

The Unified Modeling Language

Introduction and overview

Introduction

What is UML ? (1)

- UML is the industry standard for specifying, visualising and documenting complex object-oriented software systems.
- It is the descendant of a number of OO methodologies and notations, "unified" by the Object Management Group (OMG) in the 1990's.

What is UML ? (2)

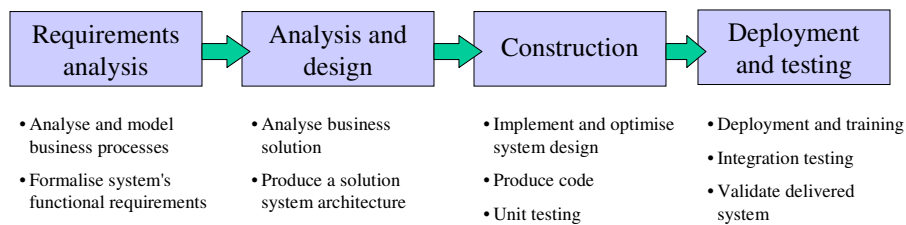
- An ensemble of graphical syntaxes for describing the different aspects of modern systems
 - data structure, control flow, algorithms, physical architecture, ...
- A meta-model which formally and unambiguously defines the meaning of the graphical notations
 - improves communications between humans
 - makes the development of automated tools possible

Why do we need UML ?

Modern software design is difficult!

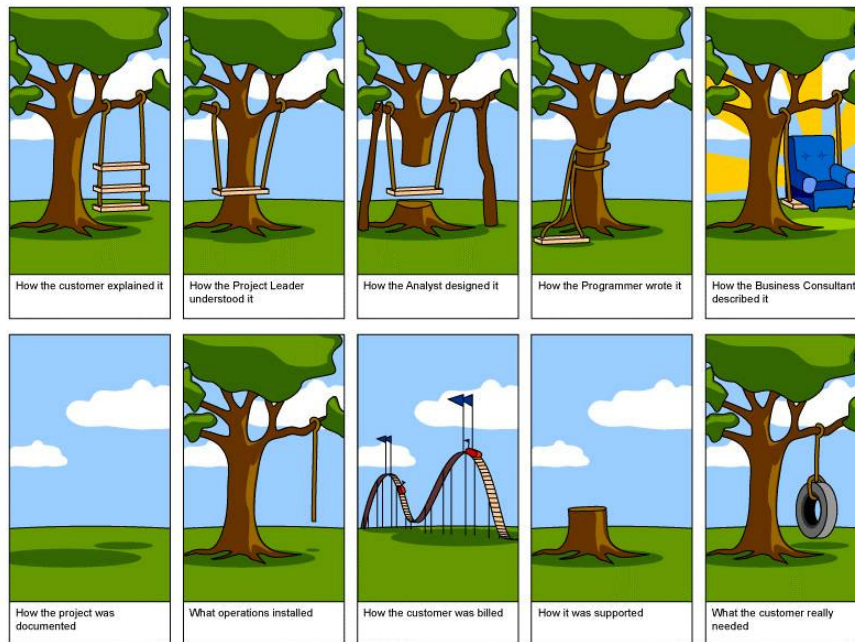
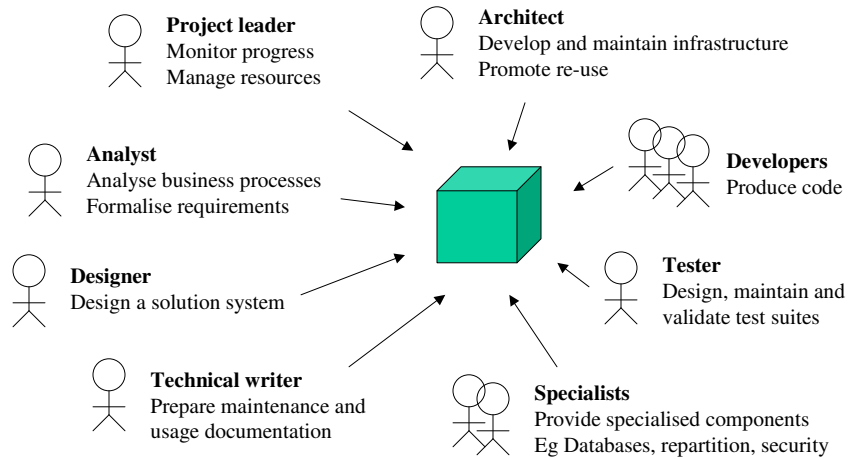
- Applications are increasingly complex
- They take a long time to produce
- Increasing specialisation means a single project requires many contributors
- No software is created in a vacuum
 - Must integrate with complex existing systems

