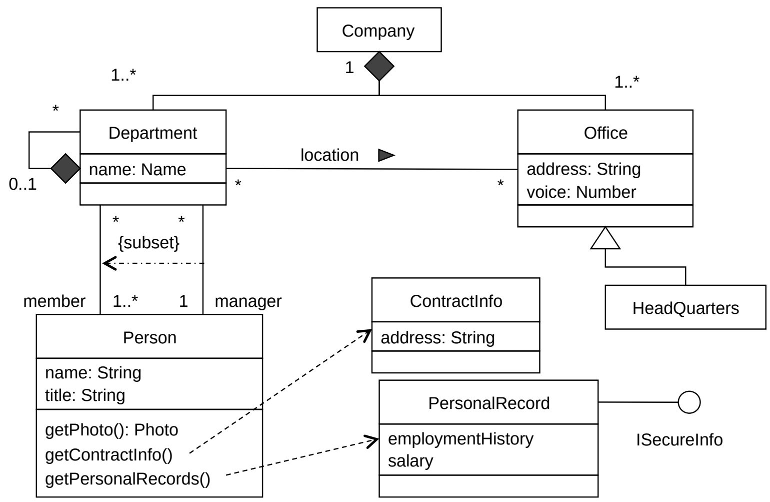
CarMatch System



Class Diagram

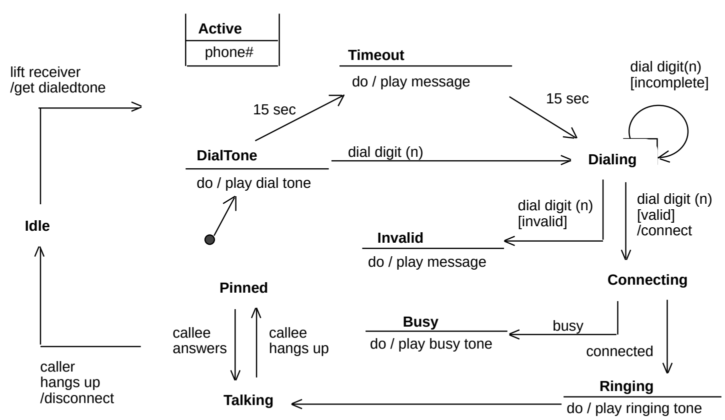


[simula . research laboratory]



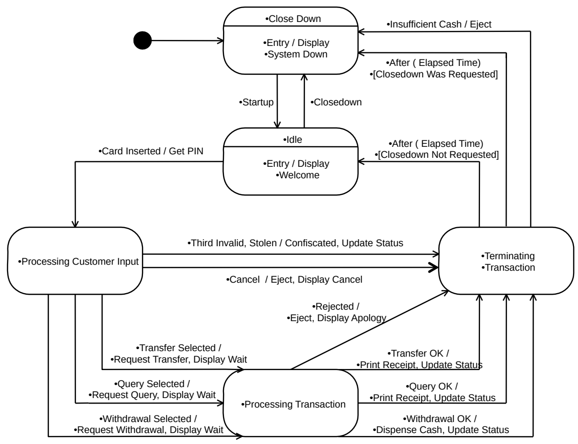
[simula . research laboratory]

State machine diagram



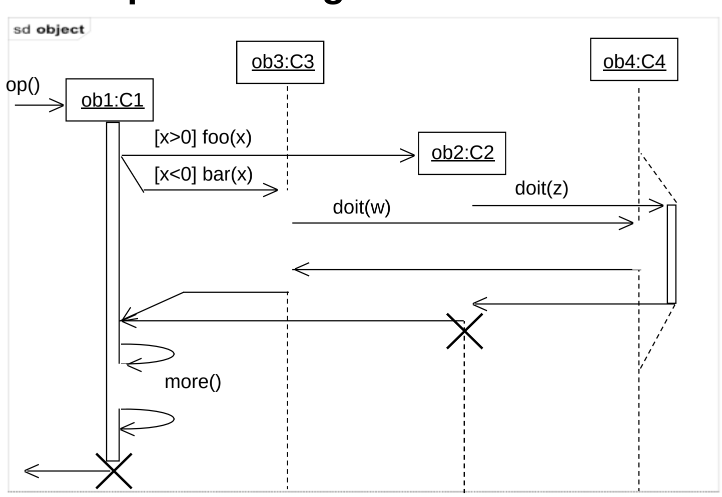
[simula . research laboratory]

