Object-Oriented Technology and UML

Introduction to UML

Important concepts associated with UML

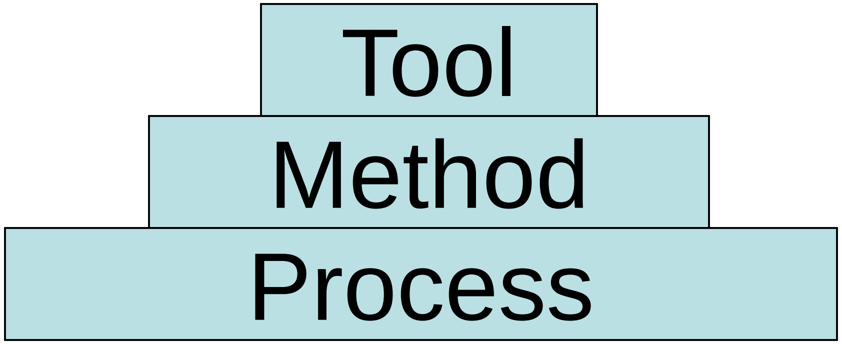
- OMG
- UML Specification
- OCL（Object Constraint Language ）
- RUP（Rational Unified Process）

Topics

- Overview of RUP(Rational Unified Process)
- Overview of UML(Unified Modeling Language)
- UML Architecture
- UML Tools

Rational Unified Process

Elements of software engineering



Tool

Method

Process

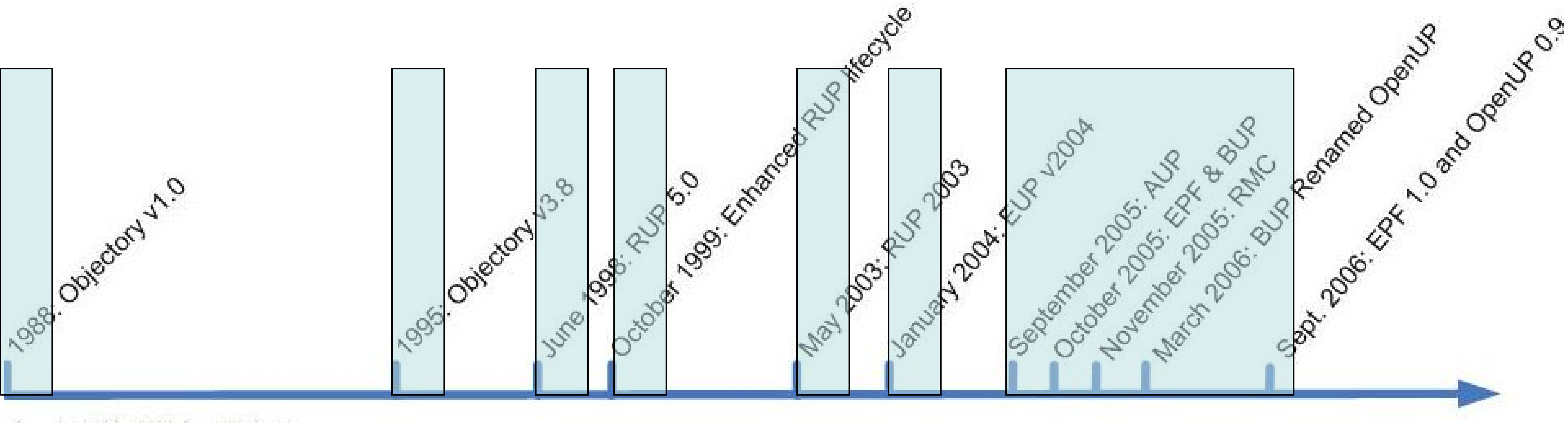
What is Software Process

- A Process defines who is doing what when and how to reach a certain goal.

What is RUP

- Rational Unified Process
  - RUP is an iterative software development process framework
  - RUP is not a single concrete prescriptive process, but rather an adaptable process framework, intended to be tailored according to requirements
  - RUP uses UML as standard language for modeling software-intensive systems

History of RUP



Six Best Practices Of RUP

- Develop iteratively
- Manage requirements
- Use components
- Model visually
- Verify quality
- Control changes

Develop iteratively

- Given today’s sophisticated software systems, it is not possible to sequentially first define the entire problem, design the entire solution, build the software and then test the product at the end
- An iterative approach is required that allows an increasing understanding of the problem through successive refinements, and to incrementally grow an effective solution over multiple iterations

Manage requirements

- The notions of use case and scenarios proscribed in the process has proven to be an excellent way to capture functional requirements and to ensure that these drive the design, implementation and testing of software, making it more likely that the final system fulfills the end user needs
- In UML, use case diagram is one of the most important digram

Use components

- The RUP supports component-based software development
- Components are non-trivial modules, subsystems that fulfill a clear function
- The Rational Unified Process provides a systematic approach to defining an architecture using new and existing components
- Component related models can be expressed by component diagram in UML

Model visually

- Visual abstractions help you
  - communicate different aspects of your software
  - see how the elements of the system fit together
  - make sure that the building blocks are consistent with your code
  - maintain consistency between a design and its implementation
  - promote unambiguous communication
- The UML is the foundation for successful visual modeling

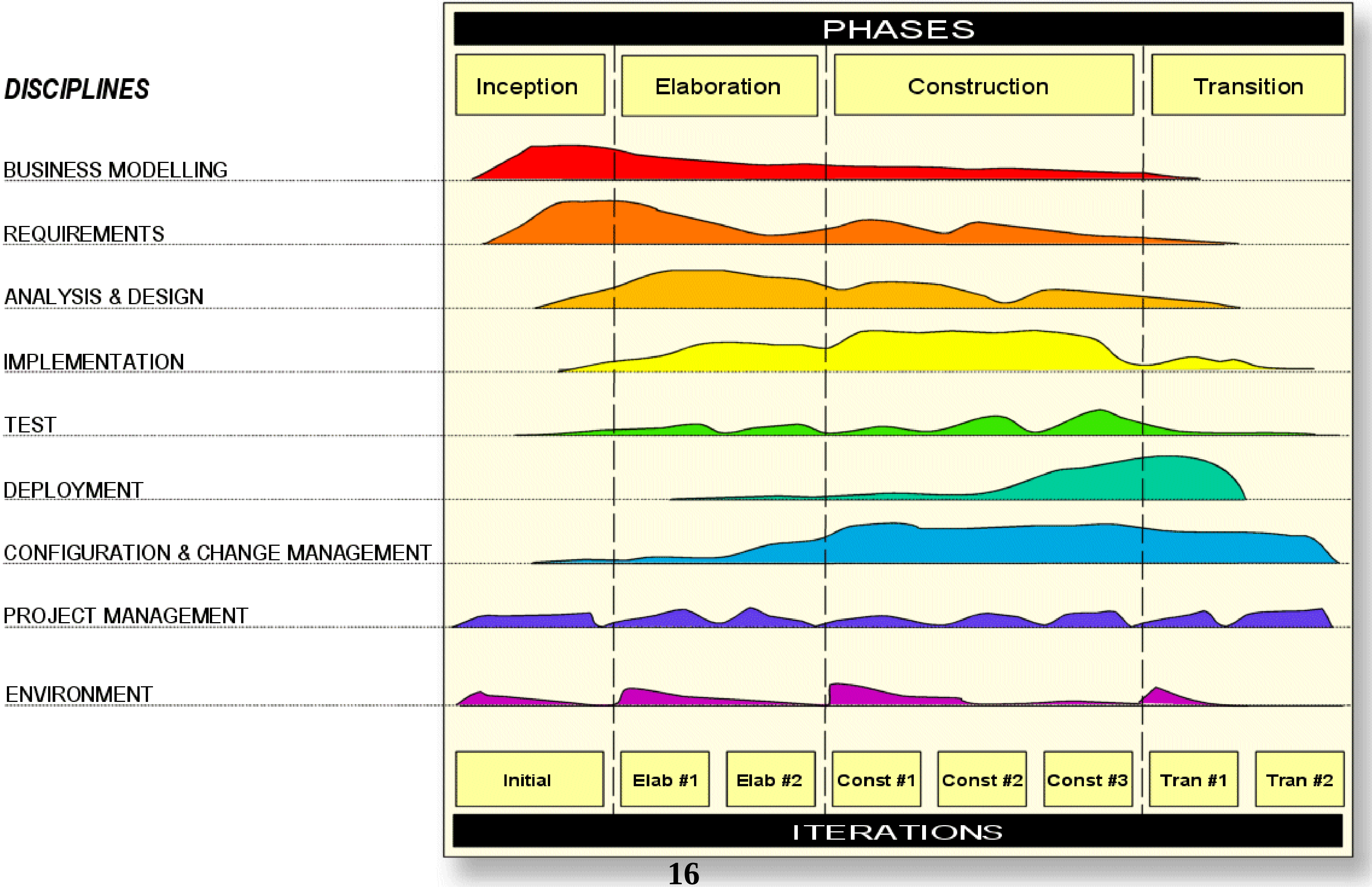
Verify quality

- Poor application performance and poor reliability are common factors which dramatically inhibit the acceptability of today’s software applications
- Quality assessment is built into the process, in all activities, involving all participants, using objective measurements and criteria, and not treated as an afterthought or a separate activity performed by a separate group

Control changes

- The process describes how to control, track and monitor changes to enable successful iterative development

RUP


DISCIPLINES
BUSINESS MODELLING
REQUIREMENTS
ANALYSIS & DESIGN
IMPLEMENTATION
TEST
DEPLOYMENT
CONFIGURATION & CHANGE MANAGEMENT
PROJECT MANAGEMENT
ENVIRONMENT
PHASES
Inception
Elaboration
Construction
Transition
Initial
Elab #1
Elab #2
Const #1
Const #2
Const #3
Tran #1
Tran #2
ITERATIONS

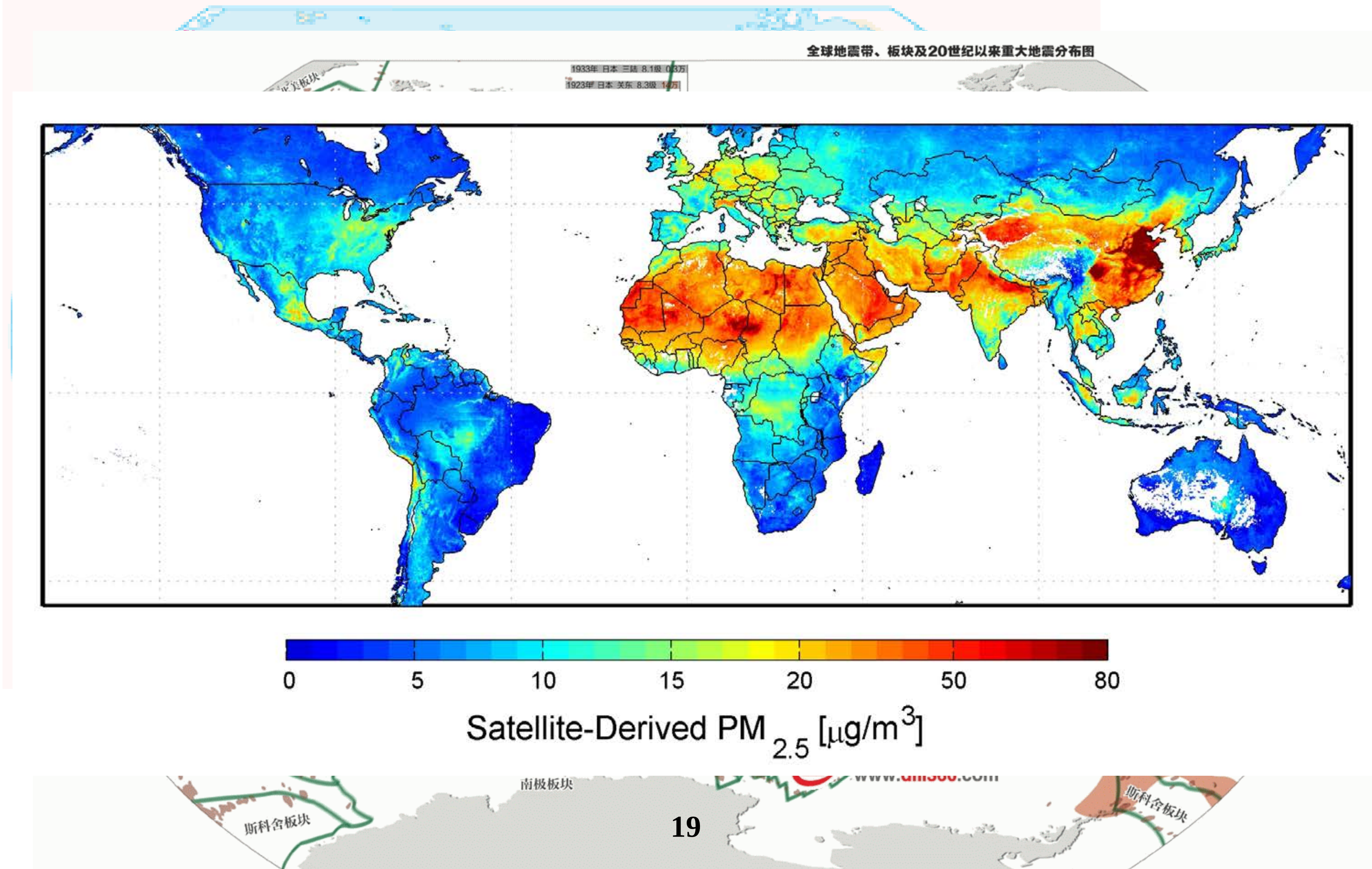
Core Workflow

- Core Process Workflows
  - Business Modeling
  - Requirements
  - Analysis & Design
  - Implementation
  - Test
  - Deployment

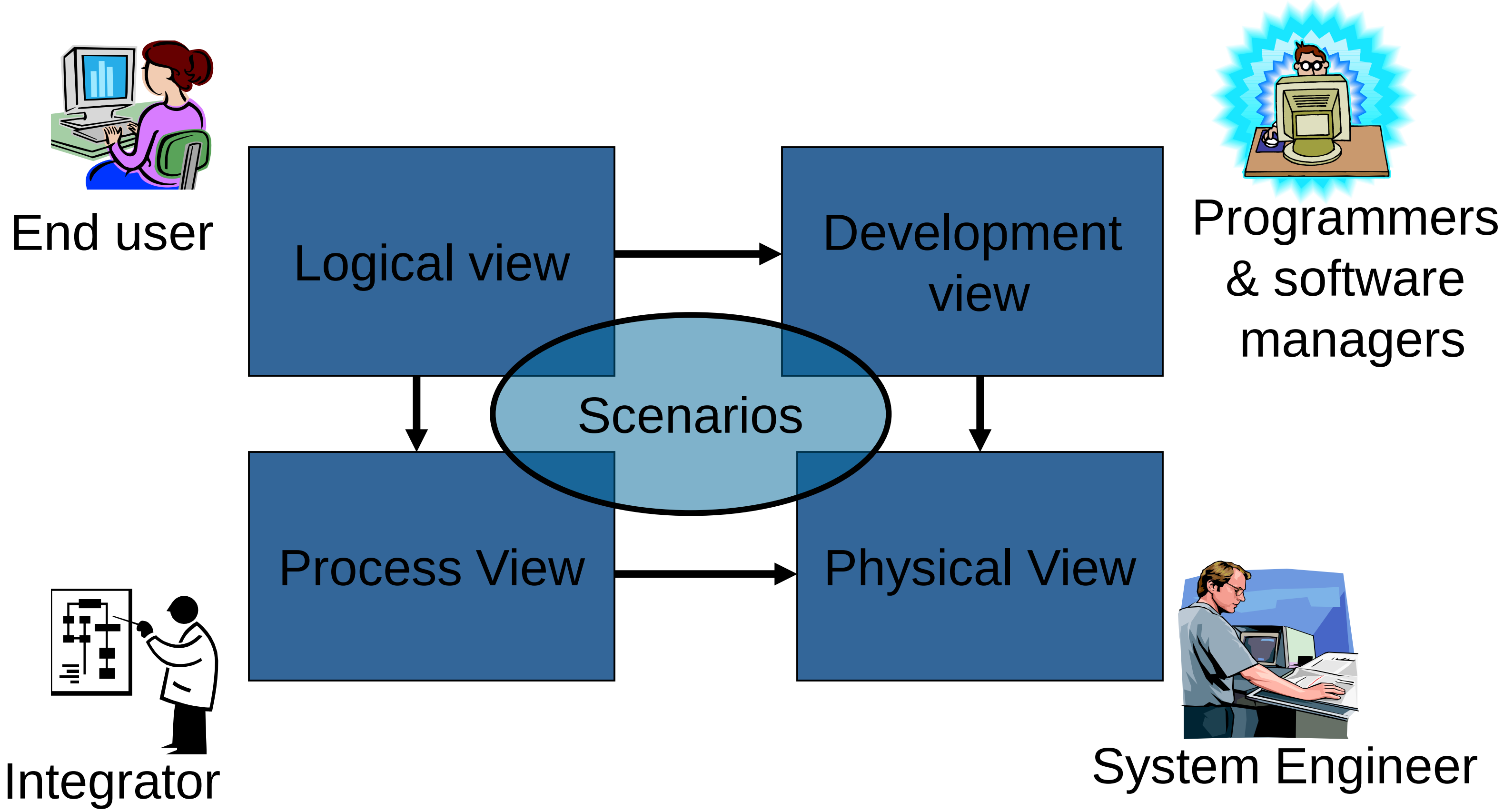
Core Workflow

- Core Supporting Workflows
  - Configuration & Change Management
  - Project Management
  - Environment

