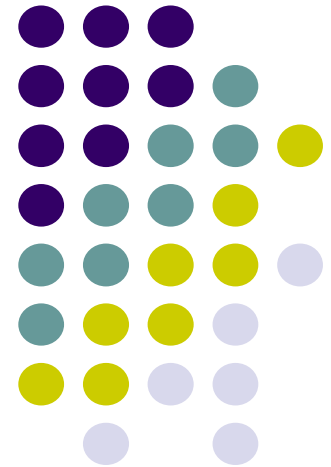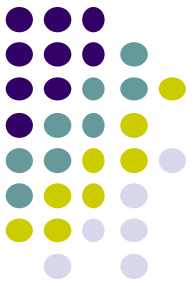# Object-Oriented Modelling with UML
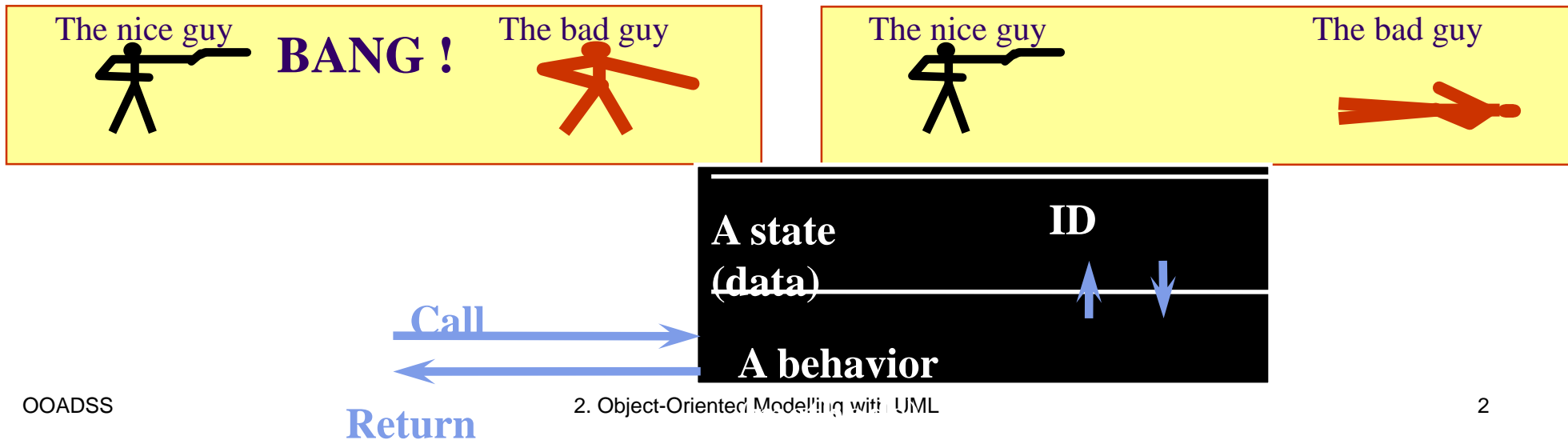## *(Unified Modeling Language)*

Objects and Systems
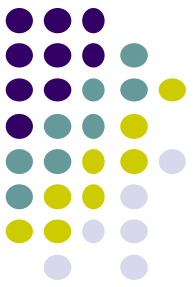
UML History

Language Objectives

UML Goals

What is an object?

- An atomic unit formed from the union of a state and a behavior

  - The encapsulation ensures the internal cohesion and the weak coupling with the outside.

- Object = Identity + Behavior + State

- A visible behavior

- Data, internal behavior, hidden. Example for J. Bush:

What is a system?

- A system is a group of objects which co-operate, having a structure and constituting an organic set.

- It is a device made of related elements constituting a coherent set.

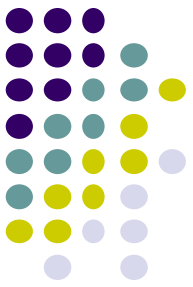- A collection of units connected and organized in order to accomplish a specific goal.
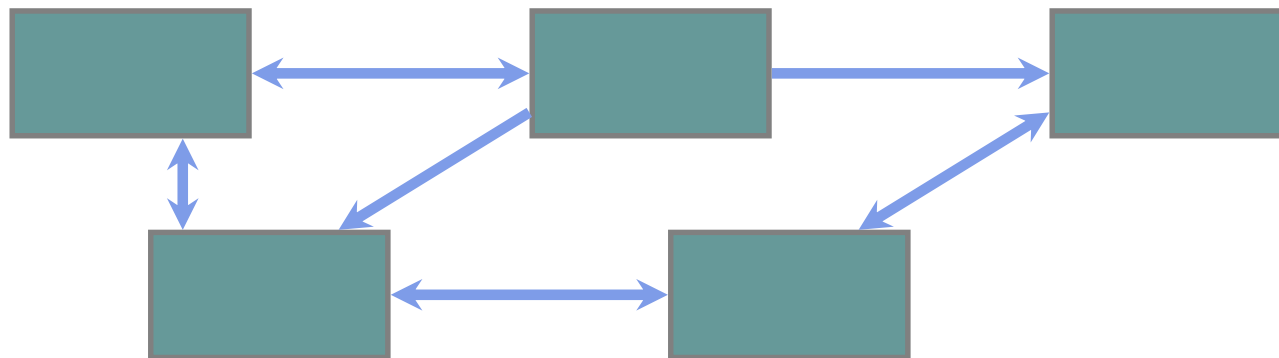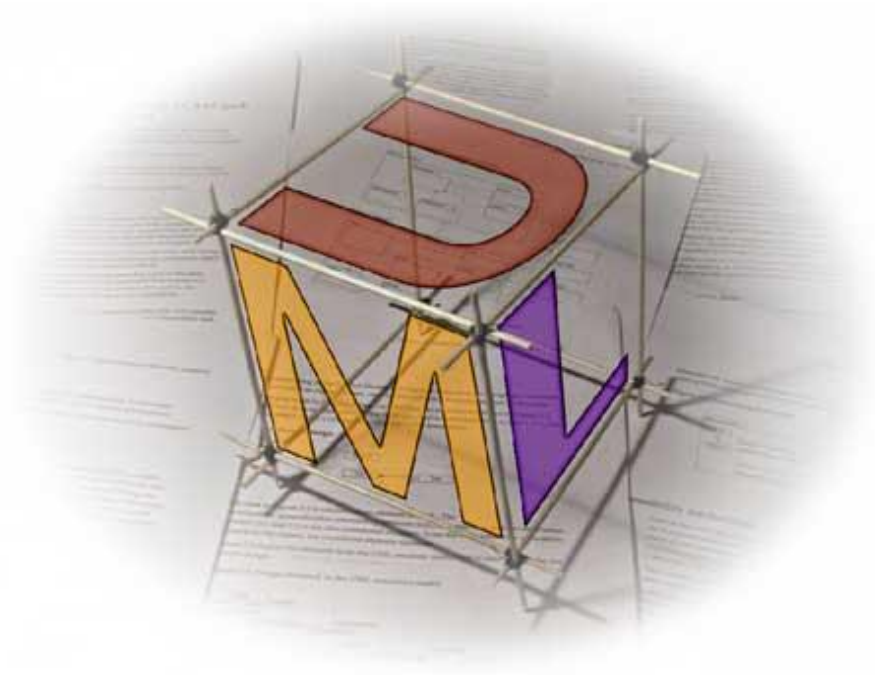
What is a model?