Top-Level Statechart for ATM Control Class

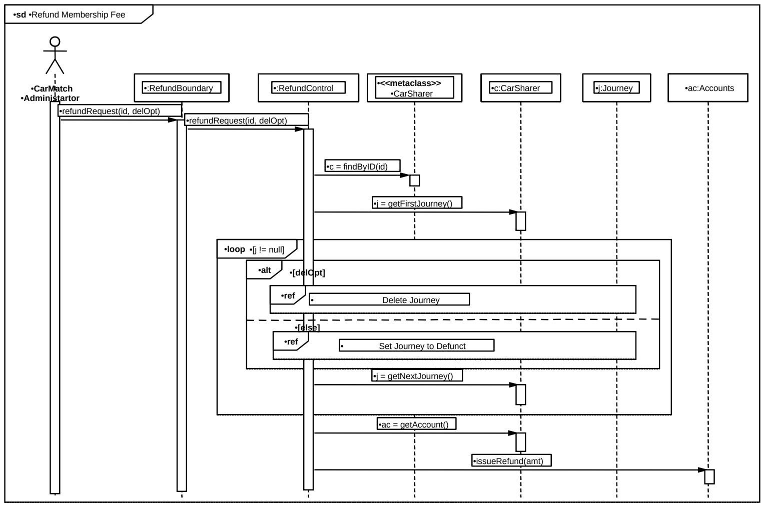


[simula . research laboratory]

Sequence Diagram

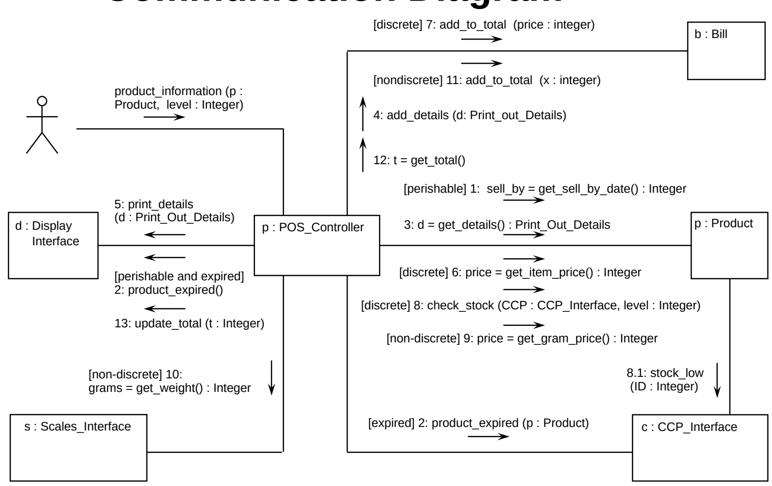


[simula . research laboratory]



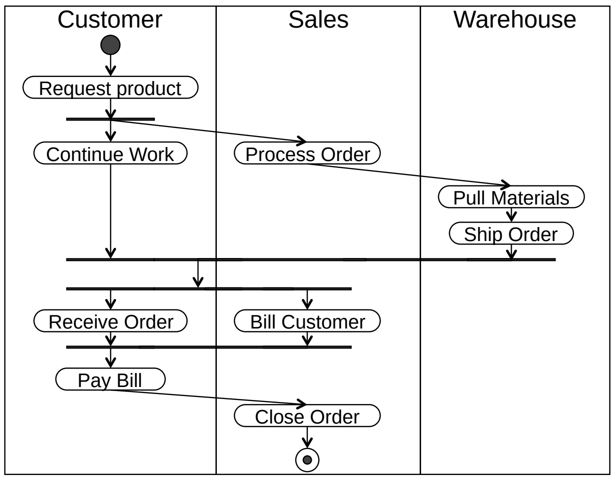
[simula . research laboratory]

Communication Diagram



[simula . research laboratory]

Activity Diagram



[simula . research laboratory]

Misconceptions about UML

- Not a “universal” language: “unified”
- Common core of good practices
- Can (should) use an adequate subset
- Extensible (Profiles) in a way that is supported by tools
- Modeling can take place at different levels of detail and rigor: choice of method