The RUP 4+1 view model


全球地震带、板块及20世纪以来重大地震分布图
1933年 日本 三陆 8.1级 0.3万
1923年 日本 关东 8.3级 14万
北美板块
0
5
10
15
20
50
80
Satellite-Derived PM $_{2.5}$ [μg/m^3]
南极板块
www.dili360.com
斯科舍板块
斯科舍板块

The RUP 4+1 view model



End user

Logical view

Development view

Programmers & software managers

Scenarios

Process View

Physical View

Integrator

System Engineer

RUP Tailoring

- The RUP framework constitutes guidance on a rich set of software engineering practices
- It is applicable to projects of different size and complexity, as well as for different development environments and domains
- This means that no single project will benefit from using of all of RUP. Applying all of RUP on a single project will likely result in an inefficient project environment
- Thus, it is recommended that all projects tailor the RUP

RUP Tailoring

- The overall approach for tailoring a process is as follows
  - Determine the needed workflows
  - Determine the input and output artifacts of each workflow
  - Determine the evolutionary plan of 4 phases
  - Determine the iteration plan of each phase
  - Determine the internal structures of workflows

RUP Summary

- Six Best Practices
- Four phases
- Nine workflows

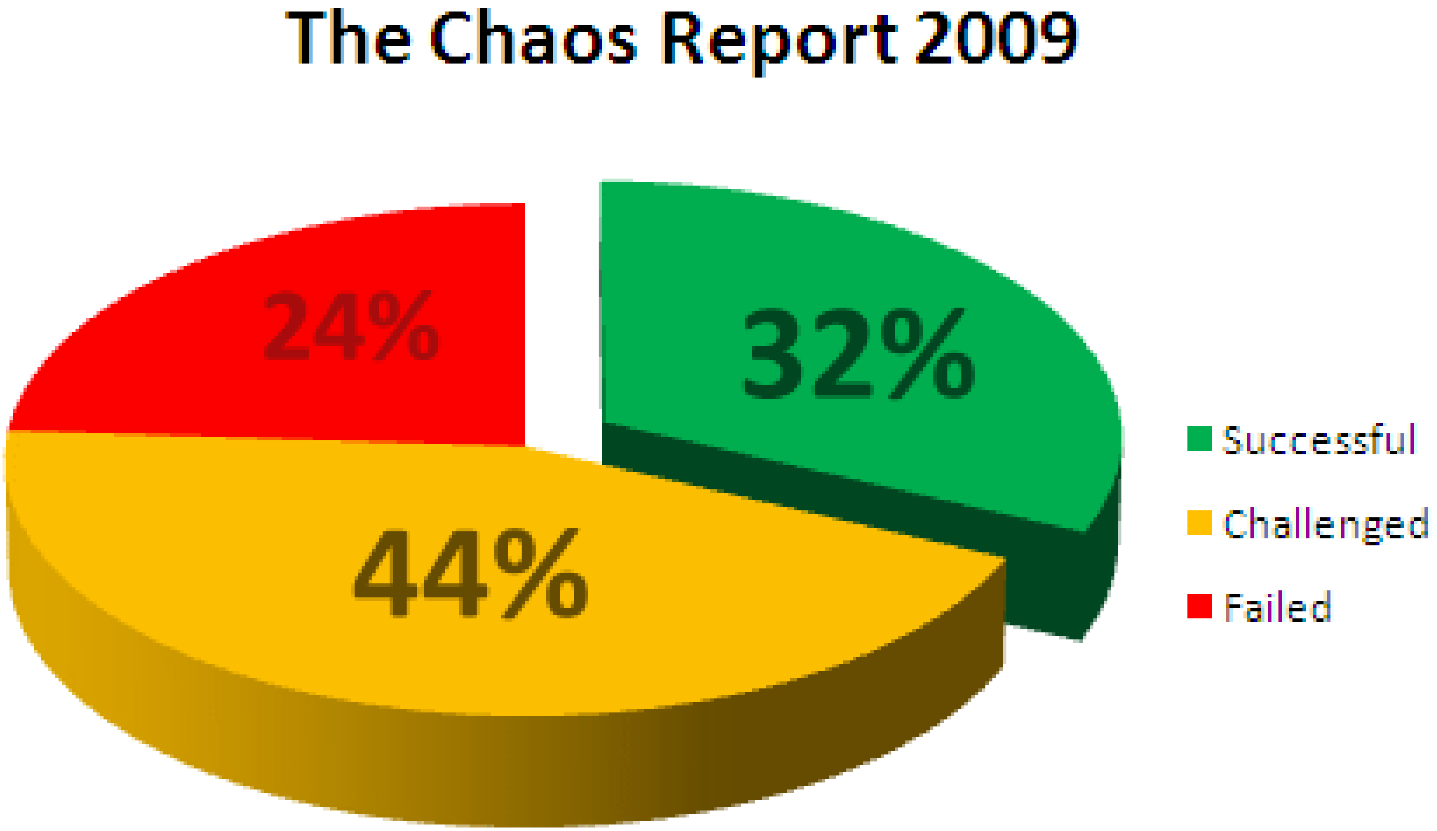
Summary

- Basic concepts of RUP
- History of RUP
- Software development life cycle of RUP
- Features of RUP

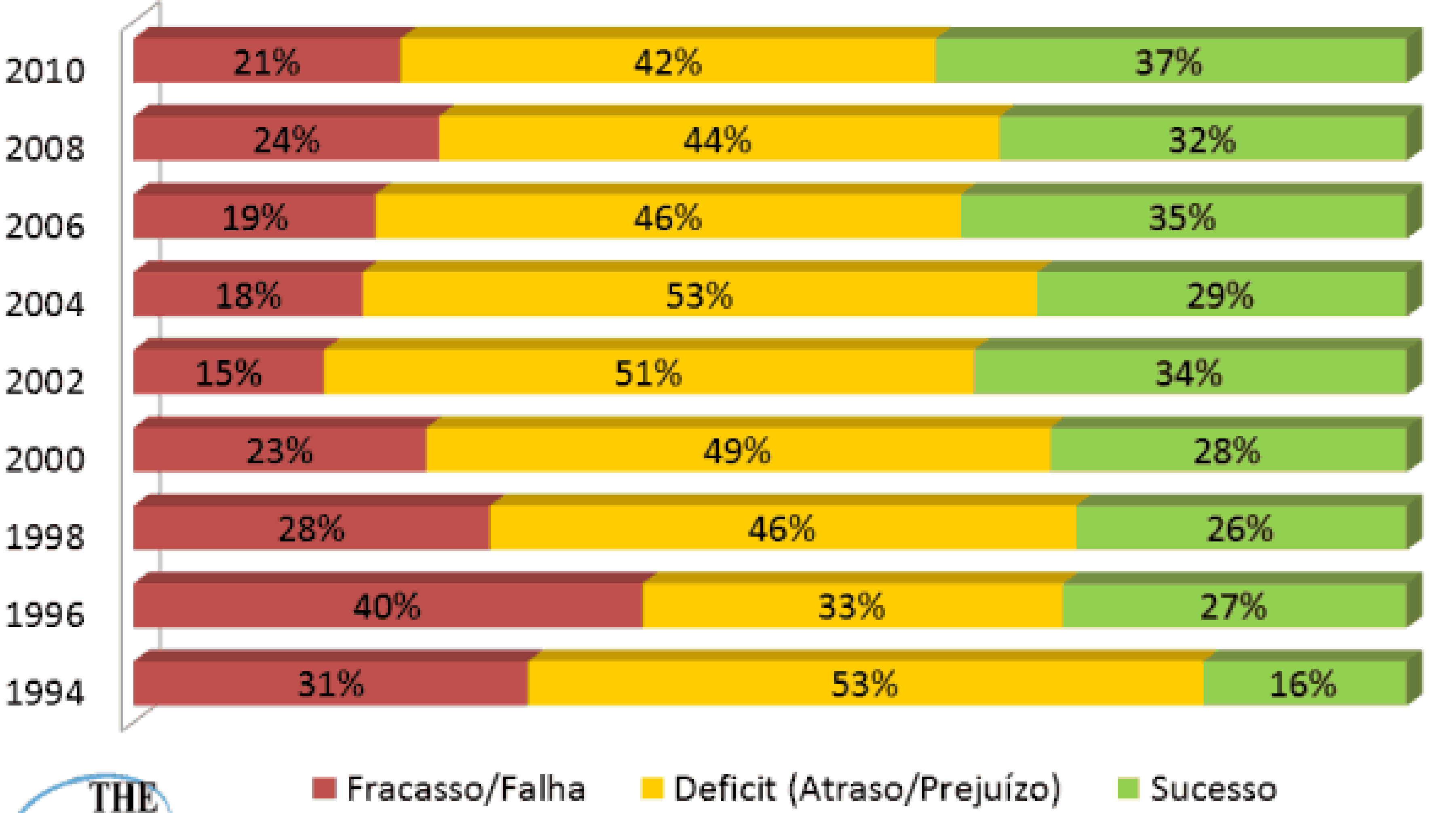
Q & A

- What is RUP? Please talk about how you understand the RUP
- What is the relationship between RUP and UML?
- Why it is necessary to tailor RUP?

Chaos Report of Standish Group



Chaos Report of Standish Group



THE STANDISH GROUP

Fonte: Standish Group; CHAOS Manifesto 2011, CHAOS Summary for 2010, Extreme CHAOS 2001.

Chaos Report of Standish Group

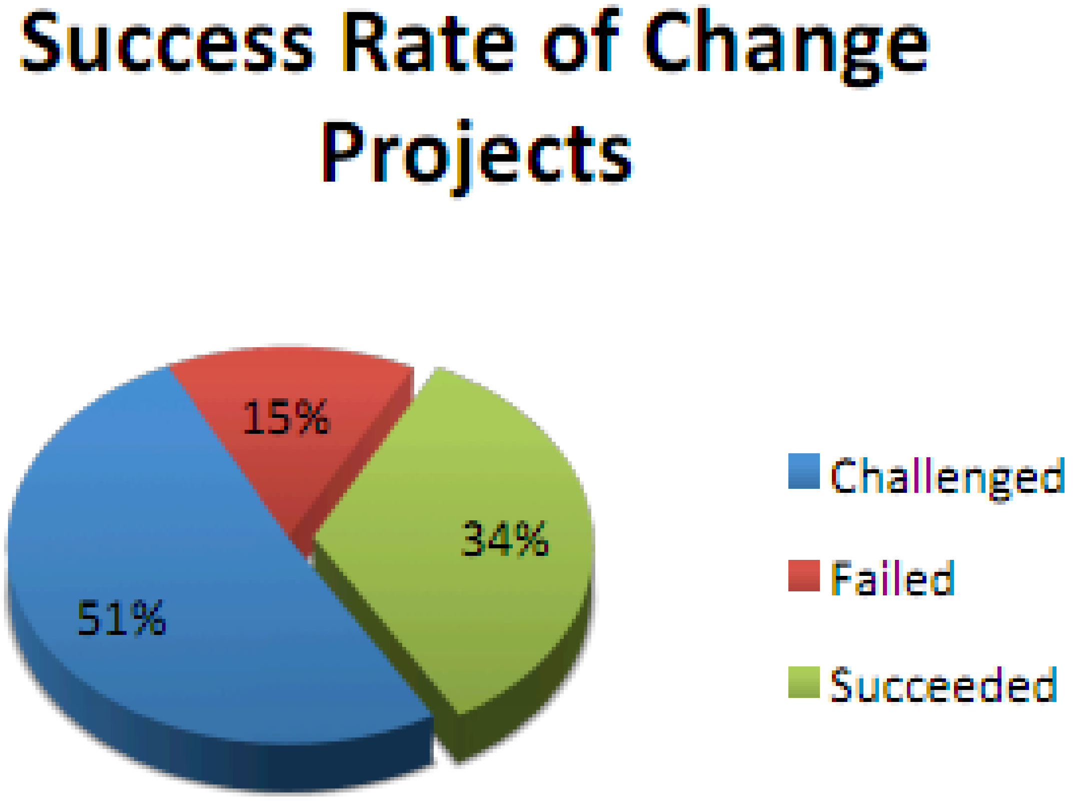
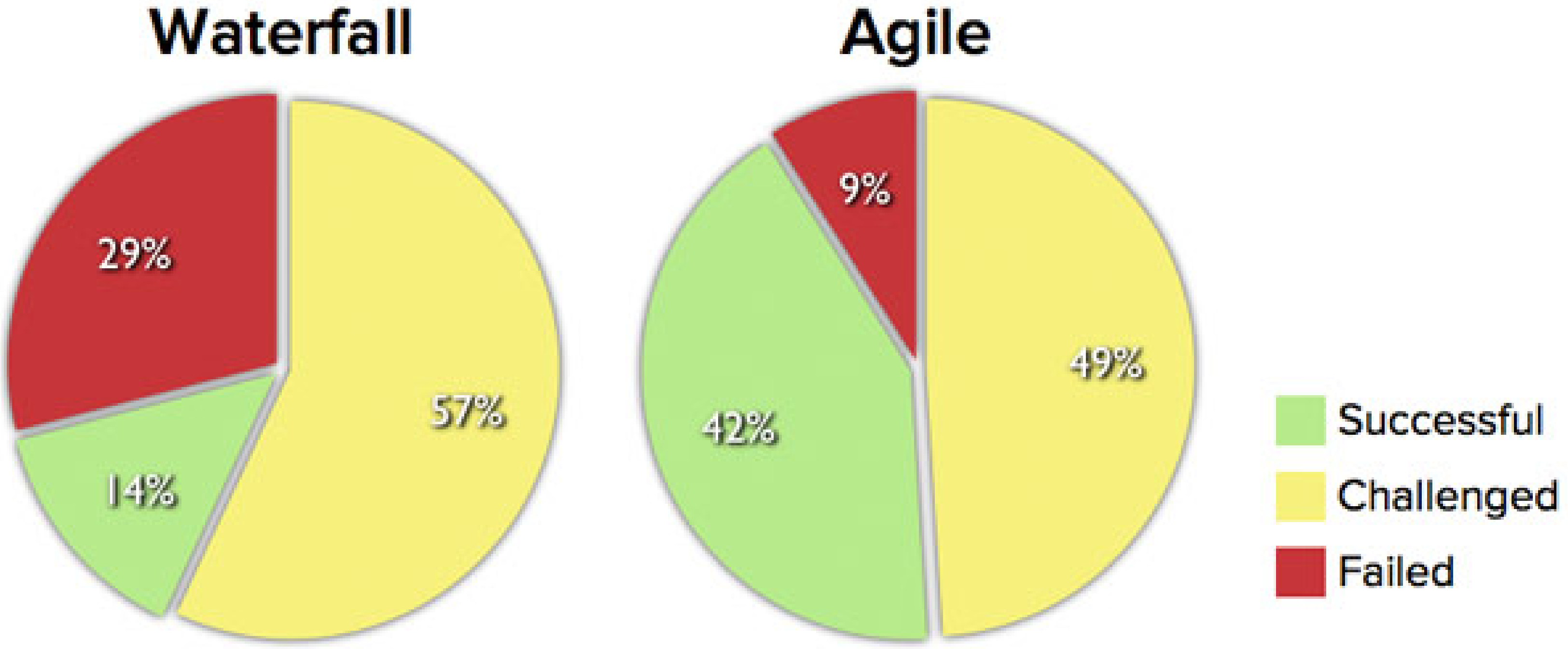