Project lifecycle

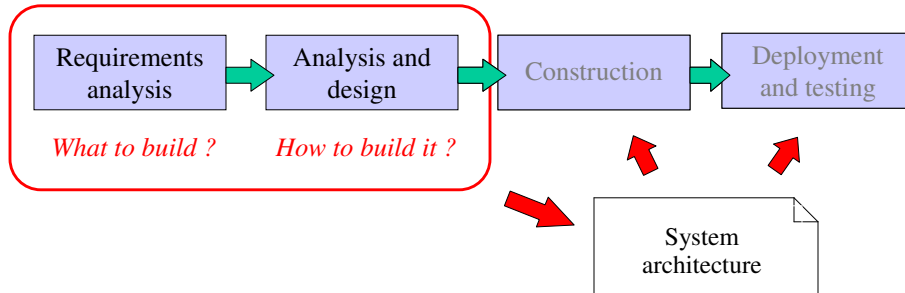


- Depending on application size, can take from a few months to a few years
- Construction is the biggest job, consuming roughly 2/3 of time and resources

Project participants



System design



- System design produces a system architecture that is used as a basis for construction and testing

The importance of good system design

Good design is all about reducing risk and saving time "downstream" in the project lifecycle

- Ensures that you build the right software, and well.
- Facilitates parallel construction which reduces linear development time.
- Identifies opportunities for re-use of components and best practices (design patterns).

Why use UML for system design? (1)

- UML is based on the object model, proven to be the most effective paradigm for system design today.
- UML provides 9 different notations allowing us to express all aspects of an OO system design.
- Being graphical, these are more abstract easier to understand than text or code.
- UML provides a single, unified language for more effective communication.

Why use UML for system design? (2)

- Supports incomplete models and different levels of abstraction, I.e. over entire product lifecycle.
- International standardisation means that everyone has access to UML (for free), and can interpret it.
- Formal semantics mean that CASE tools can automate many design tasks.
- UML is extensible, allowing it to be adapted to new or highly specialised needs.

Origins of UML

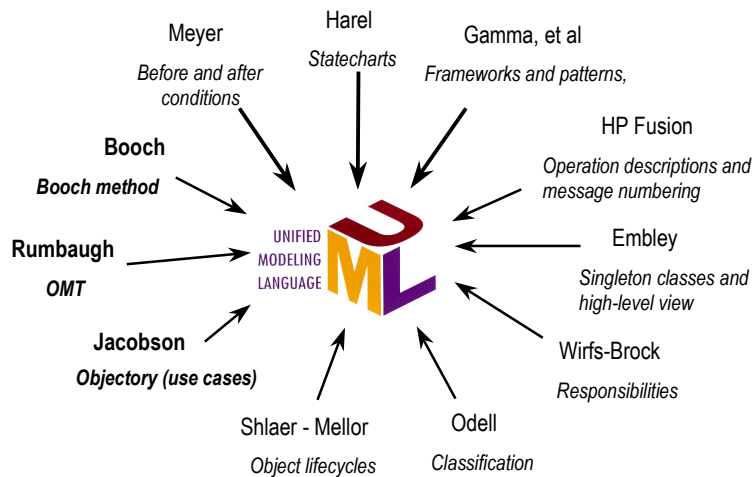
1994 : "Tower of Babel"

- More than 50 object-oriented methods!!
Fusion, Shlaer-Mellor, ROOM, Class-Relation, Wirfs-Brock, Coad-Yourdon, MOSES, Syntropy, BOOM, OOSD, OSA, BON, Catalysis, COMMA, HOOD, Ooram, DOORS
...
- «Meta-models» resemble each other
- Graphical notations differ
- Processes differ or remain vague
- *But industry needs standards!*