- **A system can be described by one or more _Models,_ i.e. from different points of view.**

- **A model:**
  - Represents a view of part of the system
  - Is formalized (construction rules)
  - Is based on a certain number of concepts
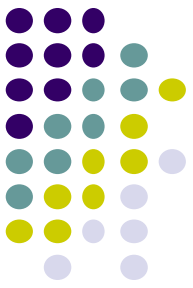 - May be shown graphically

UML

- **Unified**
- **Modeling**
- **Language**

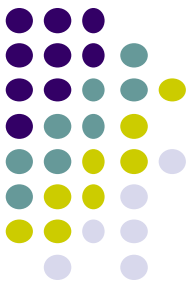# UML is an OO language for:

- **specifying,**
- **visualizing,**
- **constructing, and**
- **documenting**

  the artifacts of software systems, as well as for business modeling and other non-software systems.
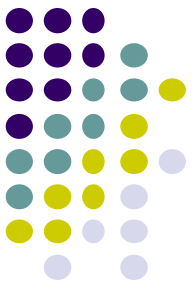
As a modeling language UML includes:

- **Model elements** — fundamental modeling concepts and semantics

- **Notation** — visual rendering of model elements

- **Guidelines** — idioms of usage

Goals of UML

- Provide users with **a ready-to-use, expressive visual modeling language to develop and exchange meaningful models.**

- Furnish extensibility and specialization mechanisms to extend **the core concepts:**

  1. build models using core concepts without using extension mechanisms for most normal applications,
  2. add new concepts and notations for issues not covered by the core,
  3. choose among variant interpretations of existing concepts, when there is no clear consensus,
  4. specialize the concepts, notations, and constraints for particular application domains.

Goals of the UML (cont.)

- Support specifications that are independent of particular programming languages and development processes.

- Provide a formal basis for understanding the modeling language.

- Encourage the growth of the object tools market.

- Support higher-level development concepts such as components, collaborations, frameworks and patterns.

- Integrate best practices.

Scope of the UML

- UML fuses the concepts of Booch, OMT, and OOSE, in a single, common, and widely usable modeling language.
- UML pushes the envelope of what can be done with existing methods (UML authors targeted the modeling of concurrent, distributed systems).
- UML focuses on a standard modeling language, not a standard process.

The 4 objectives of UML Authors

- Represent whole systems by object concepts
- Establish an explicit link between concepts and executables
- Take in mind the scale factors inherent in complex systems
- Create a useful language for humans and machines.

Outside the scope of the UML
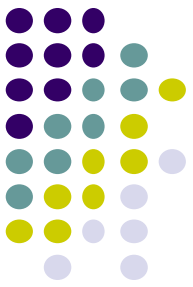
- ***Programming Languages -*** **the UML is a visual *modeling* language (for visualizing, specifying, constructing, and documenting the artifacts), but not a visual *programming* language.**

- ***Tools -*** **the UML defines a semantic metamodel, not a tool interface, storage, or run-time model.**

- ***A process* :**

• **provides guidance as to the order of a team's activities,**

• **specifies what artifacts should be developed,**

• **directs the tasks of individual developers and the team as a whole, and**

• **offers criteria for monitoring and measuring a project's products and activities.**

UML is process-independent
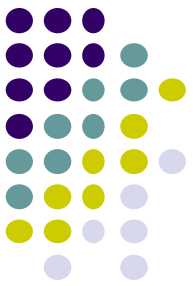
UML: History and Future

- **BOOCH 1991, 1993**

- **OMT, OMT2**

- **OOSE**

- **Unified Method V0.8**

- **Use cases - Jacobson**

- **UML V0.9, UML V1.0**

- **Others**

  Submission to OMG (Object Management Group) - 16/01/1997
  o **UML V1.1: world standard (15/11/1997)**
  o **Previous version: UML V1.5 (01/03/2003)**
  o **UML 2.4.1 – August 2011** http://www.omg.org/spec/UML/2.4.1
  o **UML 2.5 – August 2015**
  o **UML 2.5.1 – December 2017, http://www.omg.org/spec/UML/**

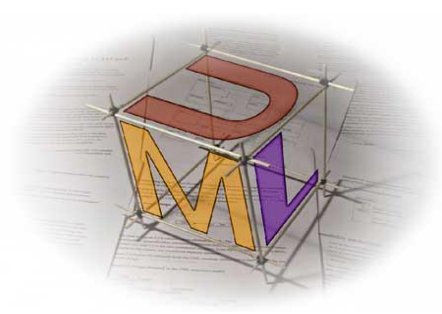# Language architecture – four-layer metamodel architecture

(from OMG UML 1.5 spec.)