Success Rate of Change Projects

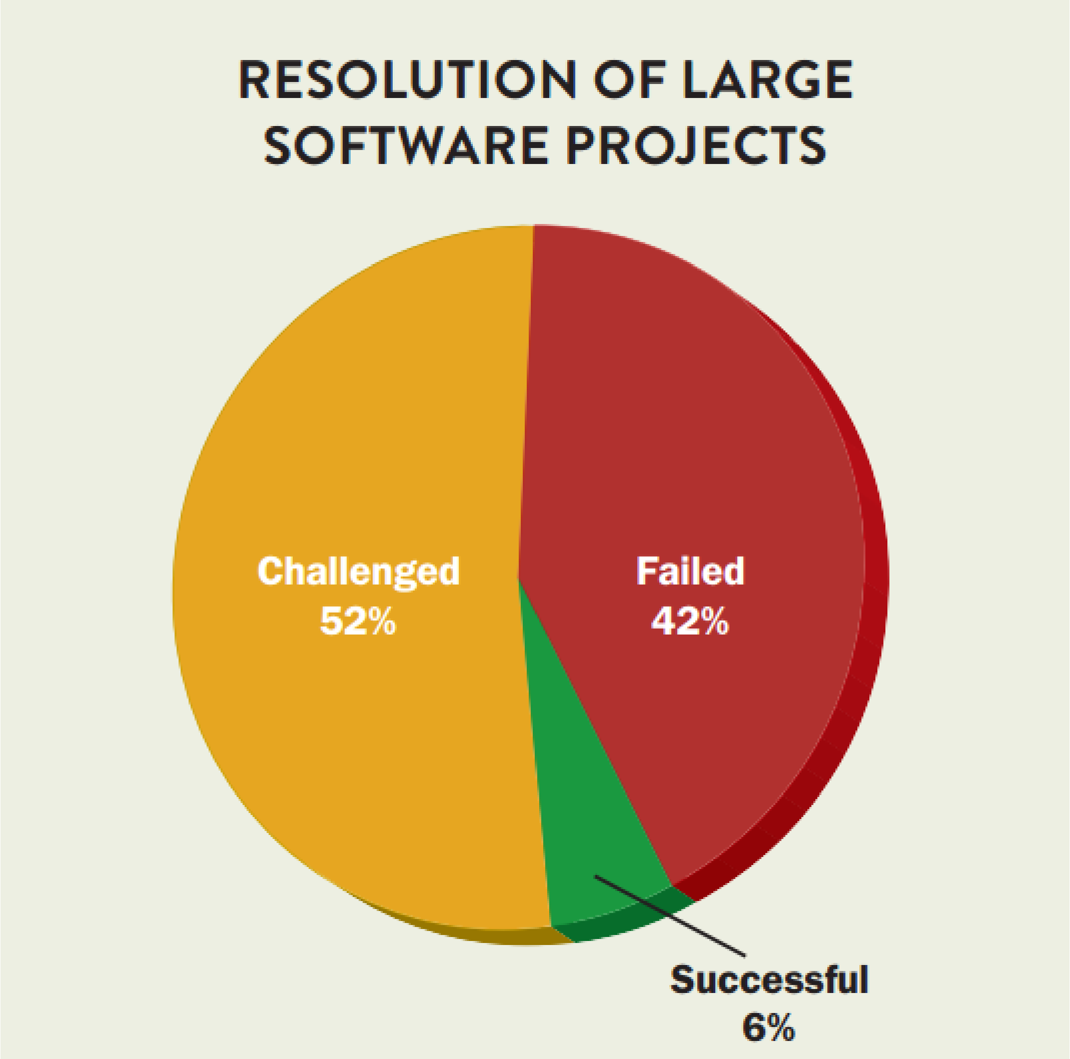
Figure 1 Source: Standish Group, Chaos Study 2011

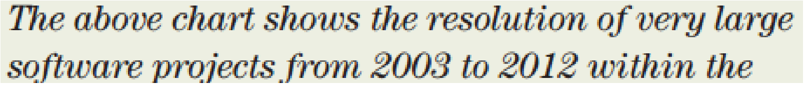
Chaos Report of Standish Group



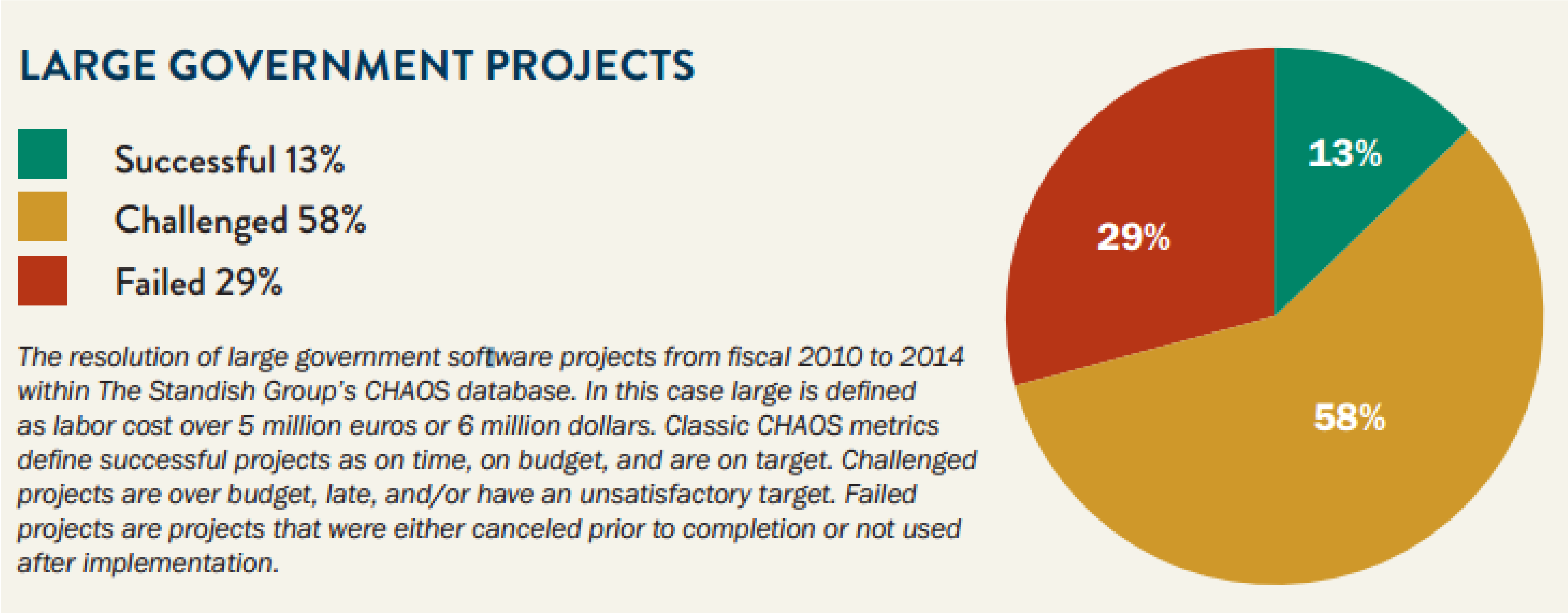
Source: The CHAOS Manifesto, The Standish Group, 2012.

Chaos Report of Standish Group




The above chart shows the resolution of very large software projects from 2003 to 2012 within the

Chaos Report of Standish Group

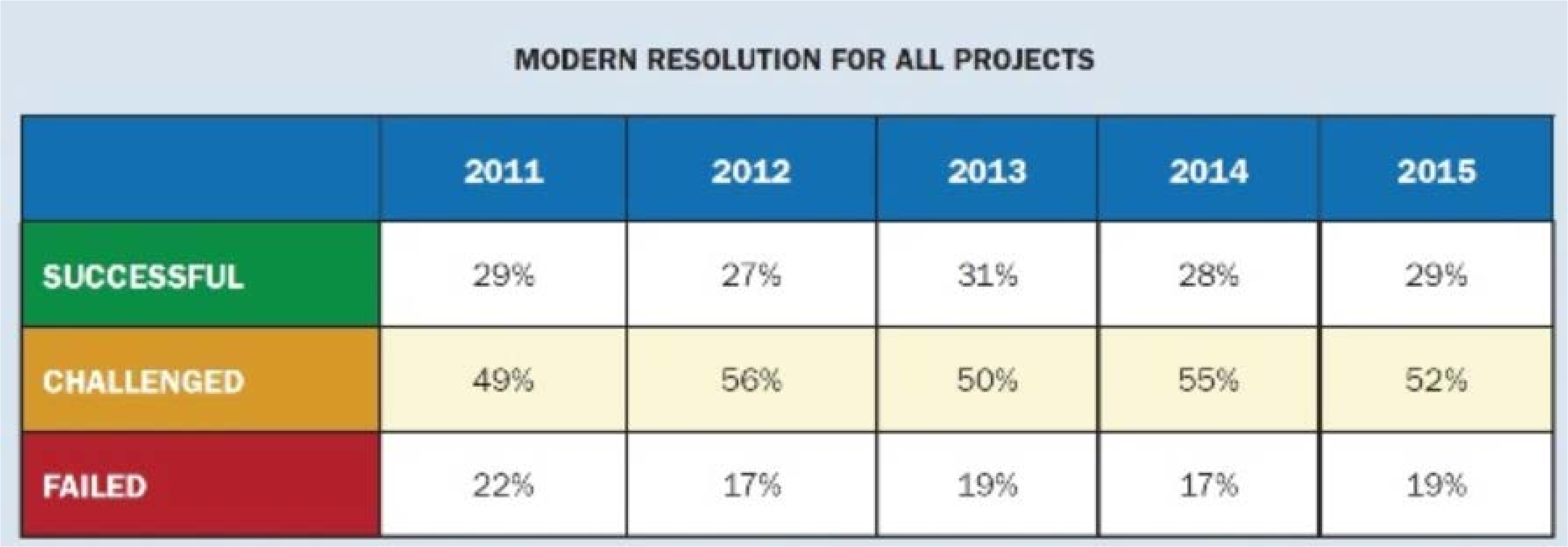


LARGE GOVERNMENT PROJECTS

- 🟩 Successful 13%
- 🟨 Challenged 58%
- 🟥 Failed 29%

The resolution of large government software projects from fiscal 2010 to 2014 within The Standish Group's CHAOS database. In this case large is defined as labor cost over 5 million euros or 6 million dollars. Classic CHAOS metrics define successful projects as on time, on budget, and are on target. Challenged projects are over budget, late, and/or have an unsatisfactory target. Failed projects are projects that were either canceled prior to completion or not used after implementation.

Chaos Report of Standish Group



MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

UML Overview

Topics

- Definition of UML
- History of UML
- UML diagrams
- Features of UML
- UML applications
- Modeling and UML

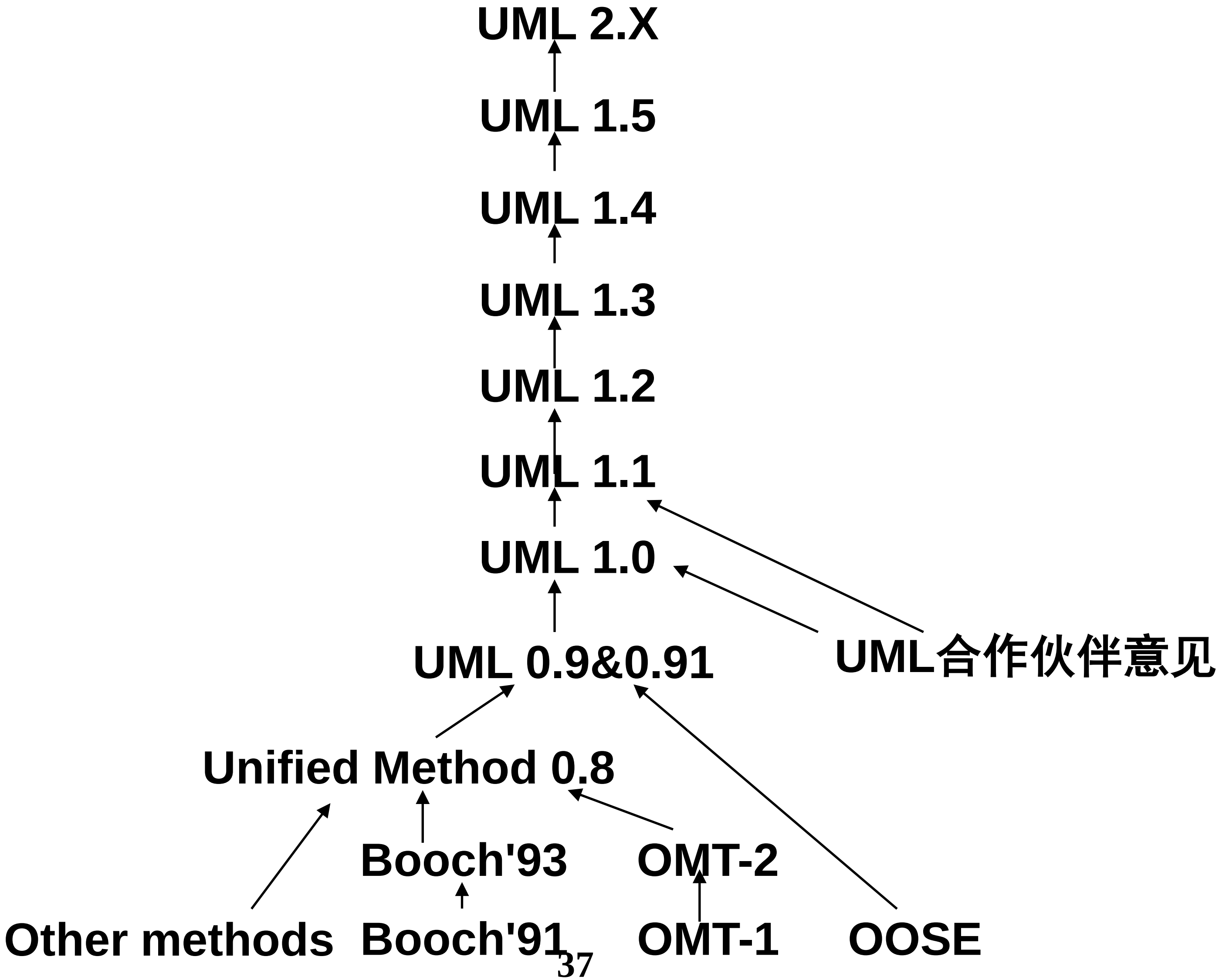
What is UML

- Unified Modeling Language
  - The Unified Modeling Language (UML) is a standard language for writing software blueprints. The UML may be used to visualize, specify, construct, and document the artifacts of a software-intensive system
    - Booch, The Unified Modeling Language User Guide

Why learn UML?