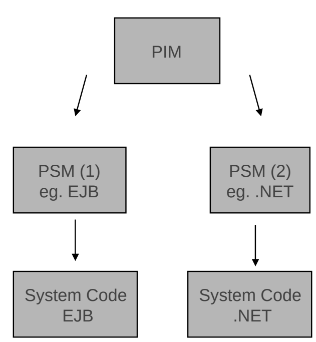
© Lionel Briand

OMG's Model Driven Architecture (MDA)

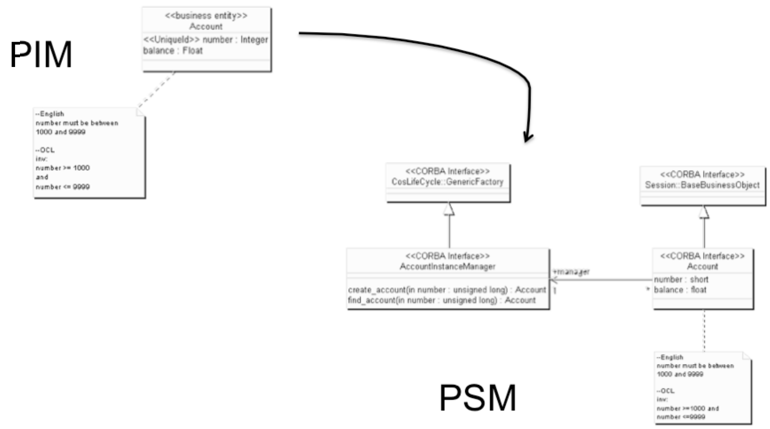
- OMG's MDD standard:
  - Unified Modeling Language (UML) – for expressing software models
  - Meta-Object Facility (MOF) – for describing metamodels
  - MOF-Query/View/Transformation (QVT) – for expressing model transformations
  - XML Metadata Interchange (XMI) – for exchanging metadata expressed in XML
  - Common Warehouse Metamodel (CWM) – for modeling metadata for databases
- Separates the model of the system domain and functionality from the model of the implementation of that functionality on a specific platform.
  - middleware (J2EE, CORBA)
  - operating system (Linux, Windows, etc.)
  - Hardware
- Model: Describes function, structure, and/or behavior of a system (UML)
- Platform Independent Models (PIM), Platform Specific Models (PSM)

Transformations

- PSMs generated from PIMs
- Source code generated from PSMs
- Automated transformations

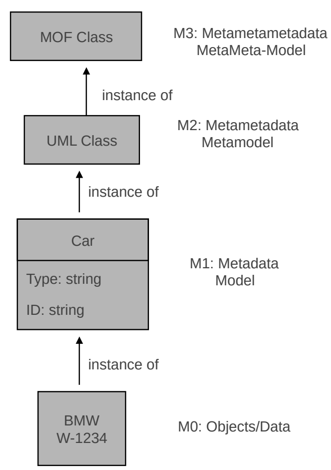


Example Transformation



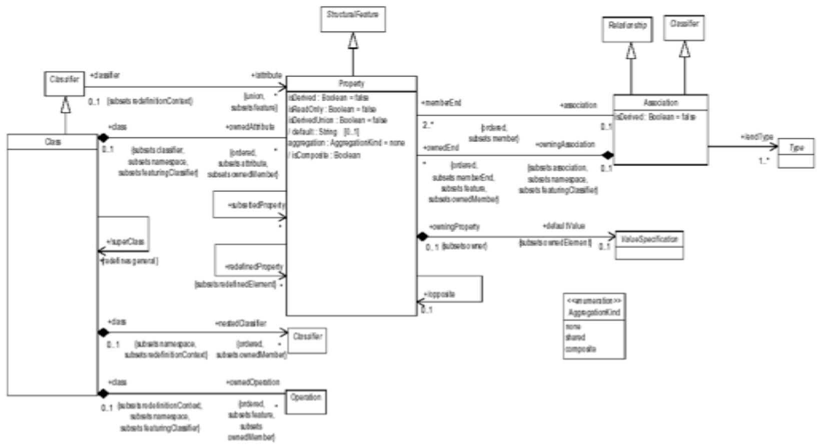
Four Layer Modeling Framework

1. Meta-Object Facility (MOF)
2. (UML) Metamodel
3. (UML) model
4. Data / Instances



[simula . research laboratory]

UML Metamodel



Taken from OMG UML2 Superstructure, Figure 30

[simula . research laboratory]