Table 2-1   Four Layer Metamodeling Architecture

Layer	Description	Example
	domai	ver
user objects (user data)	An instance of a model. Defines a specific information domain.	<Acme_SW_Share_98789>, 654.56, sell_limit_order, <Stock_Quote_Svr_32123>

Language architecture – four-layer metamodel architecture

Table 2-1 Four Layer Metamodeling Architecture (from OMG UML 1.5 spec.)

Layer	Description	Example
meta-metamodel	The infrastructure for a metamodeling architecture. Defines the language for specifying metamodels.	*MetaClass, MetaAttribute, MetaOperation*
metamodel	An instance of a meta-metamodel. Defines the language for specifying a model.	*Class, Attribute, Operation, Component*
model	An instance of a metamodel. Defines a language to describe an information domain.	*StockShare, askPrice, sellLimitOrder, StockQuoteServer*
user objects (user data)	An instance of a model. Defines a specific information domain.	*<Acme_SW_Share_98789>, 654.56, sell_limit_order, <Stock_Quote_Svr_32123>*

Example of the four-layer metamodel hierarchy *(source: OMG UML Infrasructure Ver. 2.3)*

MOF = Meta Object Facility, Core, v2.0

M3 (MOF)

Class

«instanceOf» «instanceOf» «instanceOf»

M2 (UML)

Attribute   Class   classifier   Instance

«instanceOf» «instanceOf» «instanceOf» «instanceOf»

M1 (User model)

Video
+title: String

«snapshot»

: Video
title = "2001: A Space Odyssey"
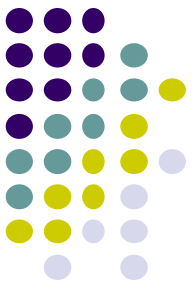
«instanceOf»

M0 (Run-time instances)

aVideo
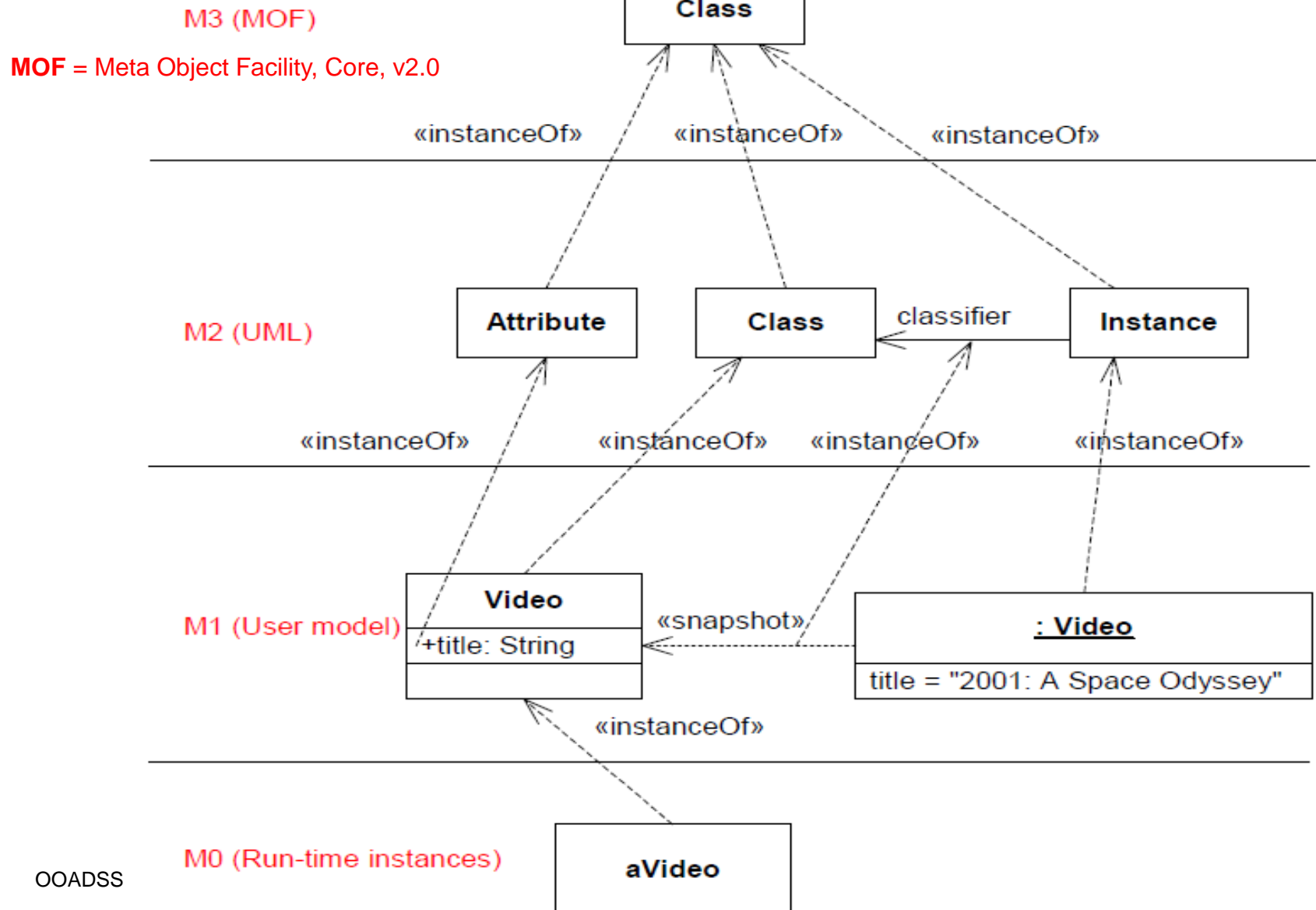
Source: OMG website

- **UML Description *:**
  A specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems.