- A picture is worth a thousand words
- To communicate with others
- To give a graphical representations to your ideas
- To avoid confusion in design of system

History of UML


UML 2.X

UML 1.5

UML 1.4

UML 1.3

UML 1.2

UML 1.1

UML 1.0

UML合作伙伴意见

UML 0.9&0.91

Unified Method 0.8

Booch'93

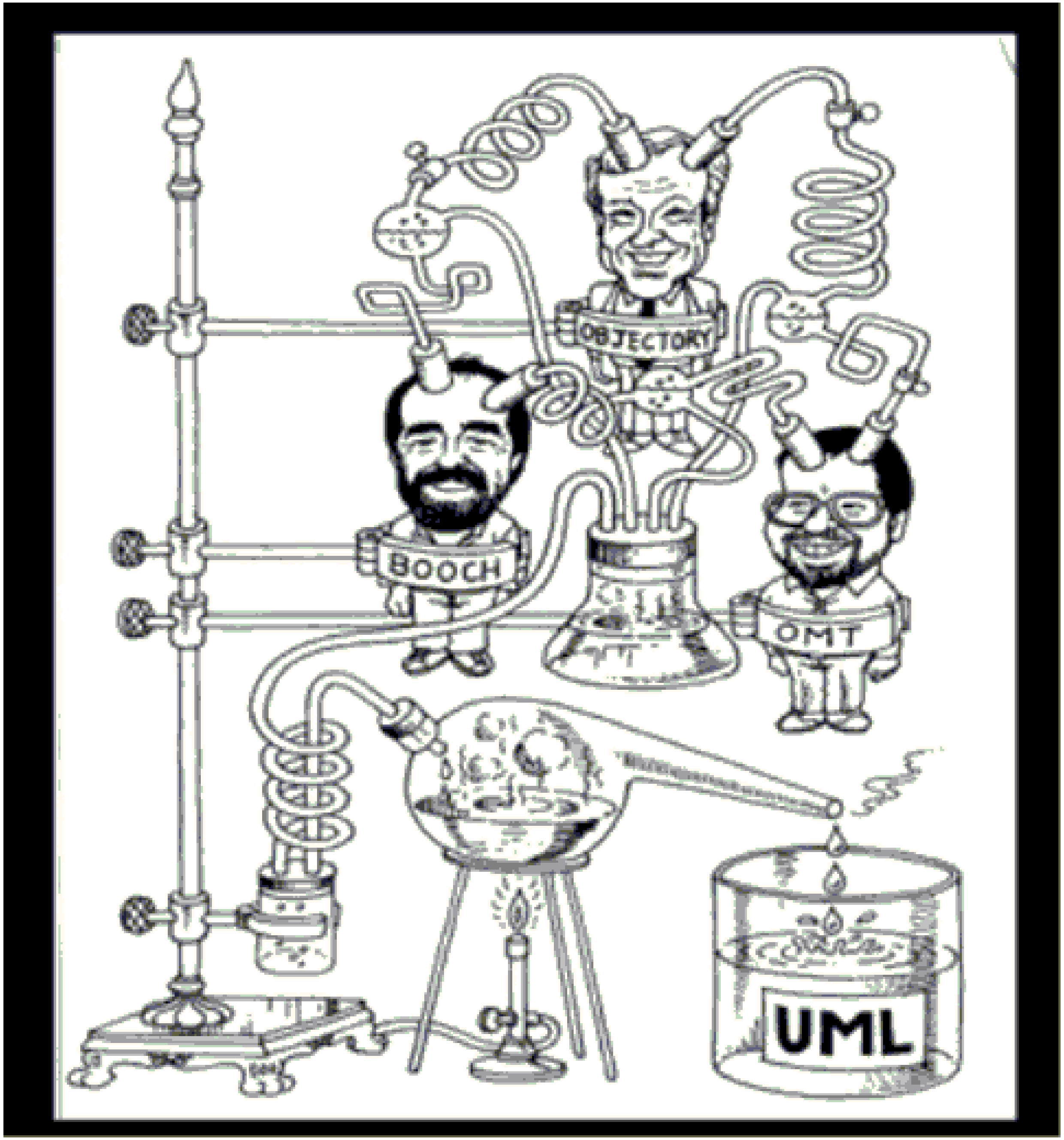
OMT-2

Other methods

Booch'91

OMT-1

OOSE


OBJECTORY
BOOCH
OMT
UML

UML Three Friends

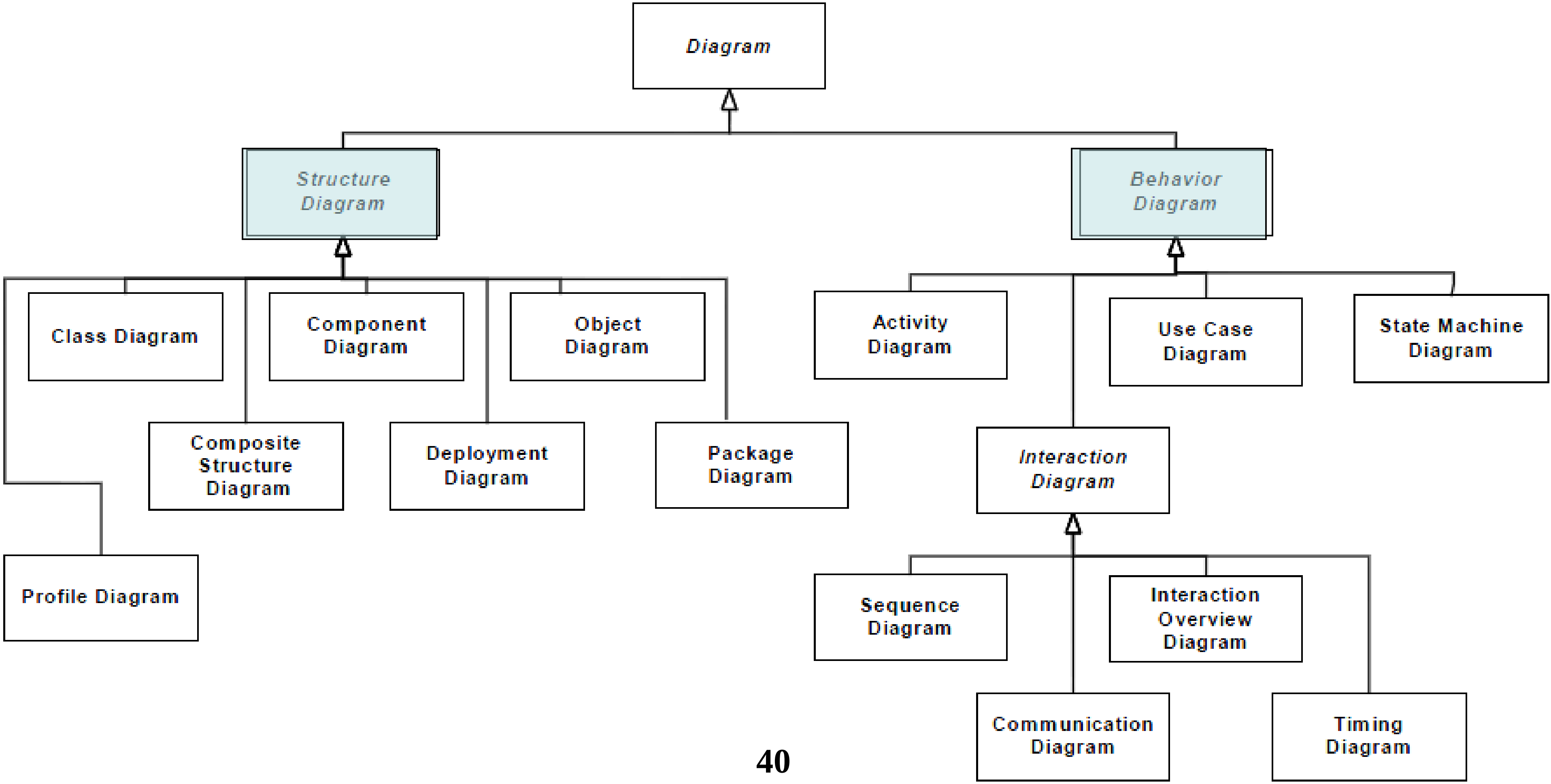
- Booch

Jacobson

- Rumbaugh

UML Diagrams

- Structural Diagrams and Behavioral Diagrams



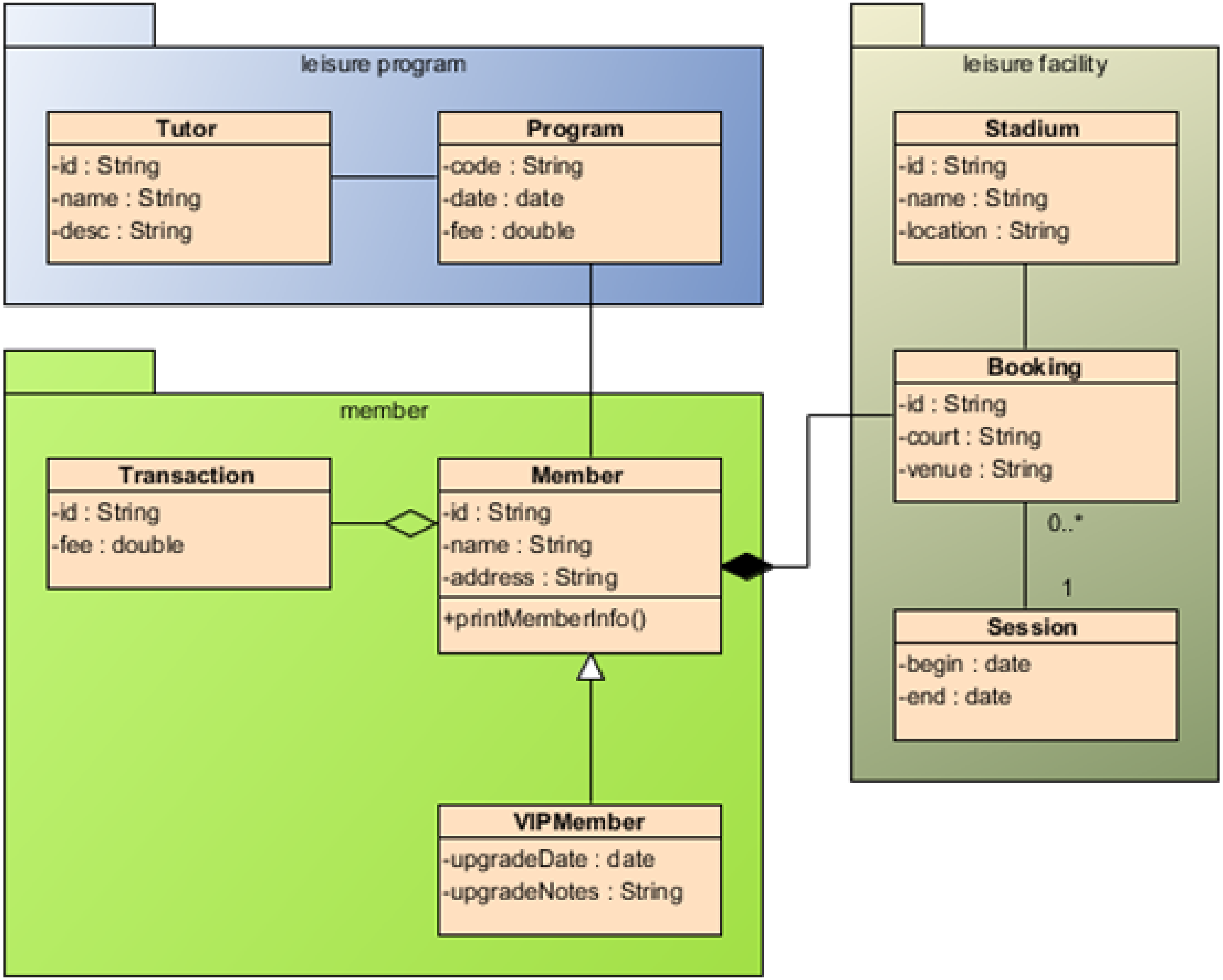
Structural Diagrams

- Class Diagram
- Component Diagram
- Object Diagram
- Deployment Diagram
- Composite Structure Diagram
- Package Diagram
- Profile Diagram

Class Diagram

- In software engineering, a class diagram in UML is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects
- May be the most commonly used diagram in practices
- Typical usage is to describe the logical design and physical design

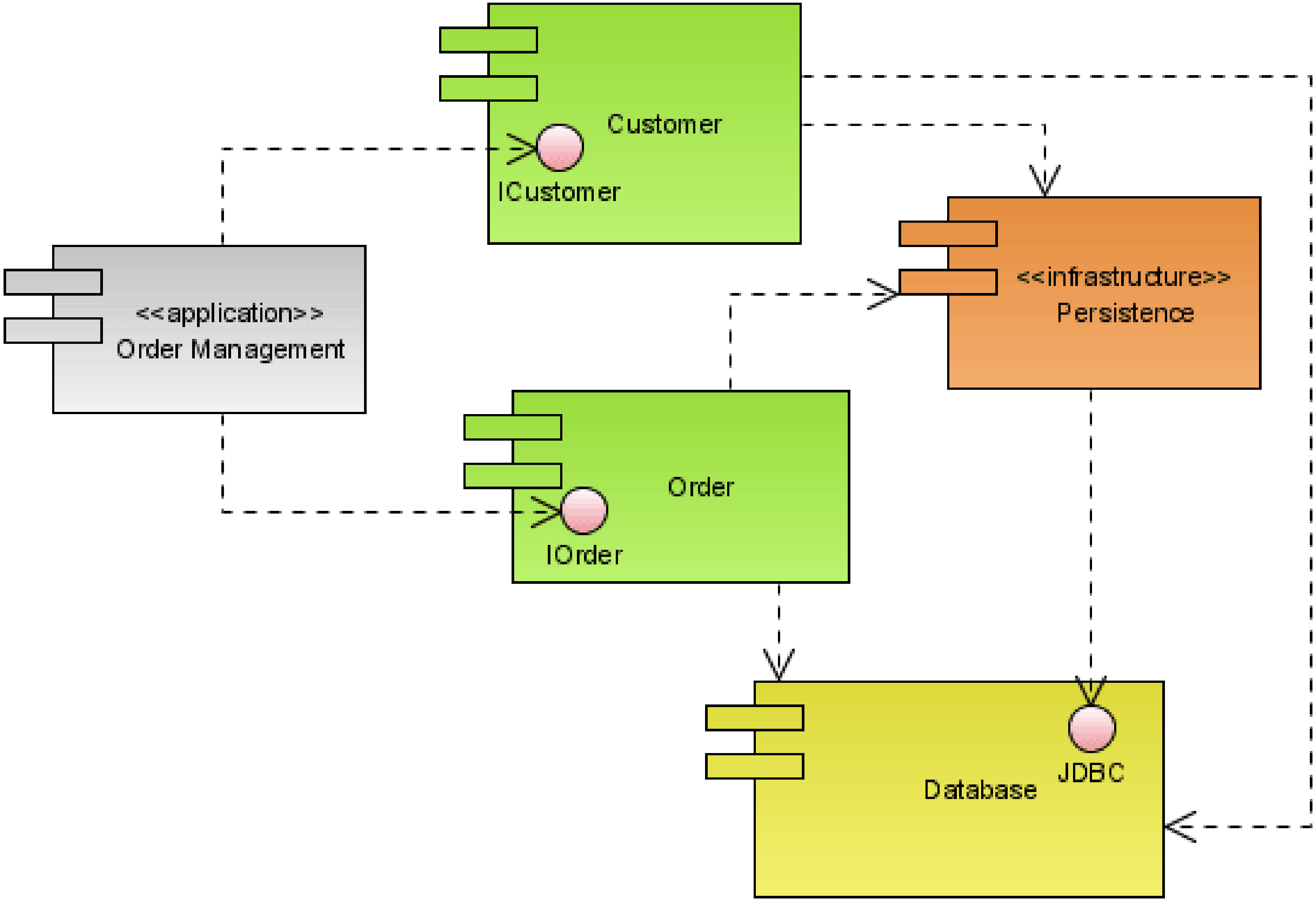
Class Diagram



Component Diagram

- An component diagram shows the internal parts, connectors, and ports that implement a component
- A component diagram also shows the organizations and dependencies among software components, including source code components, binary code components, and executable components

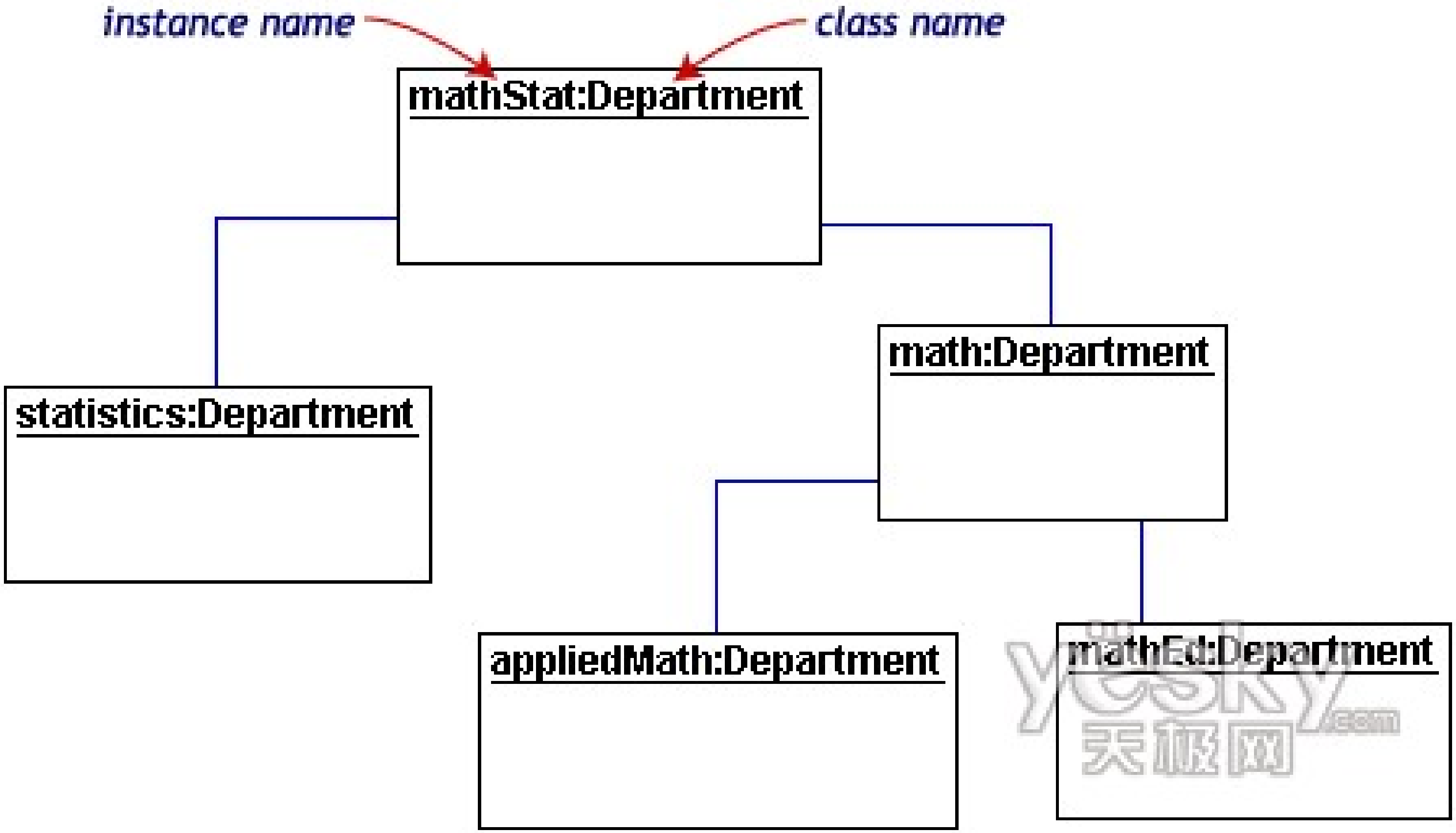
Component Diagram



Object Diagram

- An object diagram shows a set of objects and their relationships
- You use object diagrams to illustrate data structures, the static snapshots of instances of the things found in class diagrams
- Object diagrams address the static design view or static process view of a system just as class diagrams do, but from the perspective of real or prototypical cases