Examples of Reported Experiences

- Motorola:
  - Top down approach with SDL and then UML for 15 years
  - Code generation
  - 1.2X–4X overall reduction in defects and a 3X improvement in "phase containment of defects", 2X–8X fold in productivity improvement
  - Decrease in inspection and testing times, e.g., 33% reduction in the effort required to develop test cases
- ABB
  - Interviews and questionnaires were used to identify costs and benefits of introducing UML in a large safety-critical project.
  - Design was improved: greater focus and care on design
  - Documentation was improved: more unified structure, many preferred diagrams to text

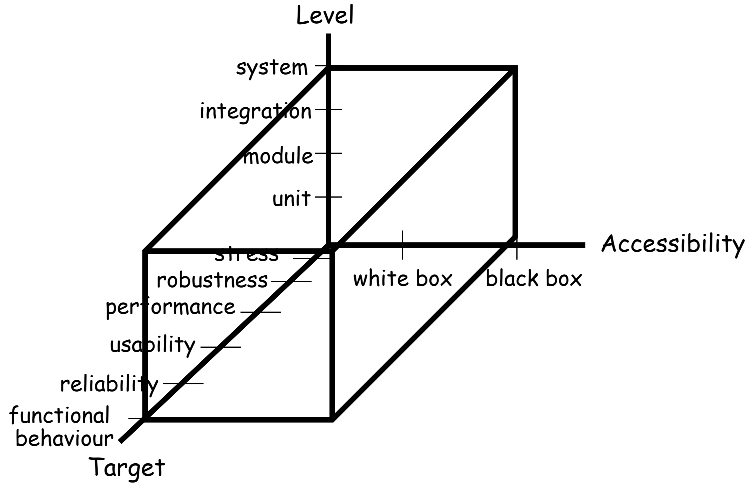
Overview of Model-Based Software Testing

Testing Software Systems

- Short definition: Systematic, controlled system execution and verification of results (outputs, system state)
- Goals of testing:
  - *Effective* at uncovering faults
  - Help *locate* faults for debugging
  - *Repeatable* so that a precise understanding of the fault can be gained and to determine whether it was successfully corrected
  - *Automated* so as to lower the cost and timescale
  - *Systematic* so as to be predictable in terms of its effect on dependability

[simula . research laboratory]

Testing Dimensions



[simula . research laboratory]

Test Automation

- Testers are often perceived as bottlenecks to the delivery of software products. They are being asked to test more and more code in less and less time.
- Test automation is the use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions.
- Model-based testing (MBT) is a way to achieve effective test automation

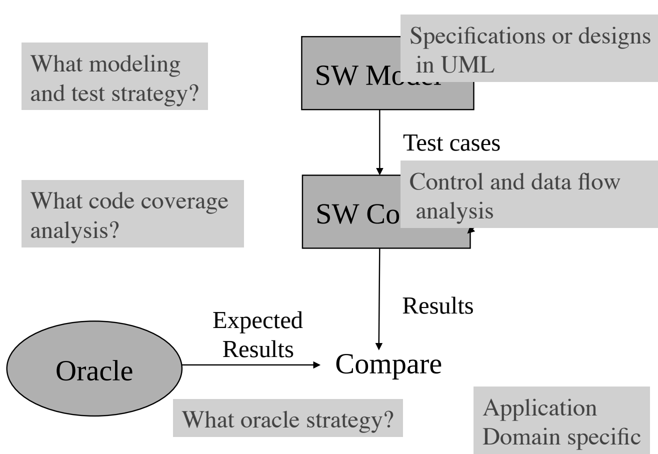
[simula . research laboratory]

Tasks to Automate

- *Test design*: selection of test cases to cover requirements of SUT
- *Test execution*: manual entry of test cases and associated data
- *Test coverage*: manual analysis to check if all combinations of logic tested
- *Test results analysis*: manual analysis to check if actual outputs/outcomes match expected ones

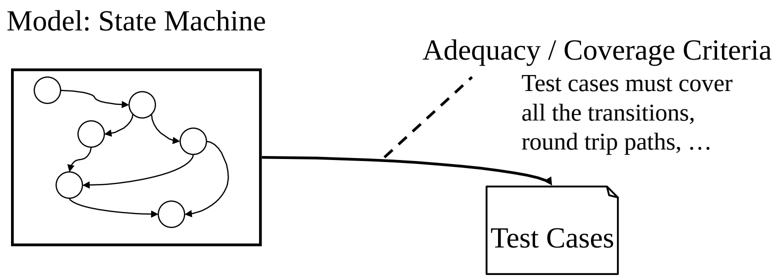
[simula . research laboratory]

