Partially based on the DSL awareness training of ESI.

Model-Based Development

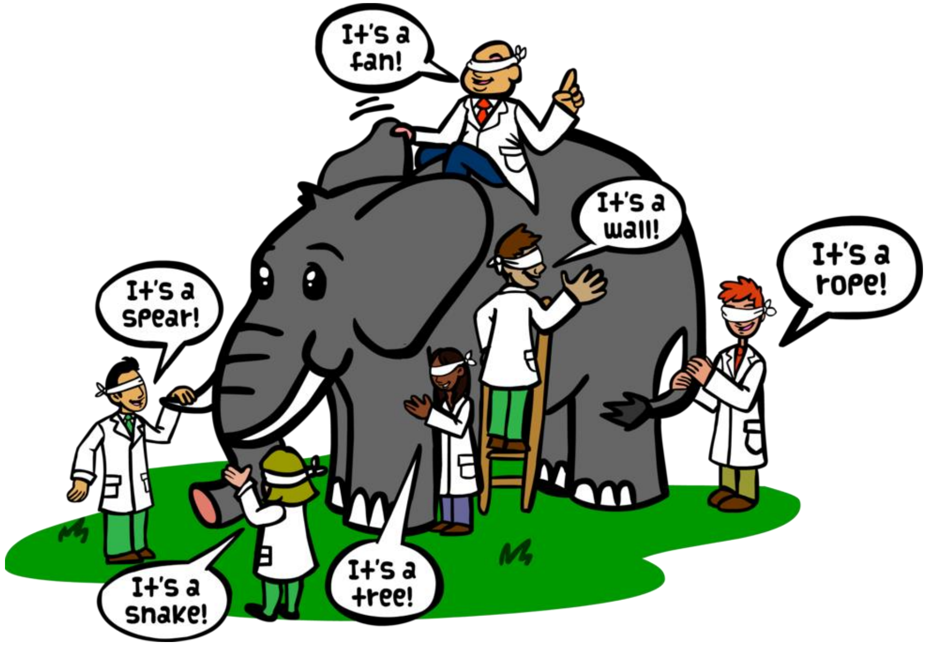
Software Systems (Computer & Embedded Systems Engineering)

Arjan Mooij

January 2023 (week 8)

An initiative of industry, academia and TNO

Modeling for a specific purpose



- **In this course we have focused on the following 3 modeling techniques:**
    - Unified Modeling Language (UML)
    - Finite-State Machines (FSM)
    - Domain-Specific Languages (DSL)

Techniques for dealing with complexity

A. **Abstraction: Identify high-level concepts that hide low-level details**
- Unified Modeling Language: generic high-level concepts that ignore implementation details
- Finite-State Machines: generic concepts that hide language-specific implementation patterns
- Domain-Specific Languages: domain-specific application concepts instead of implementation details

B. **Boundedness: Impose acceptable restrictions on the considered problem space**
- Unified Modeling Language: limited to specific aspects of the system
- Finite-State Machines: limited to a specific aspect (behavior) of a component
- Domain-Specific Languages: limited to a specific domain aspect

C. **Composition: Divide one problem into multiple independent smaller problems**
- Unified Modeling Language: multiple views on the same system, and break-down using component diagrams
- Finite-State Machines: composite and orthogonal state machines (e.g., one per component)
- Domain-Specific Languages: (depending on the specific language)

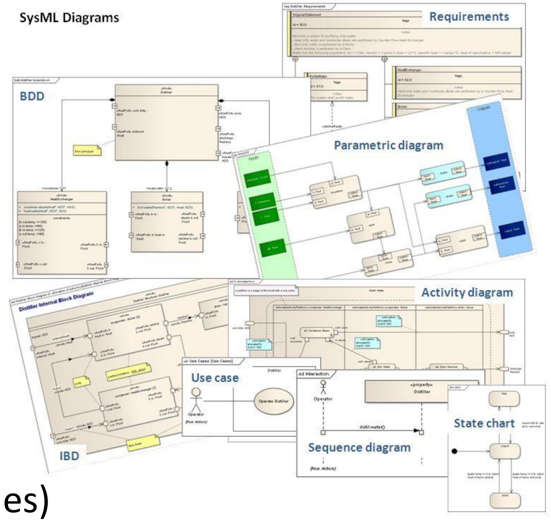
D. **Duplication: Use multiple overlapping approaches for the same problem**
- Unified Modeling Language: multiple related views on the same system
- Finite-State Machines: simulation, verification and testing the generated code (to get correct code)
- Domain-Specific Languages: generating both code and tests (to be able to detect errors in the generators)

Unified Modeling Language (UML)

Model-Based Development

Unified Modeling Language (UML)

- Which alternatives do you know?