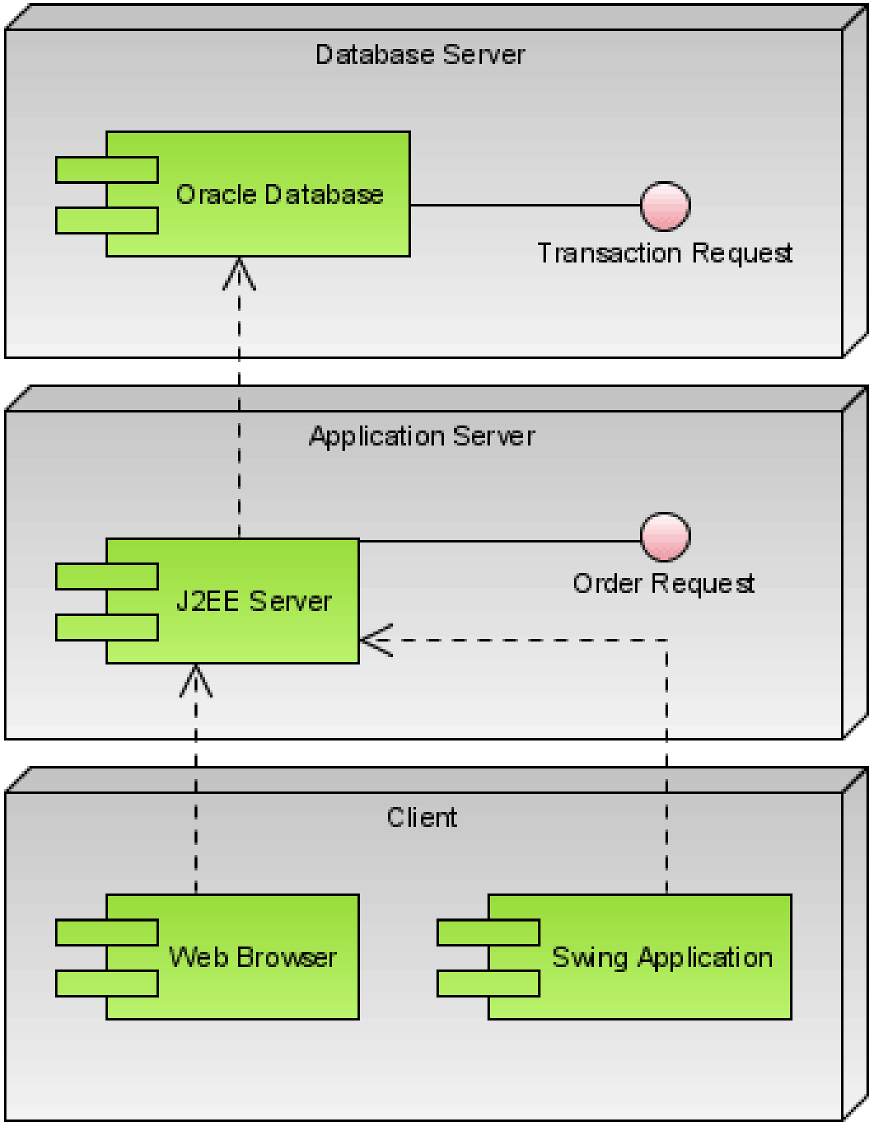
Object Diagram



Deployment Diagram

- A deployment diagram shows a set of nodes and their relationships
- You use deployment diagrams to illustrate the static deployment view of an architecture
- Deployment diagrams are related to component diagrams in that a node typically encloses one or more components