- **UML Keywords *:**
  abstraction, action sequence, action state, activity graph, architecture, association, class diagram, collaboration diagram, component diagram, control flow, data flow, deployment diagram, execution, implementation, pins, procedure
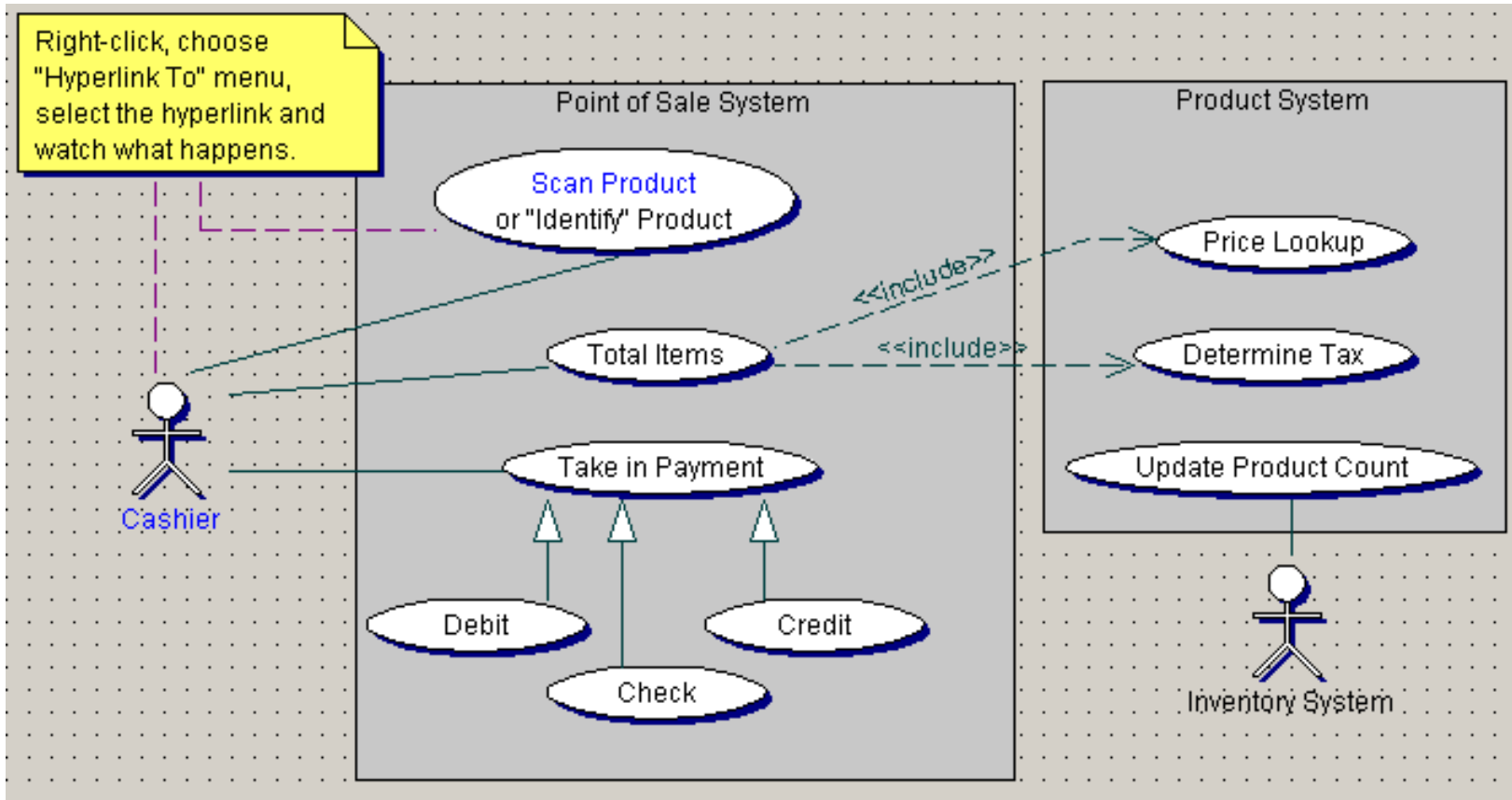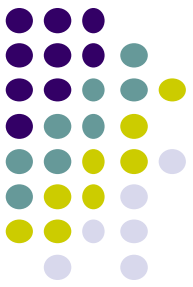
Good UML tools:

- Have to be UML compliant – UML diagram support
- Continuously synchronize models and code
- Provide team-work features - direct access to a Source Code Management (SCM) system
- Refactoring support - means for changing SW organization
- Design patterns support[1]
- Reverse engineering tools
- Documentation generation
- Integration with IDE's
- More…

[1] See *"Design Patterns"*, by Erich Gamma, Richard Helm, et al. (Addison-Wesley, 1995).

UML compliance: Use case diagrams

Right-click, choose "Hyperlink To" menu, select the hyperlink and watch what happens.

Point of Sale System

Scan Product or "Identify" Product

Total Items

<<include>>

<<include>>

Take in Payment

Debit

Credit

Check

Cashier

Product System

Price Lookup

Determine Tax

Update Product Count

Inventory System

Define system behavior without specifying how it works. Often used in requirements specification and analysis.