Examples of related languages:

- **Unified Modeling Language (UML)**
  - OMG standard focused on <u>software</u> engineering
    - 14 diagram types
- **Systems Modeling Language (SysML)**
  - OMG standard focused on <u>systems</u> engineering
    - 7 diagram types based on UML's 14 diagrams types (sometimes with slightly different names)
    - 2 new diagram types:
      - Requirement diagram: requirements engineering (functional, performance and interface)
      - Parametric diagram: performance analysis and quantitative analysis
- **Informal box/arrow pictures**
  - Focused on general drawings
    - No constraints whatsoever on the type of diagram
    - Flexibility may look nice, but would the notation be understandable?

Unified Modeling Language (UML)

- Which tools do you know?
- Which features distinguish them?

Examples of related tools:

- **PlantUML**
  - Command-line tool for single diagrams, integrated with many textual editors, models are easy to generate from a DSL
  - Open source licenses, no commercial support
- **Graphical UML editors**
  - Graphical editing of diagrams, (sometimes) with elements that can be used across multiple diagrams
  - Code import and code generation (sometimes)
  - Some specific tools:

Enterprise Architect:	proprietary license,	commercial support by Sparx Systems (Australia)
LucidChart	proprietary license,	commercial support by Lucid (USA)
MagicDraw:	proprietary license,	commercial support by Dassault Systèmes (France)
Modelio:	open source licenses,	commercial support by ModelioSoft (France)
Rational Rhapsody:	proprietary license,	commercial support by IBM (USA)
UML Designer:	Eclipse Public License,	commercial support by Obeo (France)

- **General drawing tools (like Powerpoint / Visio)**
  - Graphical editing of diagrams, but no/limited specific UML support

➔ **Note: different tools support different subsets of UML!**

Finite-State Machines (FSM)

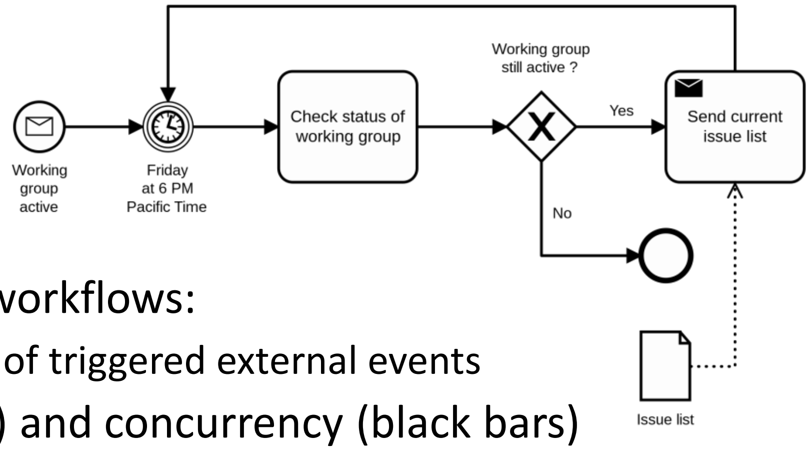
Model-Based Development

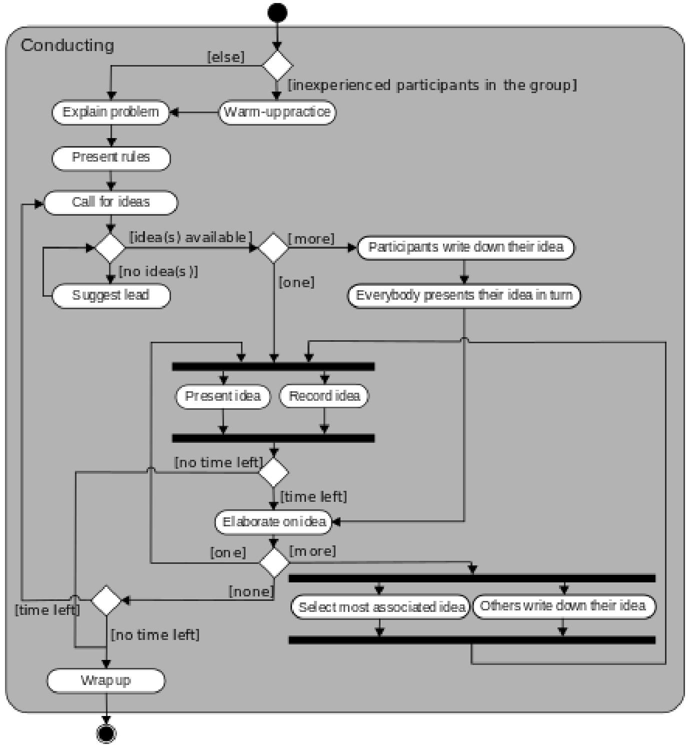
Finite-State Machines (FSM)

- Which alternatives do you know?

Examples of related languages:

- **Finite-State Machines (FSM)**
  - UML diagram type
- **Activity diagram**
  - UML diagram type
  - Focused on organizational workflows:
    - Internal activities instead of triggered external events
  - Concepts: choice (diamond) and concurrency (black bars)
- **Business Process Model and Notation (BPMN)**
  - Similar to UML's activity diagram


Working group active
Friday at 6 PM Pacific Time
Check status of working group
Working group still active ?
Yes
No
Send current issue list
Issue list



Finite-State Machines (FSM)

- Which tools do you know?
- Which features distinguish them?