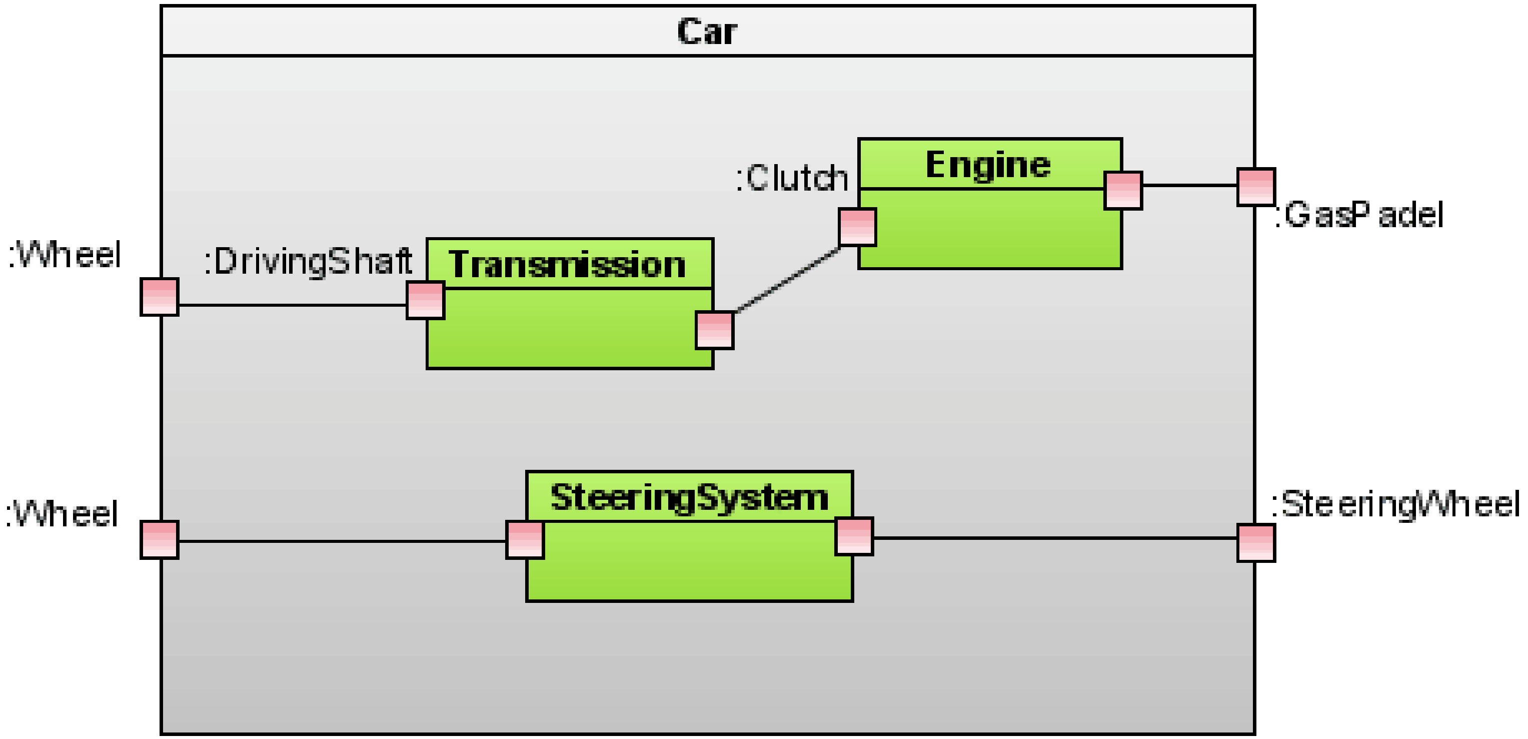
Deployment Diagram



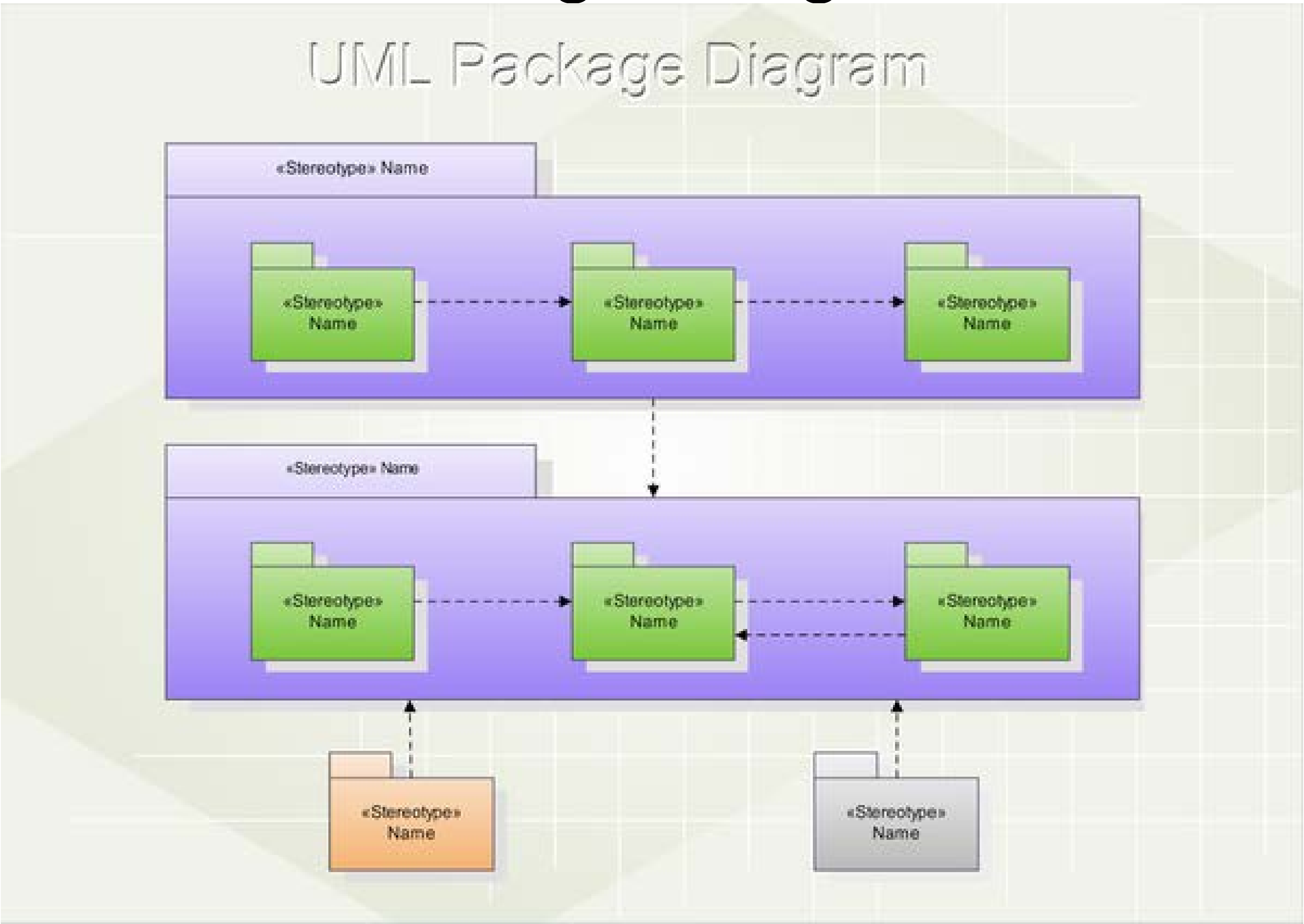
Structure Diagrams added in UML2.X

- Composite Structure Diagram
- Package Diagram
- Profile Diagram

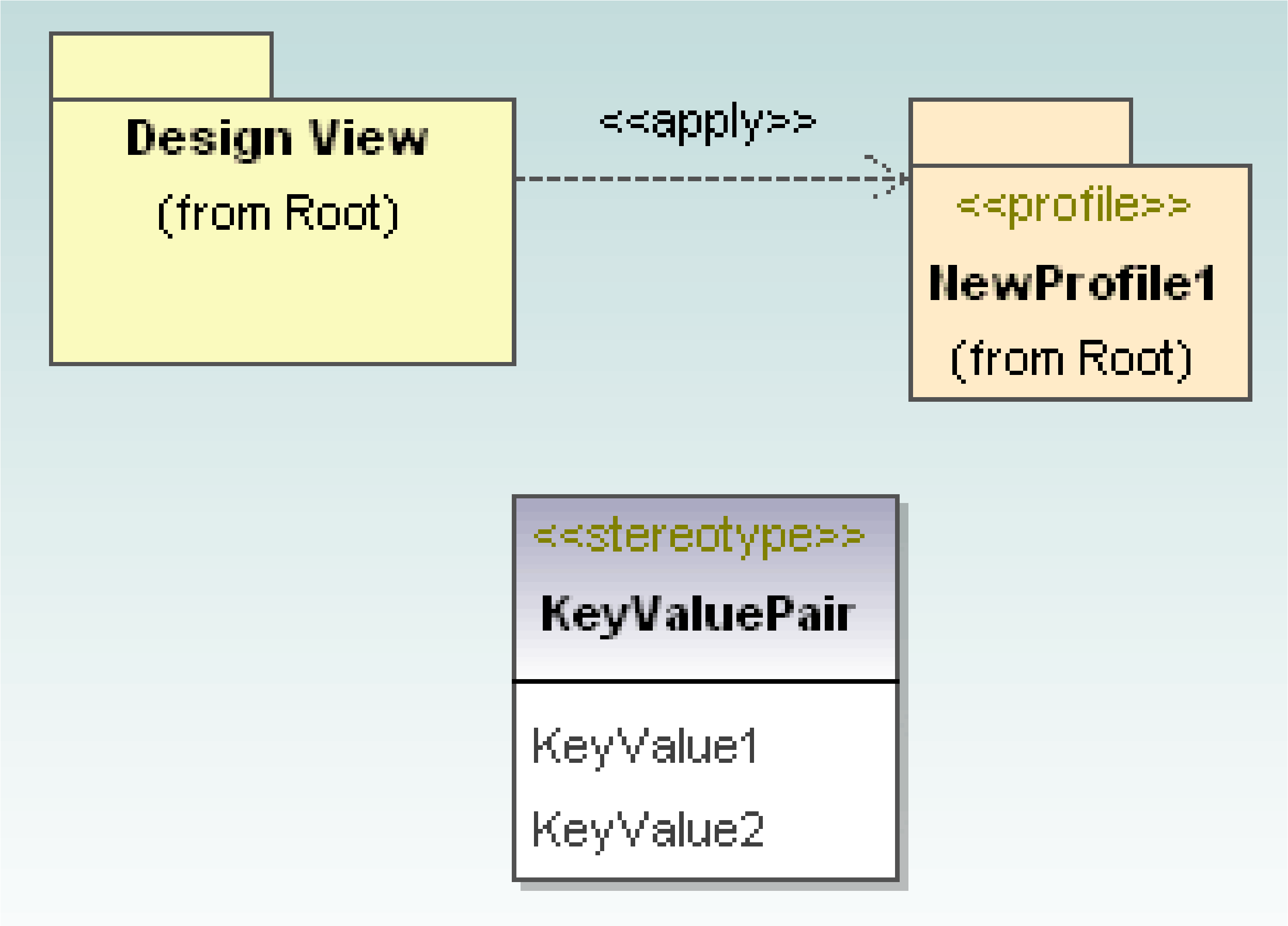
Composite Structure Diagram



Package Diagram



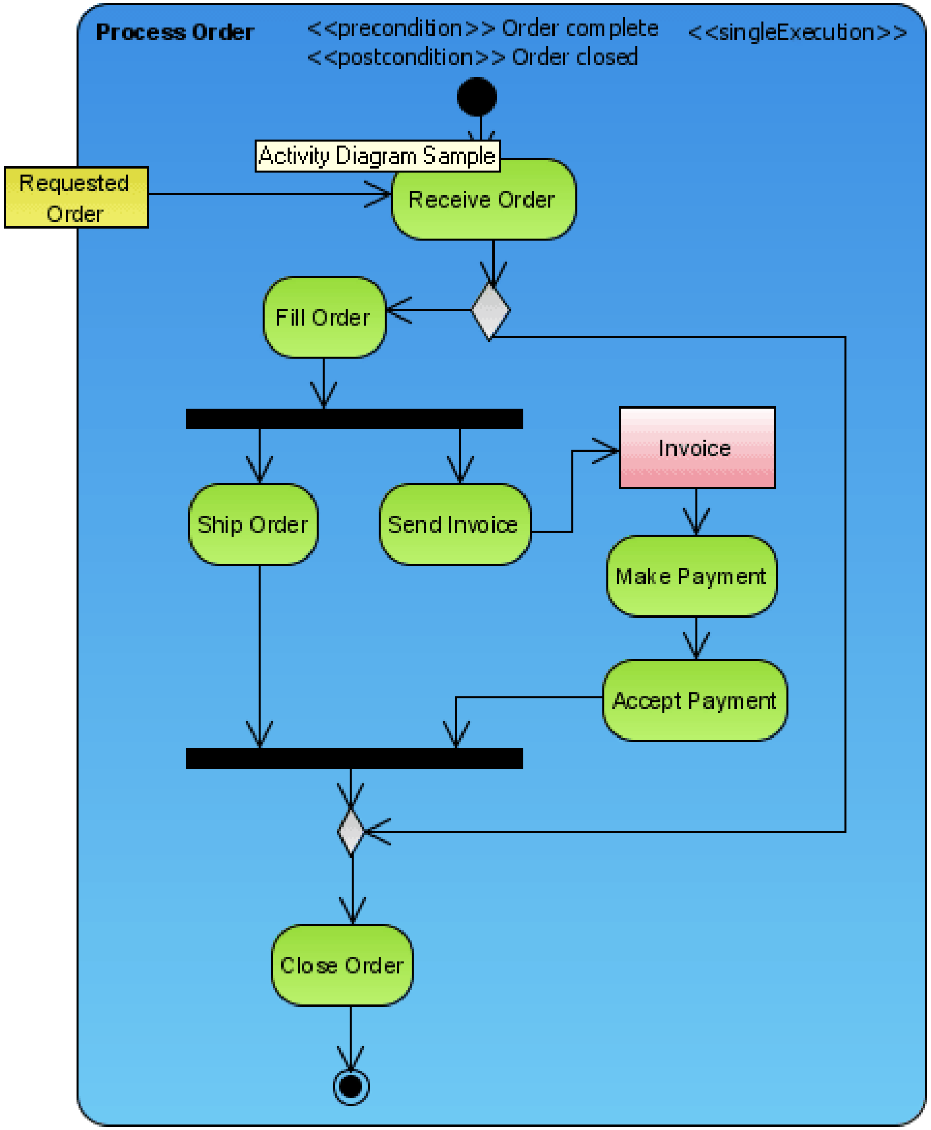
Profile Diagram



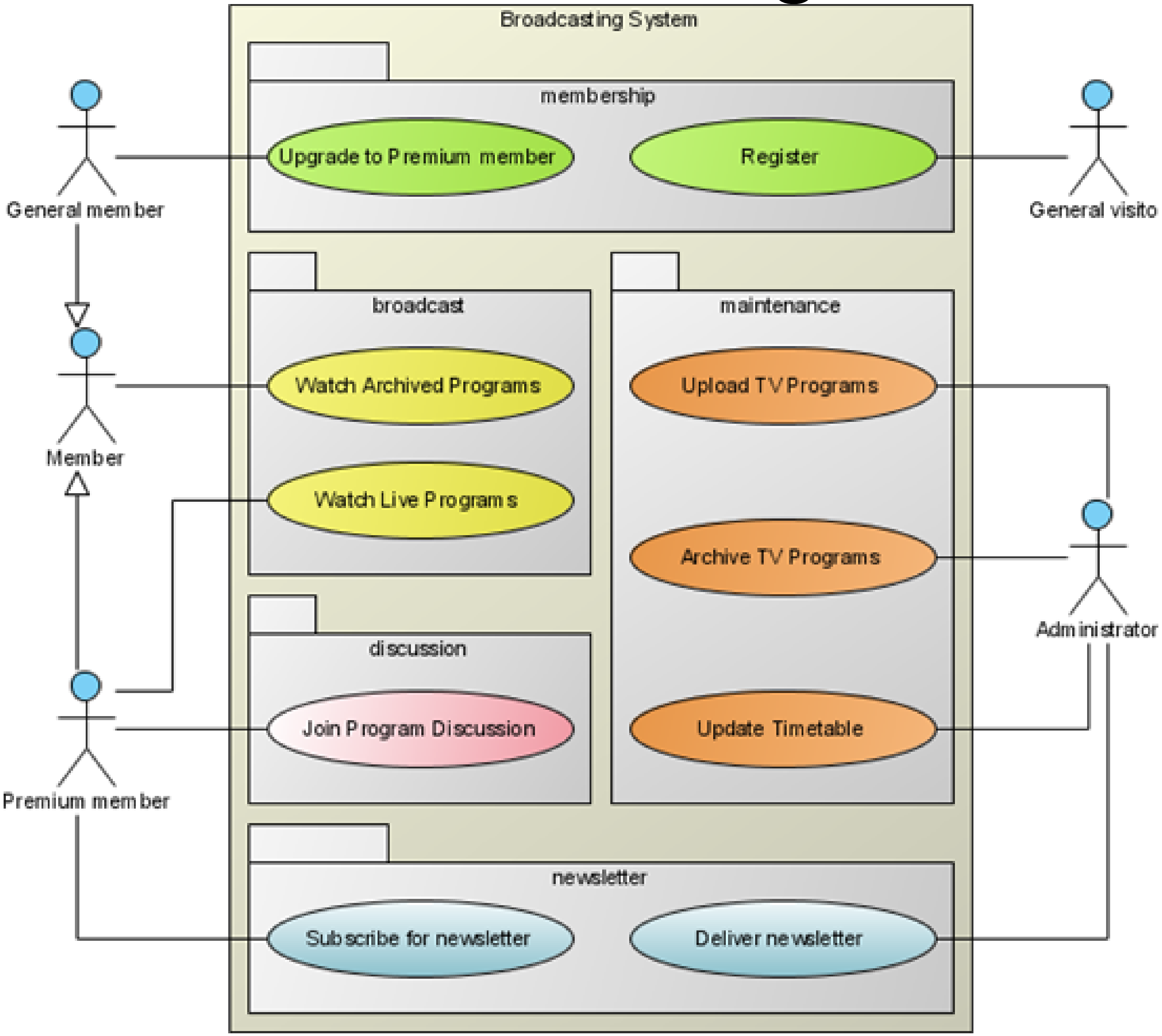
Behavioral Diagrams

- Activity Diagram
- Use Case Diagram
- State Machine Diagram
- Interaction Diagram
  - Communication Diagram
  - Sequence Diagram
  - Timing Diagram
  - Interaction Overview Diagram

