I. Introduction

- Introduction to modeling
- OMG
- UML 2.0
- UML for embedded systems
I. Introduction

- Introduction to modeling
- OMG
- UML 2.0
- UML for embedded systems

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
Who Uses Modeling?

- Architects
- Tailors
- Statisticians
- Engineers
  - Mechanics, Mechanics of fluid,
  - Protocols,
  - Electronic, microelectronic
- **No exception for software!**
  - And more particularly, for embedded systems

What is UML?

- **UML = Unified Modeling Language**
- **Main characteristics of UML**
  - Graphical modeling language for complex systems
    - Specification, design, automatic code generation, documentation
    - Independent of any programming language
  - Object-oriented design
  - Supported by many CASE Tools
    - CASE = Computer-Aided Software Engineering
  - Warning: no standard UML methodology
Why Use UML for Modeling?

- **Standard notation**
  - Known by a growing number of people
  - Supported by matured tools

- **Best understanding of systems by**
  - Clients, experts of the domain, designers, programmers

- **Support of engineering work**
  - Abstract view of the system
  - Life cycle
    - Focused on first steps: requirement analysis, design
    - Simulation, automatic generation of code (C, Java, C++, etc.)
    - Documentation, maintenance, revision
  - Reuse

Gathering on UML

- **UML gathers best practices of software engineering**

- **Modeling of complex (and software-based) systems**

- **OMG (Object Management Group) standard**
  - The reference
  - [http://www.uml.org](http://www.uml.org)

- **14 diagrams for expressing complementary point of views**
Gathering on UML (Cont.)

- **A notation**
  - Semantics
    - Metamodel
    - No formal semantics
  - No methodology
    - Process suggested by UML tool dealers
      - Unified process
        » RUP – Rational Unified Process
  - Extension capabilities
    - Profiles

UML Views and Diagrams

- **A view describes a static or a dynamic aspect of the system**
- **For each view**
  - Several diagrams are available
  - Example: interactions between objects
    - Sequence diagrams
    - Collaboration diagrams
- **Components of views**
  - Classes, ports, interfaces, actors, messages, etc.
- **Mechanisms for extending diagrams**
  - Stereotypes, notes, constraints
Towards a Federative Notation Used by Industrial Practitioners?

- MERISE for project management, information systems
- GRAFCET for automatics
- SDL for telecommunications
- VHDL for integrated circuits
- SAO for air and space

I. Introduction

- Introduction to modeling
- OMG
- UML 2.0
- UML for embedded systems
### Origin of UML

- **Booch**
- **Rumbaugh**
- **Jacobson**
- **Meyer** (Pre- and post-conditions)
- **Shlaer-Mellor** (Object life cycles)
- **Harel** (State charts)
- **Gamma et al.** (Frameworks, patterns, notes)
- **Embly** (Singleton classes)
- **Wirfs-Brock** (Responsibilities)
- **Fusion** (Operation descriptions, Message numbering)

### The OMG

- **Object Management Group**
- **Non-profit organization**
- **Goal:** definition of standards related to object-oriented services
  - MOF, UML, XMI, CWM, CORBA (includes IDL, IIOP)
- **Chronology**
  - 1989: 11 creating members
    - Hewlett-Packard, IBM, Sun Microsystems, Apple Computer, American Airlines, Data General, ...
  - Nowadays, more than 800 members
  - Members have more or less important vote weight
Development of a Norm at the OMG

- A Request for Proposal (RFP) asks for submissions
  - A RFP must be written, voted and released
  - Contains requirements about its release, an evaluation guide and provisional dates
- External companies work on submissions
  - Companies gather in consortiums
  - Each company answering to a RFP must produce a Letter of Intent (LOI) in the right period of time
    - The company agrees to support the new norm with tools in the year following the release if its submission is accepted
    - Companies have to follow initial and final submission dates
- The norm is voted
  - Different “voting bodies” must approve submissions, and make a selection of some of them, if necessary
  - When a submission is voted, a Finalization Task Force is settled to address the defaults of that submission
  - The FTF have no right to make substantial modifications over a voted submission
- The norm is released
  - When the FTF has completed its work, the norm becomes a “standard released”
  - An RTF (Release Task Force) is settled to manage the default of that norm

I. Introduction

- Introduction to modeling
- OMG
- UML 2.0
- UML for embedded systems
Genesis of UML 2.3

- OMT (Rumbaugh et al.)
- Booch
- OOSE (Jacobson et al.)
- UML 0.9
- UML 1.1
- UML 1.5
- UML 2.0
- UML 2.3

1996

MSC & SDL

ROOM

Statecharts

Statecharts (C) Ludovic Apvrille

UML 2.0
August 2003

UML 2.3
May 2010

Nov. 1997

2002

UML 2.x: Diagrams

1. Class diagram
2. Use case diagram
3. Object diagram
4. State machine diagram
5. Activity diagram
6. Sequence diagram
7. Communication diagram
8. Component diagram
9. Deployment diagram
10. Composite structure diagram
11. Interaction overview diagram
12. Timing diagram
13. Package diagram
14. Profile diagram

(C) Ludovic Apvrille

UML for Embedded Systems - Fall 2012

slide 17

slide 18
Overview

I. Introduction

- Introduction to modeling
- OMG
- UML 2.0
- UML for embedded systems
UML for Embedded Systems

**Specificity of embedded systems**
- Strict constraints
  - Performance constraints, real-time constraints, etc.
  - Critical aspect
- Limited resources

**Specific UML methodology**
- Make use of some UML diagrams rather than others
- Make use of simulation techniques as soon as possible in the development cycle
  - Critical systems

**Specific UML toolkits**
- Profiles

Methodologies for Embedded Systems

- Each toolkit vendor proposes its own methodology
- For this course, we use a methodology which
  - Is simplified
  - Shares many aspects to proposed methodologies
  - Is inspired from books, papers and concrete case studies
A UML Methodology Focused on Embedded Systems

- **Detailed design**
  - Behavior of the system

- **Validation of the system**
  - Simulation
  - Code generation

- **Design**
  - Classes of the system
  - Architecture of the system

- **Analysis**
  - Use case
  - First class diagram
  - Relevant scenarios

---

**Stages**

- **Analysis**
  - Analysis of the requirements of the system

- **Design**
  - Structure the system under the form of classes and relations among those classes

- **Detailed design**
  - Describe the behavior of the system

- **Validation**
  - Check that the behavior of the system corresponds to the targeted one
UML Diagrams for Embedded Systems

1. Class diagram
2. Use Case diagram
3. Object diagram
4. State machine diagram
5. Activity diagram
6. Sequence diagram
7. Communication diagram
8. Component diagram
9. Deployment diagram
10. Composite structure diagram
11. Interaction overview diagram
12. Timing diagram
13. Package diagram
14. Profile diagram

UML Toolkits for Embedded Systems

- Goal: edition of diagrams, animation, code generation
- IBM Rational Rhapsody
- ARTiSAN Real-Time Studio (Artisan Software)
Books


Outline of the Course

- Analysis with UML of an embedded system
  - Objects
    - Modeling of objects / classes in UML
- Design with UML of an embedded system
- Detailed design with UML of an embedded system
- Validation with UML of an embedded system
- Exercises