Examples of related tools:

- **YAKINDU Statechart Tools**
  - Graphical editing, but not linked to other UML views
  - Simulator and code generator
- **Cordis SUITE**
  - Graphical editing
  - Simulator and code generator for PLC (Programmable Logic Controller)
- **Graphical UML editors**
  - Graphical editing, linked to other UML views
  - Usually no simulator nor code generator
- **General drawing tools (like Powerpoint / Visio)**
  - Graphical editing, but no specific FSM support
  - No simulator nor code generator

Domain-Specific Languages (DSL)

Model-Based Development

Domain-Specific Languages (DSL)

- Which tools do you know?
- Which features distinguish them?

Some examples:

- **MetaEdit+ (graphical)**
  - Proprietary, commercial support by MetaCase (Finland)
- **MetaProgrammingSystem (projectional editing: text and graphical)**
  - Apache 2.0 license, commercial support by JetBrains (Czech Republic)
- **Rascal (textual)**
  - BSD license, commercial support by Swat.engineering (The Netherlands)
- **Spoofax (textual)**
  - Apache 2.0 license, no commercial support (developed in PL group of TU Delft)
- **Xtext (textual) and Sirius (graphical)**
  - Eclipse Public License, commercial support by TypeFox (Germany) and Obeo (France)

General-purpose Programming Language (GPL) ←→ DSL

GPL advantages

- Many people already use them and know them (company can choose many employees)
- Freedom, not limited to a specific domain
- Wide community that can help with problems
- (Generate only code)

DSL disadvantages

- You need to make it, so maybe not cost-effective if the problem is simple enough
- May be a completely paradigm from what you are used to

GPL disadvantages

- Freedom to shoot yourself in the foot
- A lot hard to learn – More complicated, many features
- Really large code bases that are harder to maintain
- Much more about the system than the problem domain

DSL advantages

- Easy to express stuff, because you are so restricted
- Not bother to implement domain specific concepts
- Generate multiple artifacts (code, documents)
- Can be designed in a way that people outside the field (of programming) can understand it

General-purpose Programming Language (GPL) ←→ DSL

GPL advantages

- …

DSL disadvantages

- …

GPL disadvantages

- …

DSL advantages

- …

Comparison

General-purpose Programming Languages (GPL)	Domain-Specific Languages (DSL)
+ Wide range of application areas	- Restricted to one application area
+ Widely-used, well-known languages	- Custom languages must be developed
+ Single off-the-shelf development tool	- Extra development tool and build step
- Useable by programmers only	+ Also useable by non-programmers
- Focus on technical implementation	+ Focus on domain requirements
- Difficult to avoid language abuse	+ Easy to control the possible use
- Limited set of early validation rules	+ More validation in application area
- Compiler is difficult to customize	+ Generate many customized artifacts

=> In practice aim for a combination of GPLs and DSLs

Closing remarks

Model-Based Development

Model-Based Development

- **Models-based development uses all four techniques for dealing with complexity:**
  - Abstraction: Identify high-level concepts that hide low-level details
  - Boundedness: Impose acceptable restrictions on the considered problem space
  - Composition: Divide one problem into multiple independent smaller problems
  - Duplication: Use multiple overlapping approaches for the same problem
- **General modeling goals:**
  - Speeding up software development of large complex systems
    - Human understanding
    - Early validation
    - Code generation
    - Automated testing
  - Bridging the gap between application domain expertise and technical system realization
- **Notes:**
  - Modeling is for a specific purpose; there exist many different types of models
  - Modeling often helps you to detect important unclarities

Some other techniques

- **Control of continuous-time physical processes**
  - Simulation, Analysis, Coding, Verification
  - Some tools:
    - MATLAB Simulink
- **Low-code/No-code**
  - Related to horizontal DSLs
  - Some tools:
    - Mendix
- **Model Based Systems Engineering**
  - Collaboration and traceability across multiple related diagram types
  - Some tools:
    - Capella
    - Cameo Systems Modeler

Objectives

At the end of the course, you should be able to:

- Explain some complexity challenges of software-intensive high-tech systems
- Explain 4 techniques for dealing with complexity
- Explain the purpose of Model-Based Development
- Compare Model-Based Development with other techniques

Assessment:

- Modeling assignments for 3 modeling techniques (in groups of 2 students)
- Reflection document on Model-Based Development (individual)

Reflection document

Contents:

- **Formulate your informed view on Model-Based Development for Software Systems**
- **Motivate this view based on your experiences in this course**
  - (Optional) You may relate it to other (properly-referenced) experience/information sources
  - (Optional) You may relate it to your prior software development experiences

Grading criteria:

- **Showing understanding of model-based development for software systems**
- **Providing an overarching view with supporting arguments (including your experiences in this course)**
- **Referencing all used sources (facts, experiences, etc.) in an appropriate way**

Note:

- **Individual assignment, to be submitted as PDF**
- **Length: 1-2 pages A4 (= 500-1000 words)**