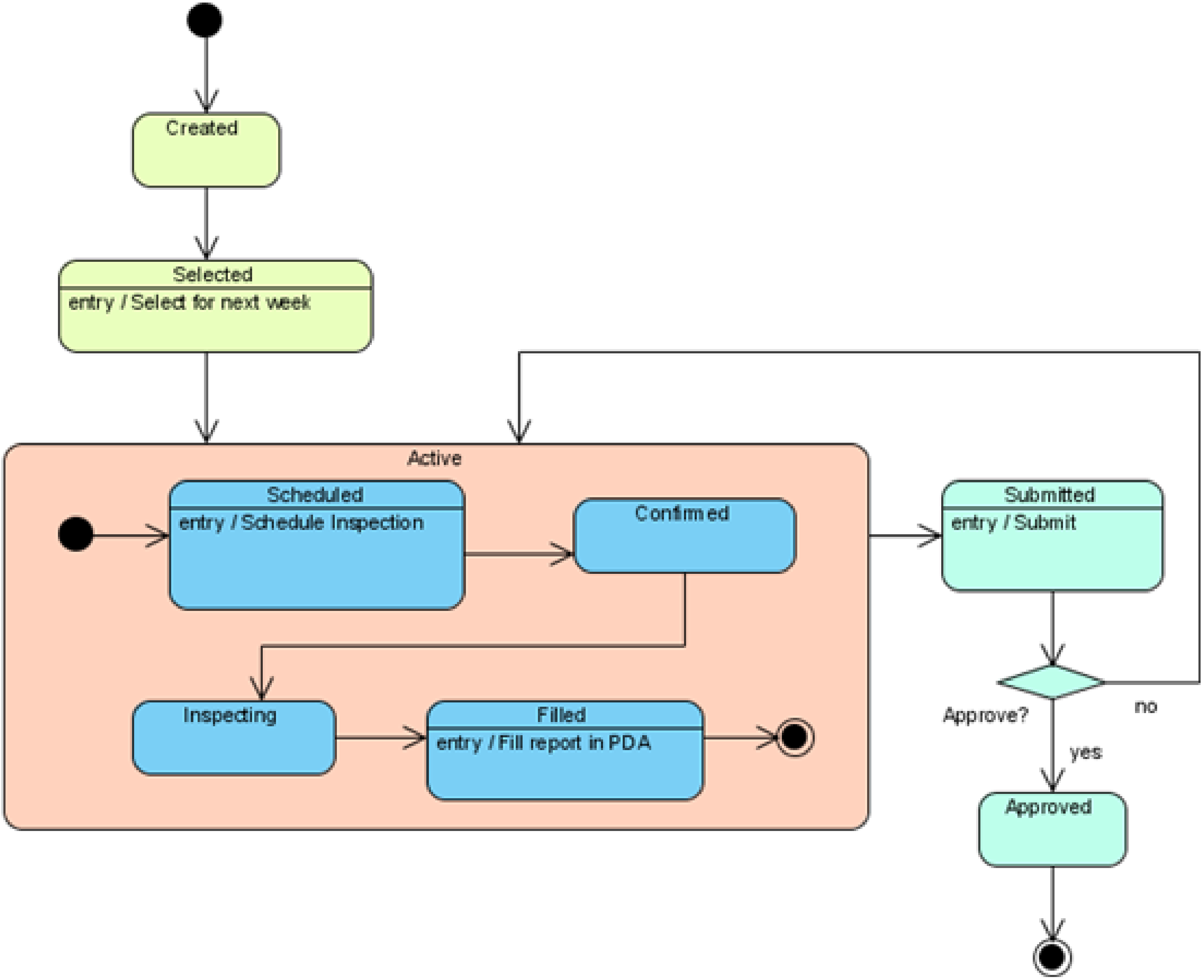
Activity Diagram



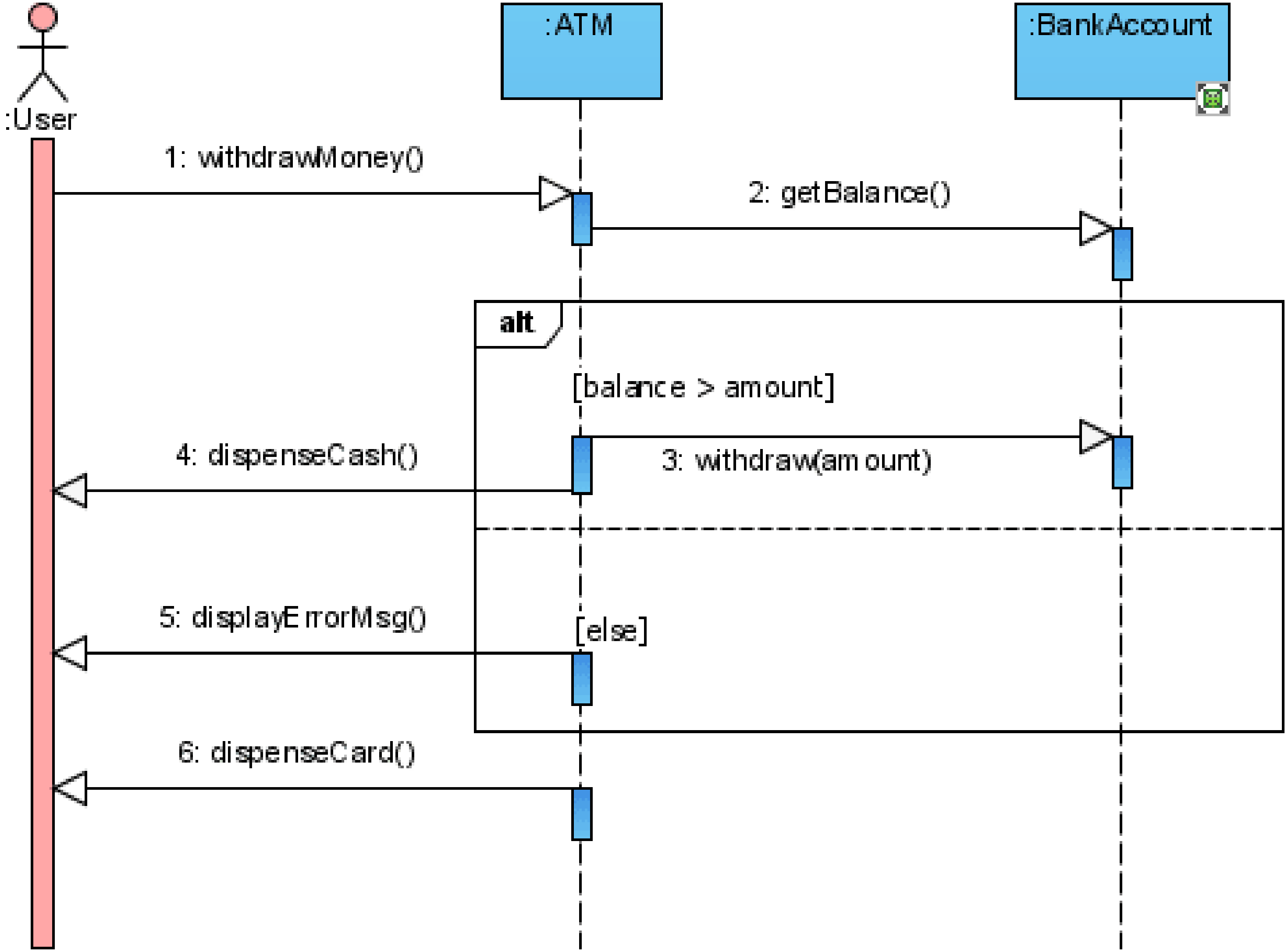
Use Case Diagram



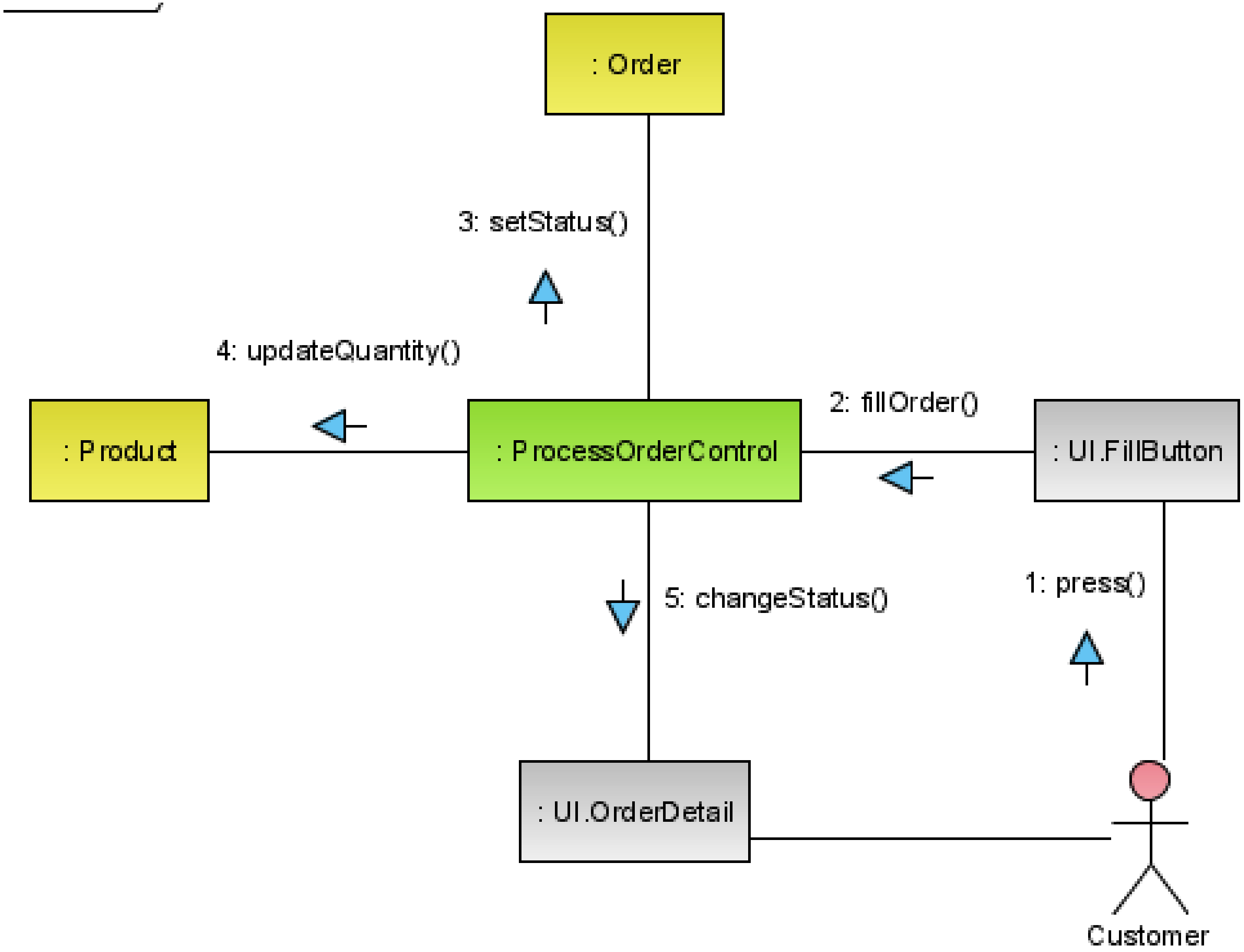
State Machine Diagram



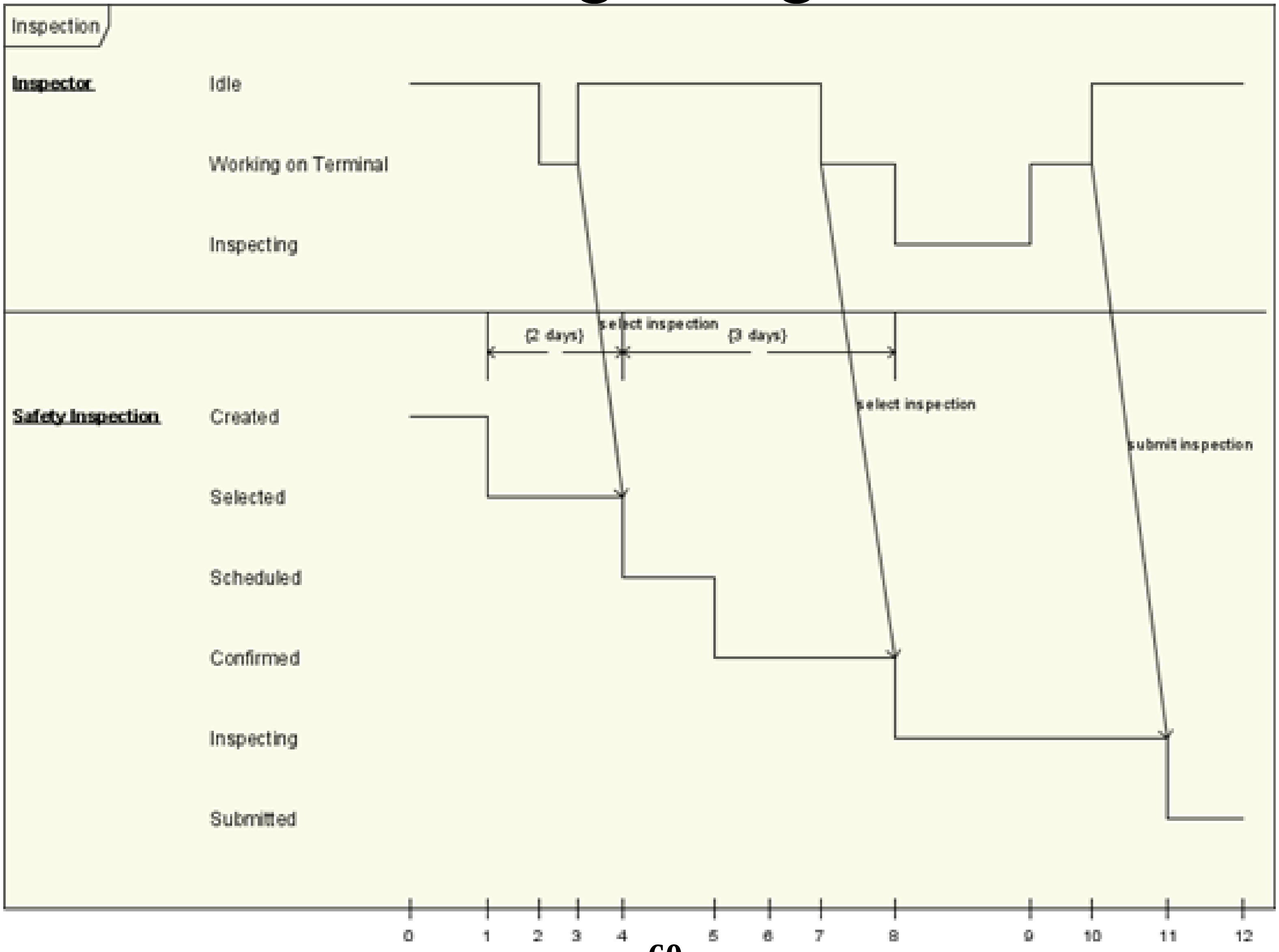
Sequence Diagram



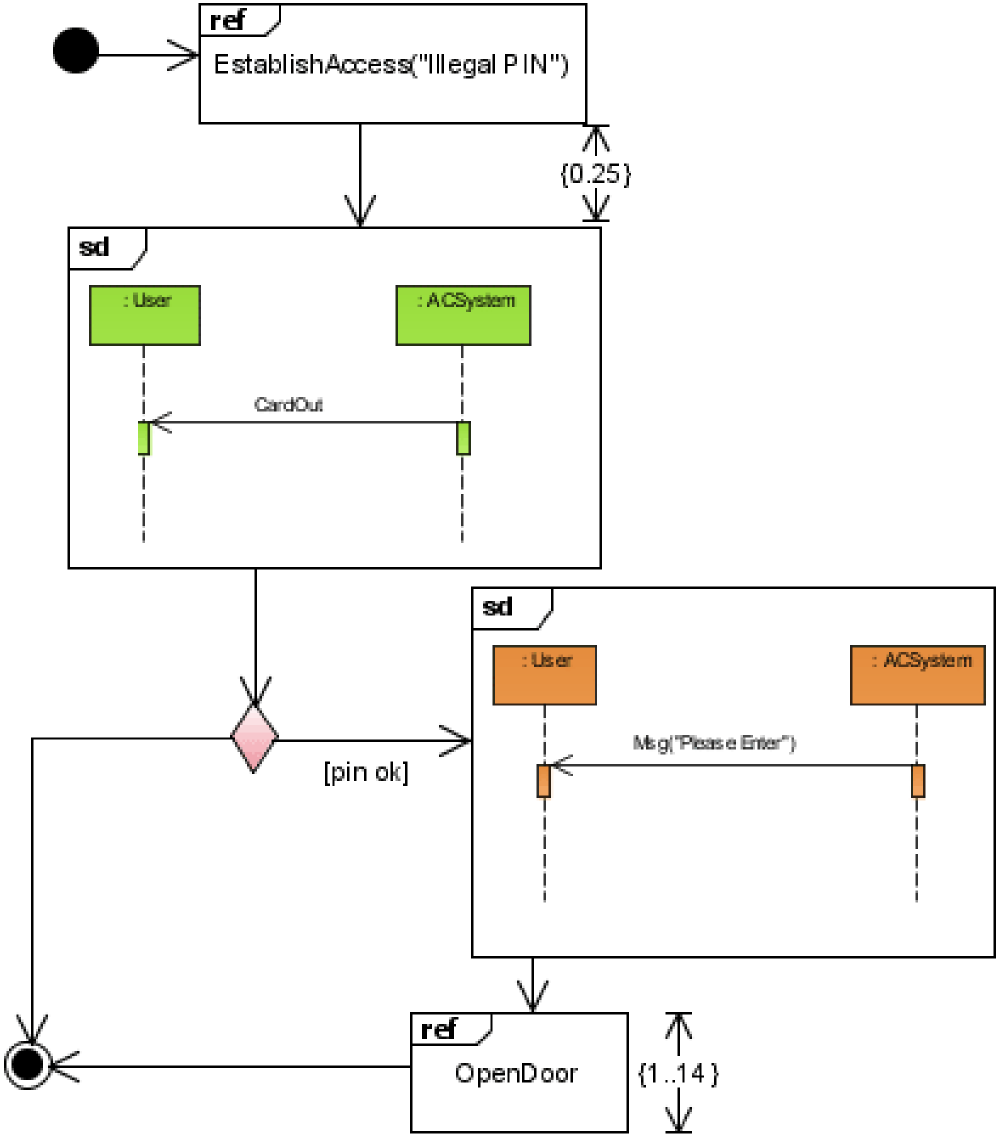
Communication Diagram



Timing Diagram



Interaction Overview Diagram



UML Diagrams May be used at different stages

- Attention
  - In the actual modeling process, almost no one uses all the diagrams defined in UML. Some of which may never be used
- Requirements Modeling
  - Use case diagram
- View the static parts of a system
  - Class diagram
  - Component diagram
  - Composite structure diagram
  - Object diagram
  - Deployment diagram
- View the dynamic parts of a system
  - Use case diagram
  - Sequence diagram
  - Communication diagram
  - State diagram
  - Activity diagram

UML Diagrams

Diagram	Learning Priority
Activity Diagram	High
Class Diagram	High
Communication Diagram	Low【High】
Component Diagram	Medium
Composite Structure Diagram	Low
Deployment Diagram	Medium
Interaction Overview Diagram	Low
Object Diagram	Low
Package Diagram	Low
Sequence Diagram	High
State Machine Diagram	Medium
Timing Diagram	Low
Use Case Diagram	Medium【High】

Features of UML

- Has been accepted as the standard modeling language by OMG
- Support for object-oriented development
- The ability of visualization and expression are strong
- Independent of the development process, and can be applied to different software development process
- Independent of the programming language

UML Applications

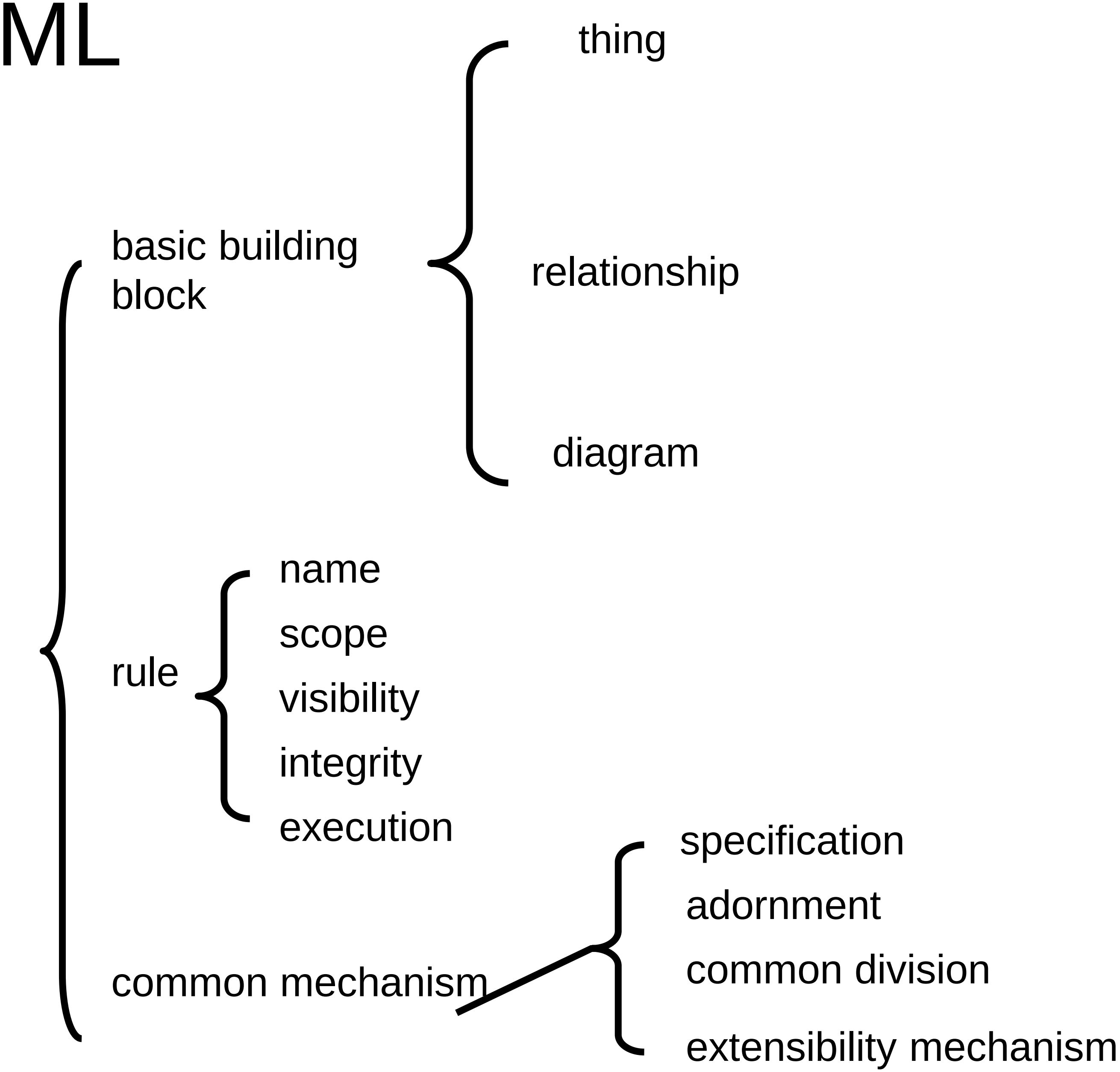
- Software system modeling
- Description of non-software systems

Q & A

- What is UML?
- In the software development process, why should we use the UML?
- Is UML a kind of programming language?
- What is the difference between UML and programming language？

A Conceptual Model of the UML

UML



- basic building block
  - thing
  - relationship
  - diagram
- rule
  - name
  - scope
  - visibility
  - integrity
  - execution
- common mechanism
  - specification
  - adornment
  - common division
  - extensibility mechanism

Topics

- Building blocks of the UML
- Things in the UML
- Relationships in the UML
- Diagrams in the UML
- Rules of the UML
- Common Mechanisms in the UML

Building Blocks of the UML

- The vocabulary of the UML encompasses three kinds of building blocks
  - Things
  - Relationships
  - Diagrams