Test Case Generation and Execution



[simula . research laboratory]

Adequacy Criteria on Test Models



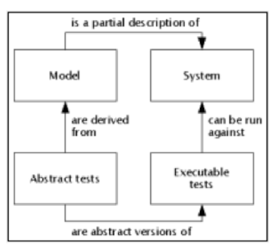
In this context:

- Complex class clusters are commonly described with state machines during OO analysis & design
- Test models (paths) are derived from state machines
- Test cases are method call sequences on classes (clusters).

[simula . research laboratory]

MBT Definition

- "Model-based testing is a testing technique where the runtime behavior of an implementation under test is checked against predictions made by a formal specification, or model." - Colin Campbell, Microsoft Research
- Model-based testing is software testing in which test cases are derived in whole or in part from a model that describes some (usually functional) aspects of the system under test (SUT). - Wikipedia

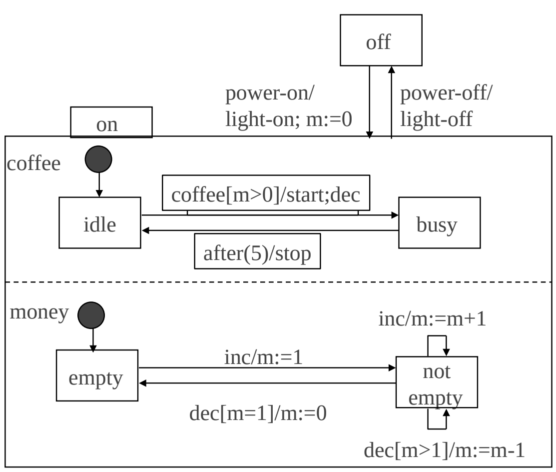


Models in Software Testing

- Finite State Machines
- Statecharts
- Markov Chains
- Grammars
- Cause-effect graphs
- No models fit all intents and purposes. Consequently, for each situation decisions need to be made as to what model (or collection of models) are most suitable.
- Unified Modeling Language (UML): Testing Profile

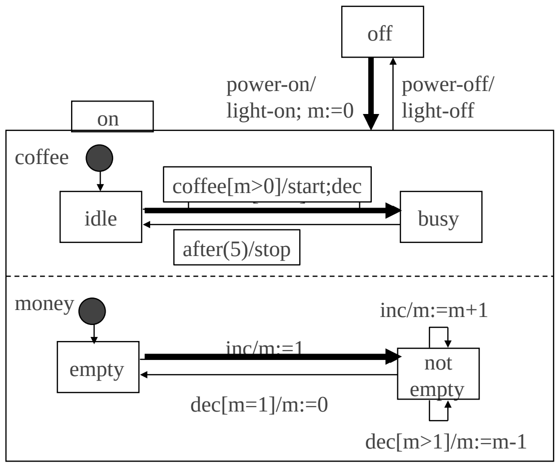
[simula . research laboratory]

Example with State Machine



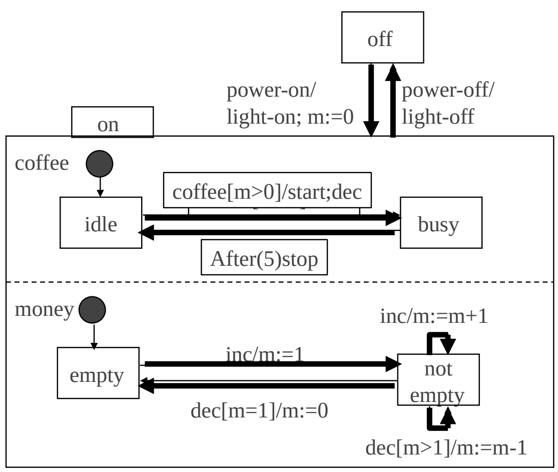
[simula . research laboratory]

