UMLEmb: UML for Embedded Systems
I. Introduction

Ludovic Apvrille, ludovic.apvrille@telecom-paris.fr

LabSoC, Sophia-Antipolis, France
Goals

System specification (includes software specification)
Goals (Cont.)

- To propose a method, a language, and a tool, that can be applied to the design of a broad variety of systems
  - Real-time and embedded systems
  - Transportation systems, smart objects, . . .
- To practice modeling using a UML/SysML framework
- To answer your questions
- To interact together e.g. be able to evaluate the model of someone else
  - And be able to evaluate your own work!
Origin of this Course

- This course was designed with Prof. Pierre de Saqui-Sannes, ISAE Sup’Aero
- It has been used worldwide for years in different formats for:
  - Master students
  - Tutorials in international conferences
  - Trainings in companies
Outline of the Lectures and Labs

From a system specification, you will learn how to:

- Capture system requirements
- Analyze the system
- Design the system
- Validate the system

All stages will be explained with UML/SysML models

BTW: Do you know what is a system specification?
Lecture Organization

Lectures: ~4 sessions
- Presentation of SysML diagrams
- Exercises

Labs: ~3 sessions
- Modeling a system with TTool
  - Requirements, analysis, design, validation

Grading policy (Applies only to Eurecom students)
- 30% on labs. Attendance is therefore obligatory.
- 70% on exam. (Exam is like a lab).
Recommended Books

(Also available on the course’s website)

- F. Kordon et al, ”Embedded systems : Analysis and modeling with SysML, UML and AADL”
- D. Alan et al, ”Systems analysis and design with UML version 2.0: an object-oriented approach”
- L. Doldi, ”UML 2 illustrated - Developing real-time and communications systems”
- See the ”link” section of UMLEmb website for videos of examples on how to model with similar approaches
Outline

Introduction to modeling

OMG, UML and SysML

UML/SysML for Embedded Systems
Outline

Introduction to modeling

OMG, UML and SysML

UML/SysML for Embedded Systems
Designing Embedded Systems

How to Handle Complexity?
Modeling and verification!
(But there are other options)
Modeling is not Really a New Technique...

...and it is not limited to Software!
Modeling is not Really a New Technique...

"If you fail to plan, you are planning to fail!"

Painting by Duplessis.
Abstraction Level

HOW TO CREATE A STABLE DATA MODEL

(source: Geek and Poke, 2013)
So, What is Modeling?

A modeling = An abstraction of the system to design
- Representation of the main functionalities of a complex system
- Non relevant details are ignored

Abstractions make it possible to deal with complexity
An engineer, or a development team, cannot have a global understanding of complex systems

A modeling is a view of a system according to some assumptions
Software Development Techniques for E. S.

**Code-based approaches**

- **Extreme Programming**
  - Strongly tested step-by-step code increments

- **Agile Software Development**
  - Focus on change in specification

**Model-based approaches**

- **V-Cycle**
  - KAOS, AADL, MDE, ...

- **Formal models**
  - B, LOTOS, Petri nets, ...

```
chout(m1)
m.data = secretData
m1 = sencrypt(m, sk)
```
Outline

1. Introduction to modeling
2. OMG, UML and SysML
3. UML/SysML for Embedded Systems
What is UML?

**UML = Unified Modeling Language**

**Main characteristics of UML**

- Standard graphical modeling language for complex systems
  - Defined by OMG
- Specification, design, automatic code generation, documentation
- Independent of any programming language
- Object-oriented design
- Supported by many CASE Tools
  - CASE = Computer-Aided Software Engineering
- **But**: No standard UML methodology
Origin of UML

- Booch
- Rumbaugh
- Jacobson
- Odell
- Meyer
- Harel
- Shlaer-Mellor
- Gamma et al.
- Wirfs-Brock
- Embly
- Fusion

UML

Classification
Object life cycles
Frameworks, patterns, notes
Singleton classes
Operation descriptions
Message numbering
Pre- and post-conditions
State charts
Responsibilities
Genesis of UML

- **ROOM** (1994)
- **MSC & SDL** (1993)
- **Statecharts** (1980s)
- **OMT** (Rumbauch et al.)
- **Booch**
- **OOSE** (Jacobson et al.)
- **UML 0.9** (1996)
- **UML 1.1** (1997)
- **UML 1.5** (2003)
- **UML 2.0** (2005)
- **UML 2.5** (2015)
UML 1.5