Birth of UML

- In 1994 OMG called for proposals for a single, standardised OO method
- Accepted the proposal of *Booch, Rumbaugh, Jacobson*, the "three amigos"
- Version 1.0 released in 1997
- Version 1.5 is most widely used at the moment
- Version 2.0 released 2004/2005

UML is ... unified



Object Management Group

- Non-profit organization founded in 1989.
- Usually known for CORBA (IDL, IIOP ...)
- Consortium of over 700 companies and organisations
 - Borland, IBM, Boeing, HP, Lockheed, Thales, Unisys, INRIA, ...
- But not controlled by any one company
- On the web : <http://www.omg.org>

General goals of OMG for UML

- Facilitate modeling of systems using OO concepts
- Be readable by both humans and machines
- Provide multiple views to better represent the different aspects of a system
- Be extensible to new problems and domains
- Model different types of systems
 - traditional software, network protocols, embedded systems, real-time systems, distributed systems, web applications, business processes, development processes, UML itself, ...

State of UML today

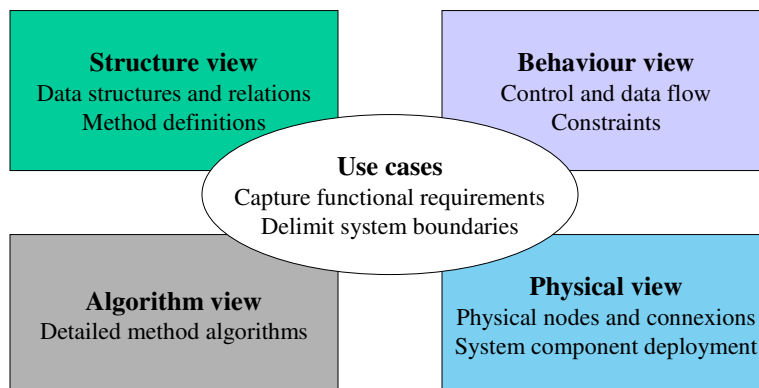
- The industry standard for describing large, complex systems
 - web applications, traditional software, embedded real-time systems, communication protocols, medical systems, ...
- Natural choice for programmers in object-oriented languages
 - Java/ J2EE, C++, C#, PHP5.0, .net, ...
- Supported by a number of sophisticated CASE tools
 - Visual Paradigm, Rational Suite (IBM), Together (Borland)
- OMG is an active and open forum constantly improving and extending the standard
 - includes major industrial and academic players

How UML describes a system design

"4+1 View" Architecture

- Unifies Requirements Analysis and Analysis and Design
- System requirements are described by the Use case notation
- System architecture is described by 4 views which explicitly realise the use cases.
- Each of the four views focuses on a different aspect of a system design.

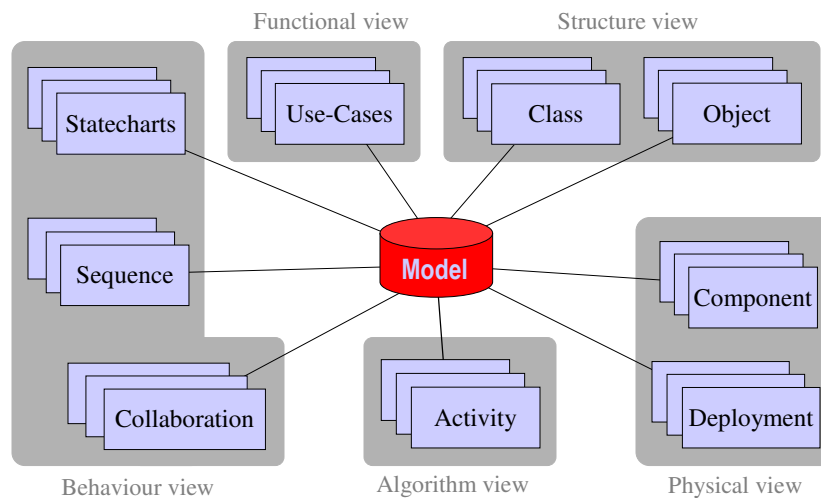
"4+1 View" Architecture



9 diagrams

- UML defines 9 standard notations for describing the 4+1 views.
- A UML diagram is not a model in itself, but the **projection** of a model from a certain point of view.
- A diagram is meant to **highlight** specific aspects of the system.
- All projections of the same model must be **coherent**.
- While each diagram has a primary use, they can be used in other views.