Things in the UML

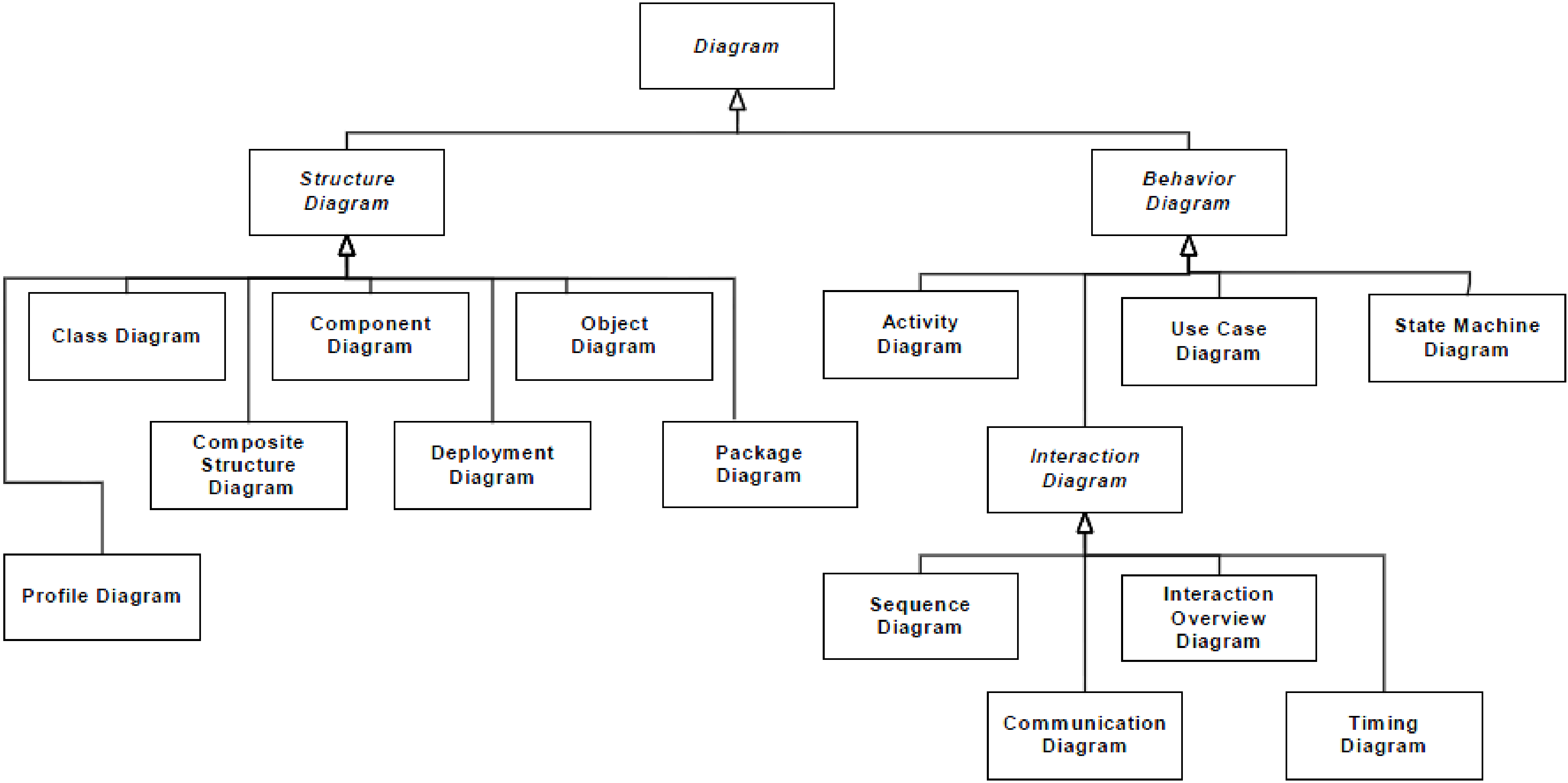
- There are four kinds of things in the UML:
  - Structural things: class, interface, collaboration, use case, active class, component, artifact & node
  - Behavioral things: interaction, state machine, activity
  - Grouping things: package
  - Annotational things: note
- These things are the basic object-oriented building blocks of the UML

Relationships in the UML

- There are four kinds of relationships in the UML
  - Dependency
  - Association
  - Generalization
  - Realization

Diagrams in the UML

- Structural Diagrams and Behavioral Diagrams



Rules of the UML

- The UML has syntactic and semantic rules for
  - ➢ Names
    - What you can call things, relationships, and diagrams
  - ➢ Scope
    - The context that gives specific meaning to a name
  - ➢ Visibility
    - How those names can be seen and used by others
  - ➢ Integrity
    - How things properly and consistently relate to one another
  - ➢ Execution
    - What it means to run or simulate a dynamic model

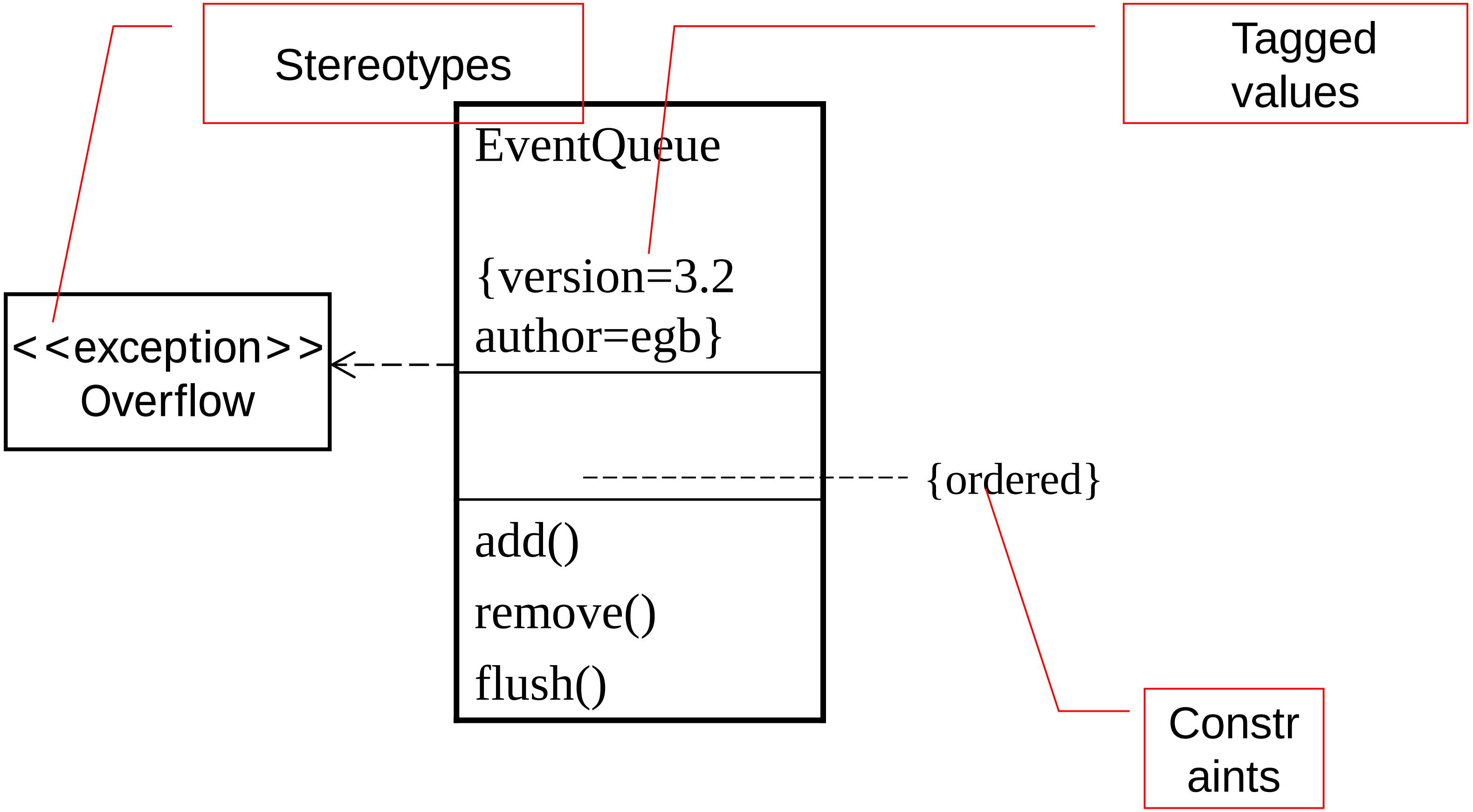
Common Mechanisms in the UML

- It is made simpler by the presence of four common mechanisms that apply consistently throughout the language
  - ➢Specifications
  - ➢Adornments
  - ➢Common divisions
  - ➢Extensibility mechanisms

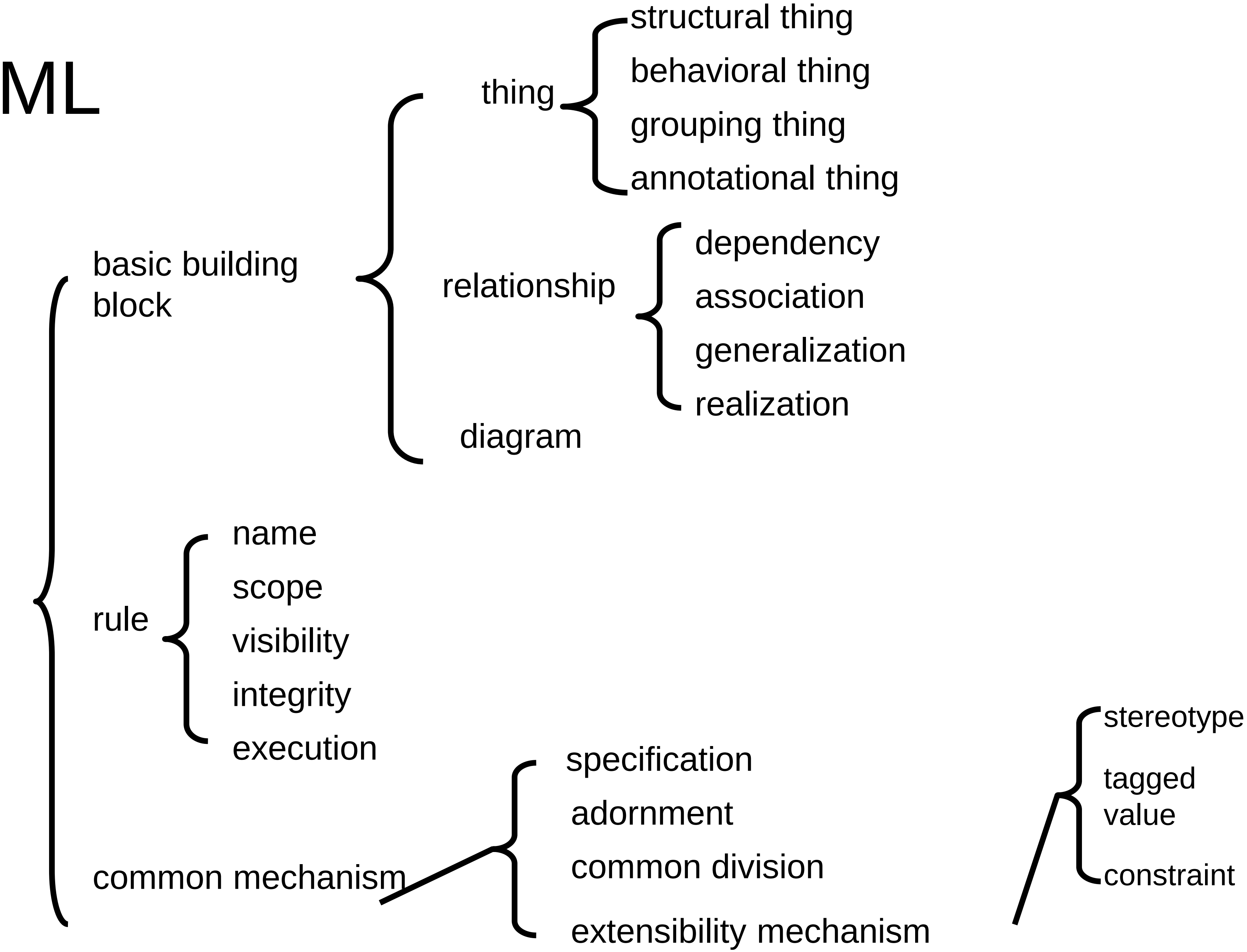
Extensibility mechanisms in the UML

- The UML is opened-ended, making it possible for you to extend the language in controlled ways
- The UML's extensibility mechanisms include
  - Stereotypes
  - Tagged values
  - Constraints

Extensibility mechanisms in the UML

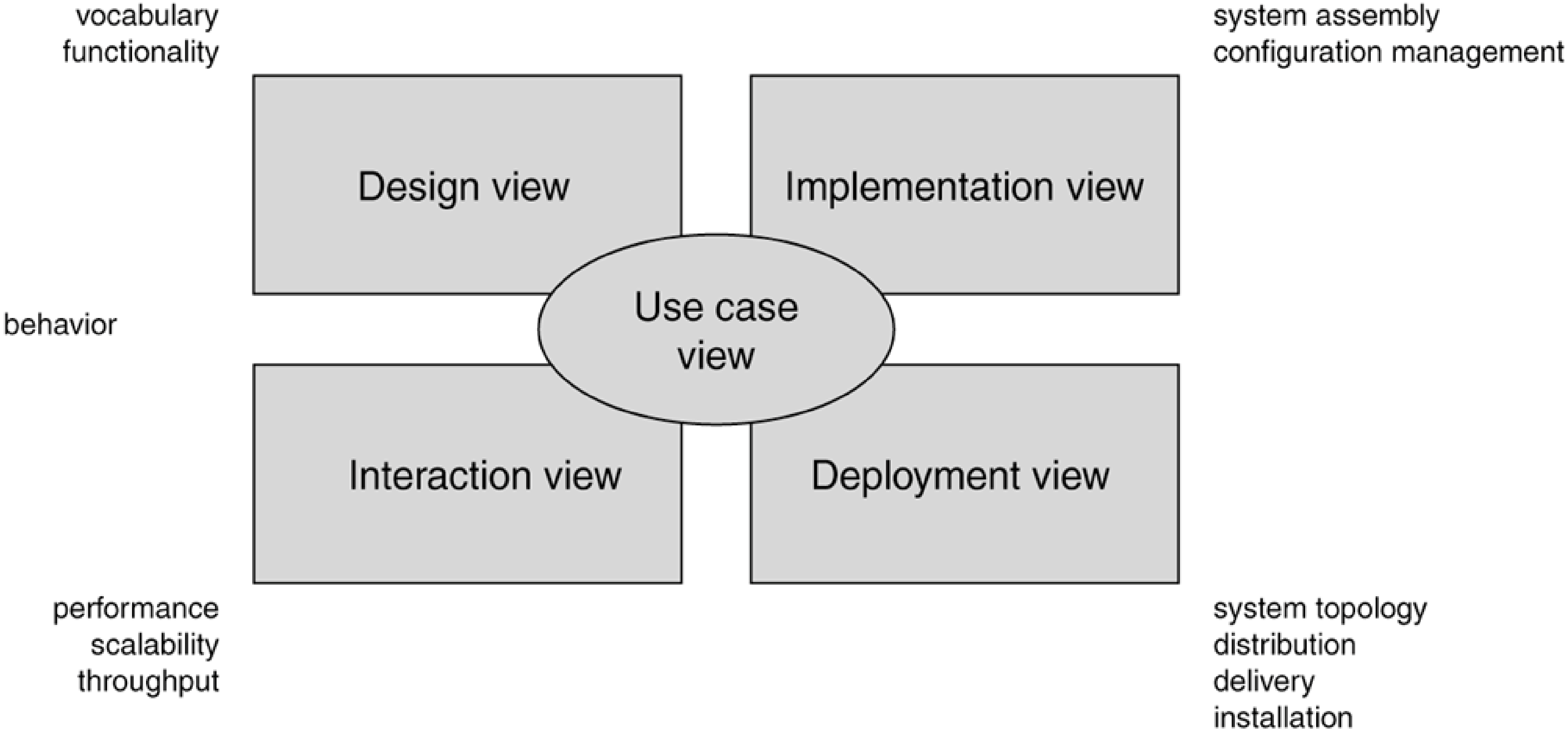


UML



- basic building block
  - thing
    - structural thing
    - behavioral thing
    - grouping thing
    - annotational thing
  - relationship
    - dependency
    - association
    - generalization
    - realization
  - diagram
- rule
  - name
  - scope
  - visibility
  - integrity
  - execution
- common mechanism
  - specification
  - adornment
  - common division
  - extensibility mechanism
    - stereotype
    - tagged value
    - constraint

UML 4+1 View Model



How To Use UML Efficiently

- Under the guidance of the principles of UML modeling, select the appropriate modeling things and diagrams, and use the necessary textual description, build system model

Q & A

- Which three parts make up the UML?
  - Building blocks, Rules and Common Mechanisms
- Talk about the UML visibility rules
- Talk about the UML diagrams
- For the models, views, diagrams and model elements, please talk about your understanding

UML Tools

UML Tools

- Rational Rose
- Visual Paradigm
- StarUML
- Enterprise Architect
- ArgoUML
- Visio
- Trufun(Chinese)
- ……
- More
  - http://www.umlchina.com/Tools/Newindex1.htm