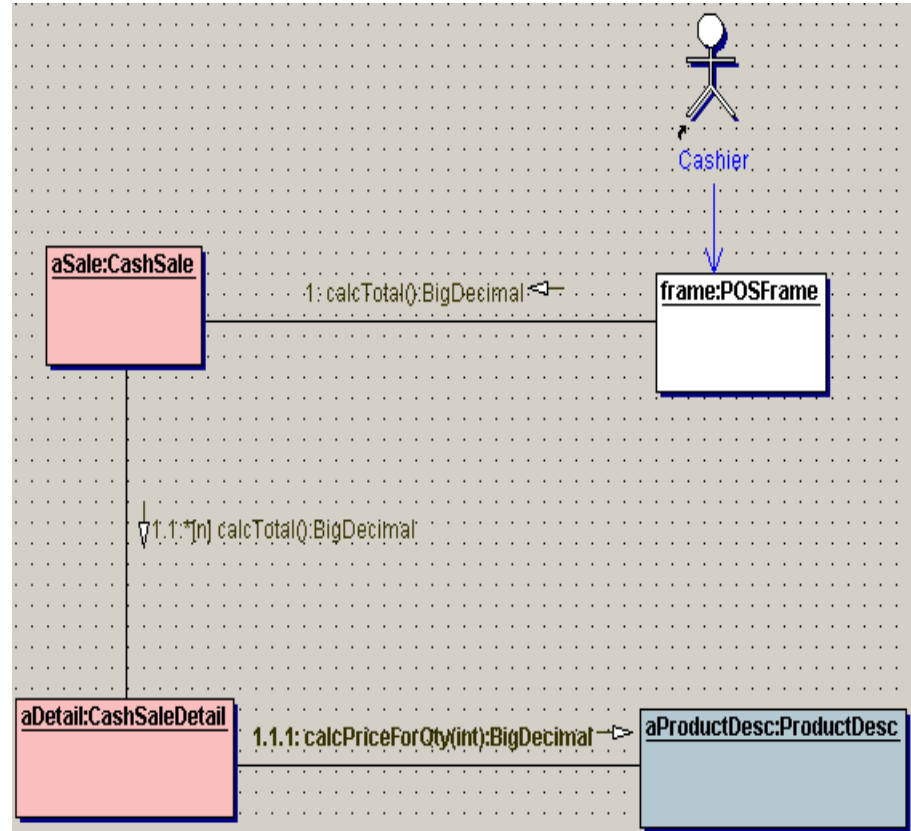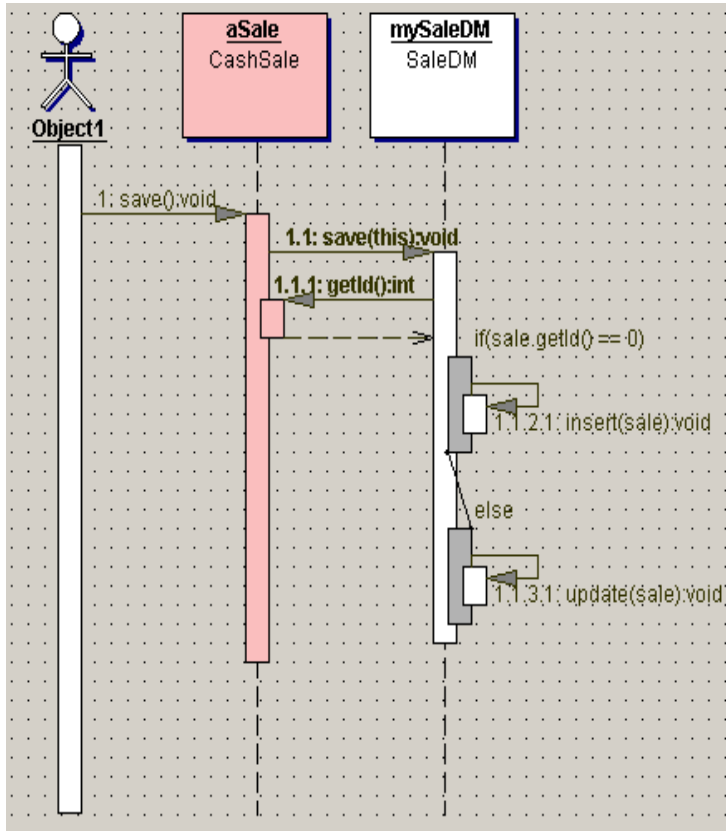**Diagrams from CashSales sample project of Together Architect®**