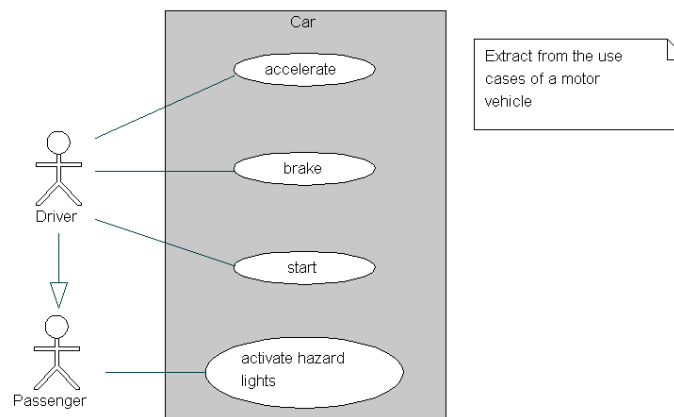
9 diagrams



Use cases

- *What to build*
- Definition of a set of functional requirements which formally document:
 - who (or what) will use the system
 - how this interaction will take place
 - data flows and their presentation
- Delimits the extent of the system to be built
- Elaborated in collaboration with client
 - meetings with end users, studying business processes, studying existing interfacing systems, ...

Use case diagram



Use case description

UC 1 : Accelerate

Actor : Driver
Objective : Make the car go faster
Constraints : Speed may not exceed a governor-imposed limit of 130kph

Nominal scenario

1. Driver presses acceleration pedal with foot.
2. The car increases speed.

Alternative scenario A : The car is already at the maximum speed

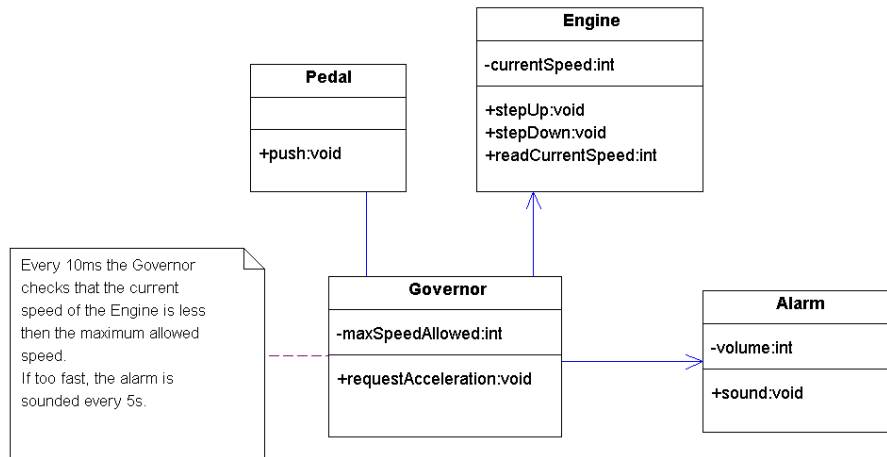
1. Driver presses acceleration pedal with foot.
2. The car responds does not change speed.
3. Resistance is returned to the driver through the pedal.
4. A warning alarm sounds.

Alternative scenario B : ...