Requirement capture
Outside the UML model

Use case driven analysis
- Use case = main function
- System / environment (actors)
- Use cases need documentation

Object-oriented design
- Object = Name + Attributes (state) + Methods
- Objects communicate using method calls
- Class diagrams for software architectures

Customer
ATM
DisplayBalance
DeliverBankNotes

BankAccount
- balance
+ balance()
+ credit(amount)
UML 2

Requirement capture
Outside the UML model

Analysis
- Enhanced sequence and activity diagrams

Object-oriented design
- Object can communicate
  - by method calls
  - via ports
    - Input and output signals are defined by interfaces
OMG: Object Management Group

- Non-profit organization
- Goal: definition of standards related to object-oriented services
  - MOF, UML, XMI, CWM, CORBA (includes IDL, IIOP)
- 11 creating members
  - Hewlett-Packard, IBM, Sun Microsystems, Apple Computer, American Airlines, Data General, ...
- Nowadays: ~300 members
  - https://www.omg.org/cgi-bin/apps/membersearch.pl
Outline

Introduction to modeling

OMG, UML and SysML

UML/SysML for Embedded Systems
UML for Embedded Systems

Specificity of embedded systems

- Strict constraints
  - Performance constraints, real-time constraints, limited resources, etc.

→ Specific UML operators, diagrams, methodologies, toolkits

- Make use of some UML diagrams rather than others
- Make use of simulation techniques as soon as possible in the development cycle
- Specific UML toolkits
- Profiles
UML Profiles

**Definition**
- UML defines extension mechanisms to e.g.,
  - Define new operators
  - Provide a semantics
  - Give a methodology

**Example of profiles**
- Profiles defined by OMG (e.g., SPT, MARTE, SysML)
- Profiles defined by tool vendors (e.g., in Rhapsody, Artisan)
- User-defined and company-defined profiles
From UML to SysML

What’s wrong with UML? (as far as system modeling is concerned)

- Objects are for computer-literate, not for systems engineers
- Requirements are described outside the model using, e.g., IBM DOORS
- Allocation relations are not explicitly supported

Nevertheless SysML is a UML 2 profile

- Developed by the Object Management Group (OMG) and the International Council on Systems Engineering (INCOSE)

SysML standard: www.omgsysml.org
SysML

- **An international standard** at OMG
  - UML profile

- **A graphical modelling language** that supports the specification, analysis, design, verification, and validation of systems that include hardware, software, data, staff, procedures, and facilities

- **A notation**, not a method

- **Proprietary tools**
  - Enterprise Architect, Rhapsody, Modelio, ...

- **Free software tools**
  - Polarsys, Papyrus, TTool, ...

- **User communities**
  - [http://sysmlfrance.blogspot.com/](http://sysmlfrance.blogspot.com/)
  - [http://sysmlbrasil.blogspot.fr/p/sysml-brasil.html](http://sysmlbrasil.blogspot.fr/p/sysml-brasil.html)
SysML Diagrams vs. UML Diagrams

- **SysML Diagram**
  - **Behavior Diagram**
  - **Requirement Diagram**
  - **Structure Diagram**

- **Activity Diagram**
- **Sequence Diagram**
- **State Machine Diagram**
- **Use Case Diagram**
- **Block Definition Diagram**
- **Internal Block Diagram**
- **Package Diagram**

- Same as UML 2
- Modified from UML 2
- New diagram type
From SysML to AVATAR

**AVATAR reuses most SysML diagrams**
- Requirement capture: requirement diagrams
- Analysis: use case, sequence and activity diagrams
- Design: block and state machines diagrams

**AVATAR does not entirely comply with the OMG-based SysML**
- In AVATAR, block diagrams merge block and internal block diagrams
- AVATAR does not support continuous flows

**AVATAR gives a formal semantics to several diagrams, including:**
- Block instance and state machine diagrams
  - Starting point for simulation, verification and code generation
TTool: A Multi Profile Platform

**TTool**
- Open-source and free toolkit mainly developed by Telecom ParisTech
- Multi-profile toolkit
  - DIPLODOCUS, AVATAR, ...
- Support from academic (e.g. LIP6, ISAE) and industrial partners (e.g., Nokia)

**Main ideas**
- Lightweight, easy-to-use toolkit
- Simulation with model animation
- Formal proof at the push of a button
So, what’s next?

1. **Modeling in SysML/AVATAR**
   - Methodology
   - Diagrams

2. **Validation**
   - Simulation
   - Formal verification
   - Code generation, and execution of that code