UML compliance: Class diagrams
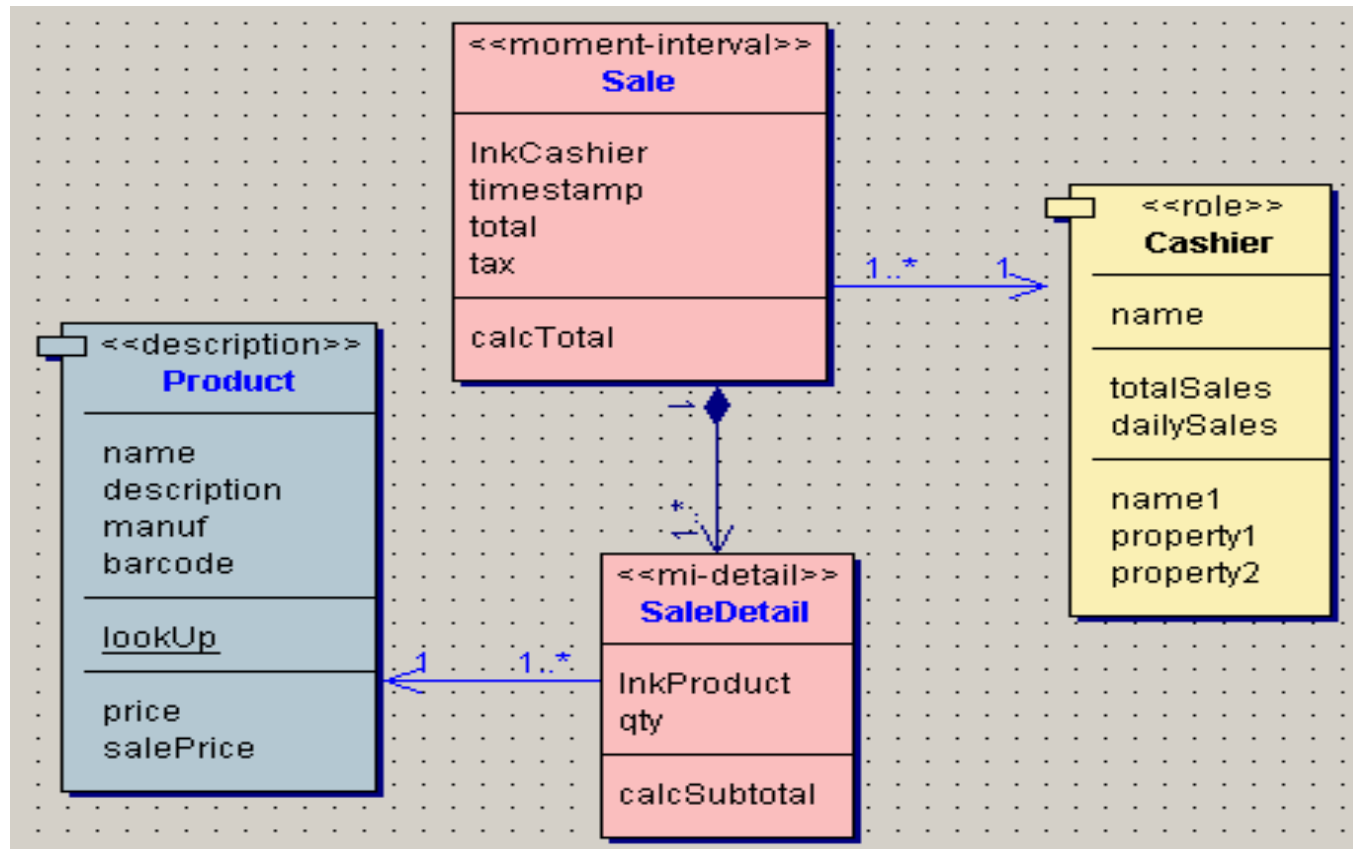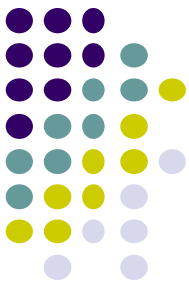
Defines the static organization or structure of software. Helps explain hierarchical and collaborative relationships between classes and objects.
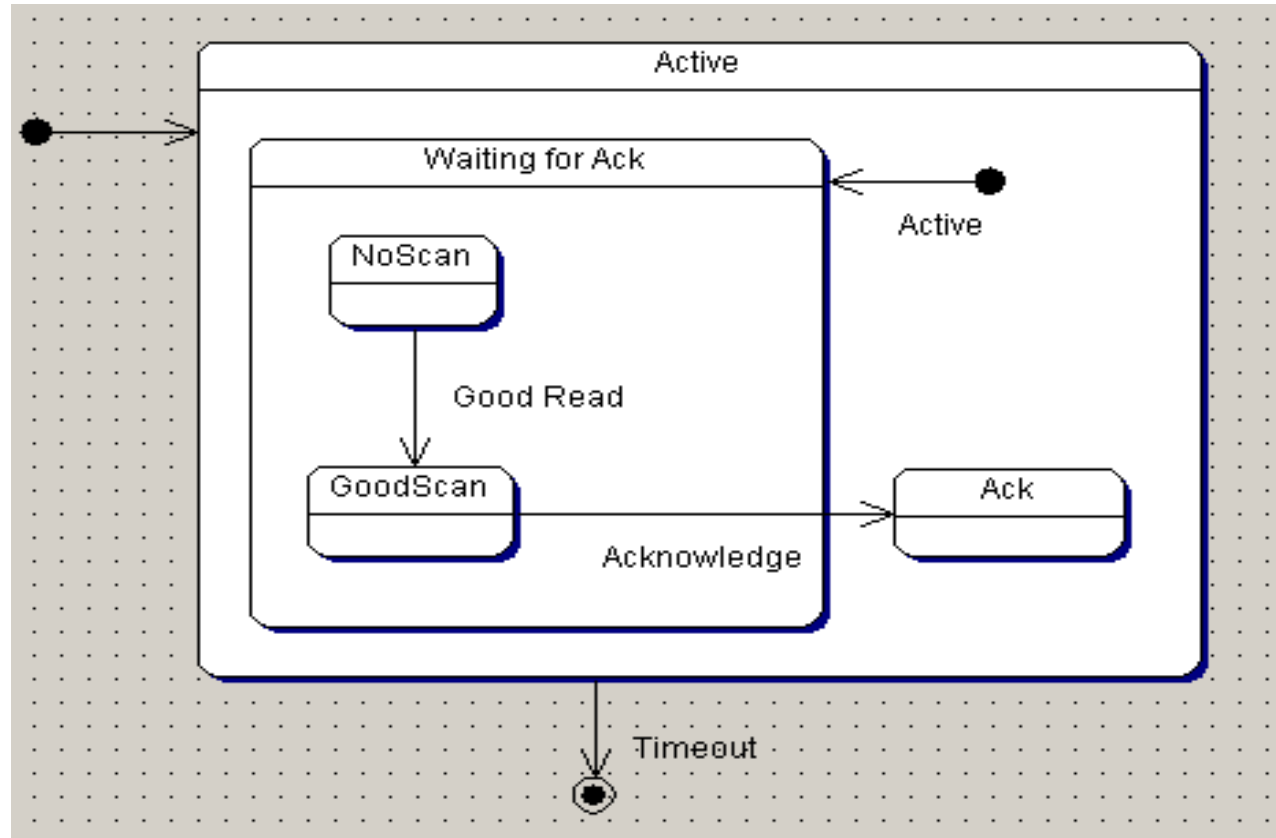
Diagrams from CashSales sample project of Together Architect®

UML compliance: Interaction (sequence/communication) diagrams

Represents active interaction and communi-cation between two or more objects.