State coverage



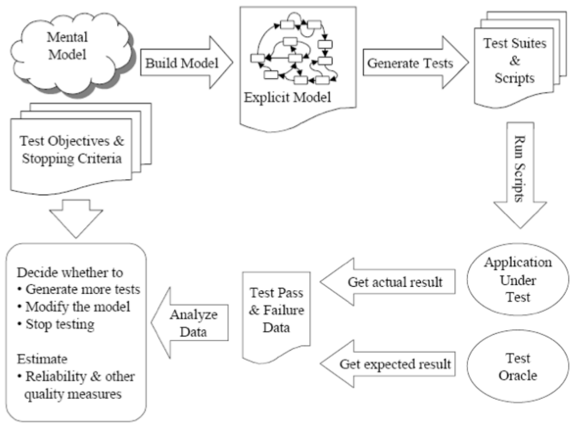
[simula . research laboratory]

Transition coverage



© Lionel Briand

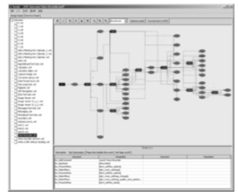
Process of MBT



El-Far and Whittaker, 2001

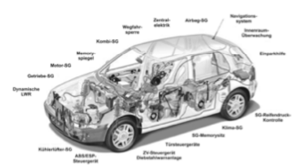
MBT Benefits

- *Comprehensive tests*: models determine logical paths, locations of program boundaries, help identify reachability problems
- *Defect discovery*: studies suggest model-based testing results in early and efficient defect detection, significant Return On Investment
- *Improved requirements*: testable requirement has to be complete, consistent, unambiguous; testing may expose “feature interaction” requirement defects
- A model serves as a unifying point of reference that all teams and individuals involved in the development process can share, reuse, and benefit from. For example, confusion as to whether the system under test needs to satisfy a particular requirement can be resolved by examining the model.
- Most popular models have a rich theoretical background that makes numerous tasks such as generating large suites of test cases easy to automate. For example, graph theory readily solves automated test generation for finite state machine models.

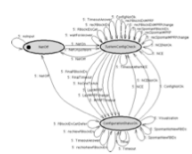


NOKIA Study

- KENDO tool, based on finite state machines
- Models created from scratch or from UI specifications
- 100% transition coverage with Chinese postman algorithm
- Test cases executed on real device or emulator
- On two applications (Image viewer, voice recorder)
- 1000 man-hours would be saved yearly compared to manual testing
- Even more savings would be expected if GUI specification were more precise and formal as a basis for (test) models

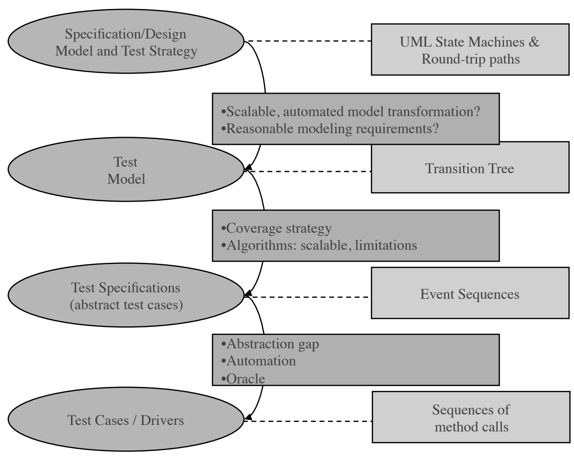


BMW Study



- Pretschner et al. 2005
- SUT: Network controller
- Dev. Model: Informal MSCs
- AutoFocus modeling: System structure diagrams, EFSMs were the test models
- MBT increased detected requirements errors sixfold and detected programming errors by 30% .
- Automation does not increase test effectiveness.
- Model and implementation coverages correlate moderately.

MBT Challenges: Technical



MBT Challenges: Practical

- Skills, time, and other resources need to be allocated for making preparations, overcoming common difficulties, and working around the major MBT drawbacks.
- This overhead needs to be weighed against potential rewards in order to determine whether a model-based technique is sensible to the task at hand.
- Certain skills of testers (basic familiarity with formal languages, automata theory, and perhaps graph theory and elementary statistics).
- The most prominent problem for state models (and most other similar models) is state space explosion. Briefly, models of almost any non-trivial software functionality can grow beyond management even with tool support.

Industrial Research at Simula

- Problem-driven research on large scale, complex software-based systems
- Risk-driven testing at Telenor
- Robustness and stress testing at Tandberg
- State modeling and testing at ABB
- Safety analysis and testing at DNV

Thank you!

[simula . research laboratory]

Acknowledgments

- Bran Selic
- Dorina Petriu
- Davide Buscali
- Johann Oberleitner