Structure view

- *How to build it : Structure*
- Describes the static data structures of the system.
- A system is broken down into components using the principles of decomposition and delegation.
- Each component represents a business entity with a well defined responsibility (data + behaviour).
- These components are designed to work together to realise the use cases.
 - This collaboration is described in the behaviour view

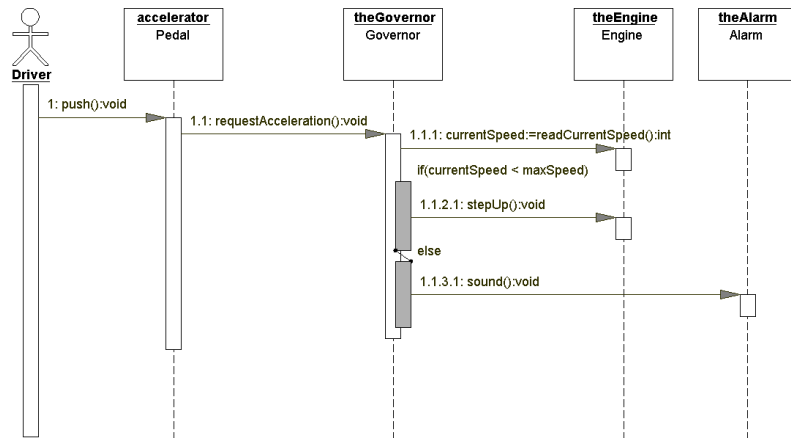
Class diagram



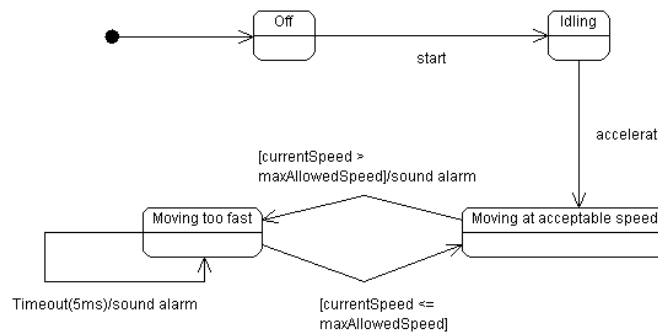
Behaviour view

- *How to build it : Behaviour*
- The dynamic behaviour an OO system is described by sequences of messages sent between components.
- The message flow divides the realisation of use cases between components according to responsibility.
 - Delegation is used to render each component as light as possible.
- The reaction of a component to the reception of a message is conditioned by an internal state.

Sequence diagram



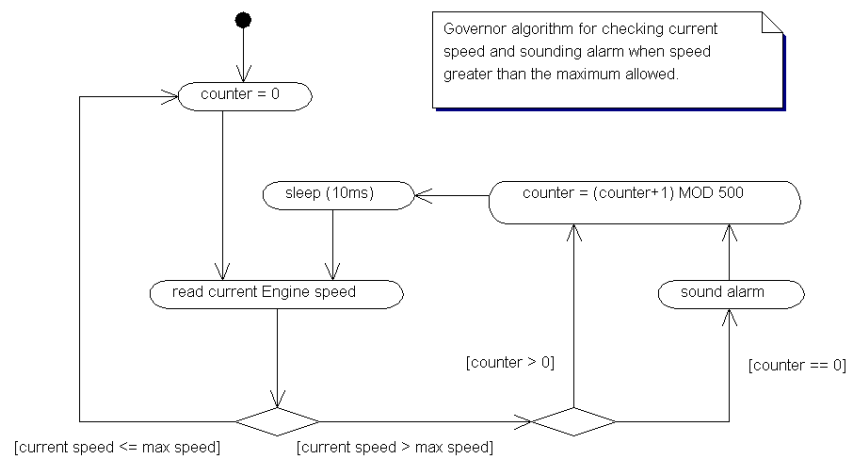
Statechart



Algorithm view

- *How to build it : Algorithms*
- Describes the algorithms used in the system with a flow chart notation.

Activity diagram



Physical view

- *How to build it : Physical structure*
- Describes the physical arrangement of system components ; and
- Their deployment in a networked infrastructure.

Conclusion

- Big systems need design to avoid construction failure.
- UML is the industry standard for expressing object-oriented system design.
- 5 views for expressing system design :
 - Use cases which describe functional requirements
 - 4 architecture views which describe realisation of the use cases
- 9 diagrams to investigate each view in greater detail, without being overwhelmed.

Still to come ...

- The 9 UML syntaxes in detail
 - The object model
- A process for using UML in a project
- Understanding and using Design Patterns

Any questions ?