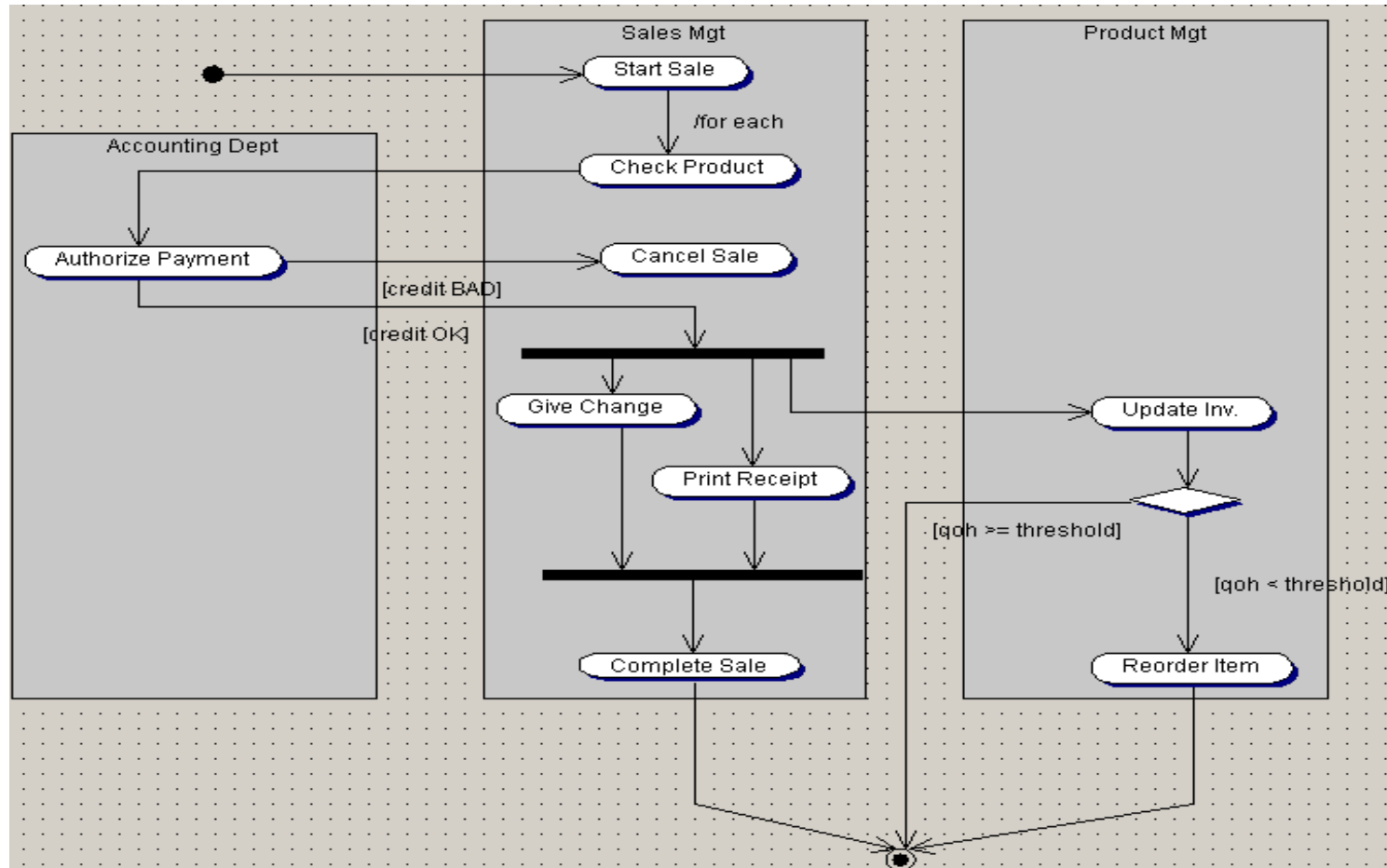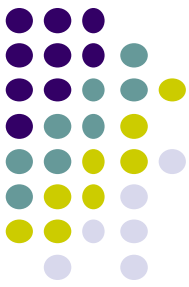**Diagrams from CashSales sample project of Together Architect®**

UML compliance: State diagrams

Define how states of objects or systems change as events occur.

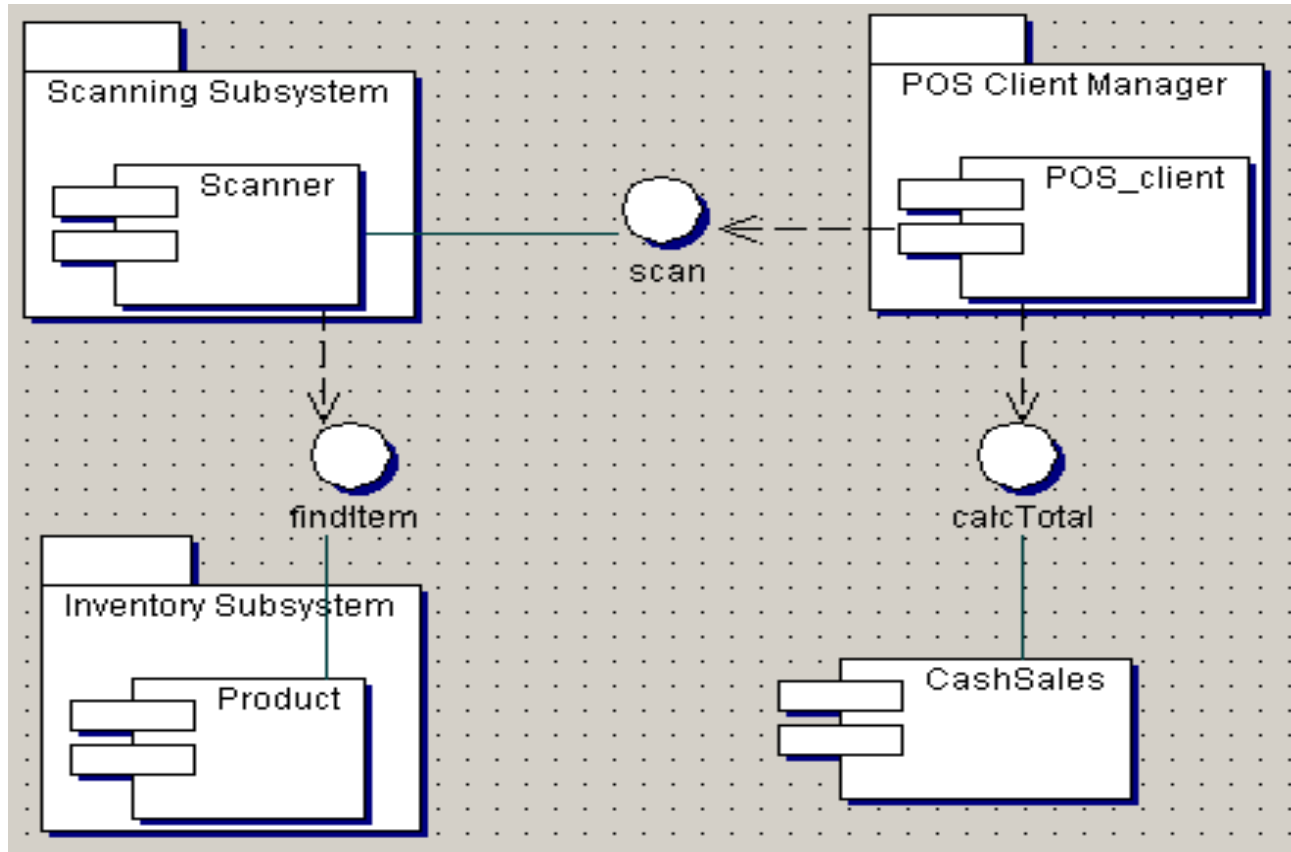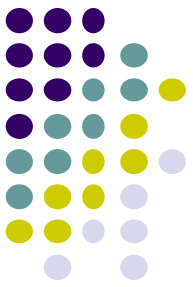**Diagrams from CashSales sample project of Together Architect®**

UML compliance: Activity diagrams

Describe sequencing of activities. Supports conditional and parallel behavior. Useful for analysis, workflow, parallel processes.
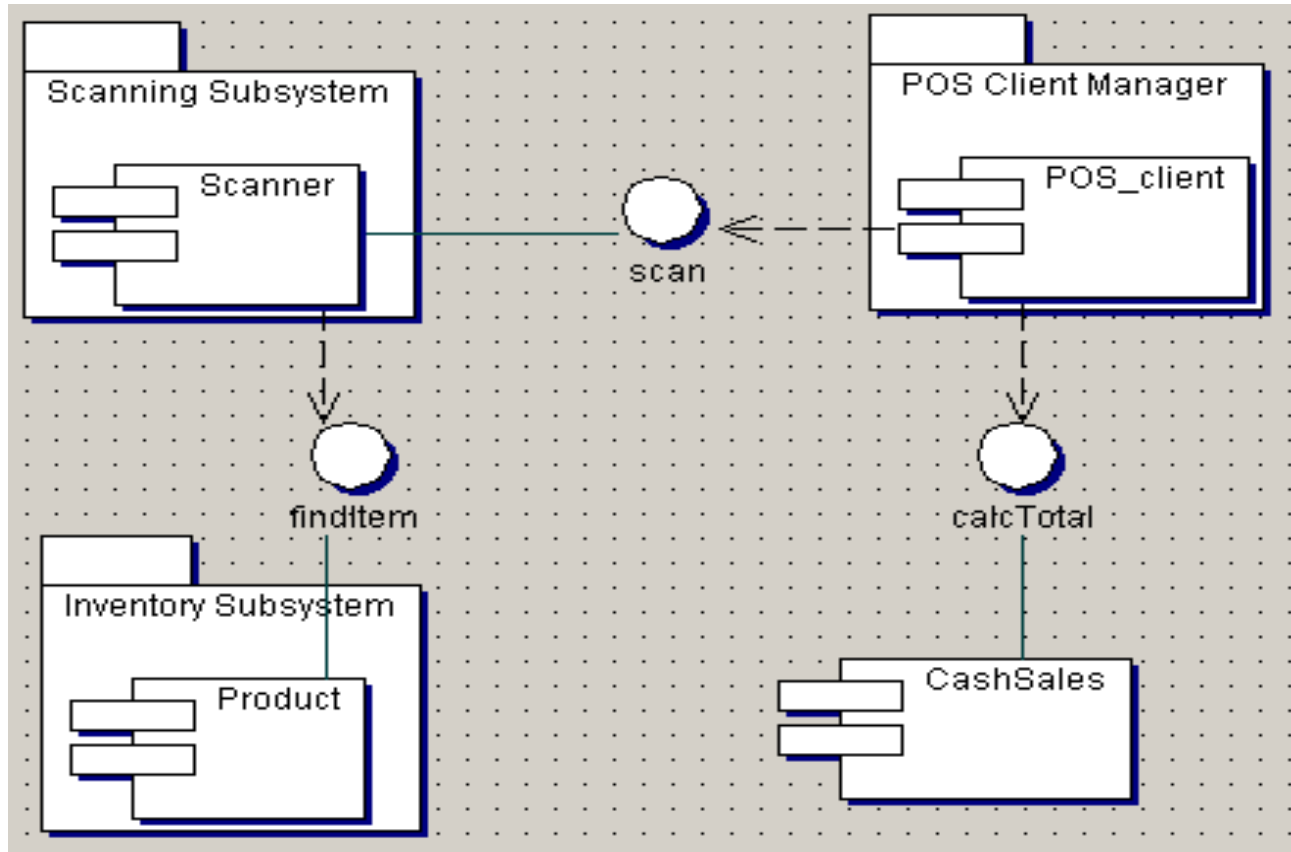
Diagrams from CashSales sample project of Together Architect®

UML compliance: Component diagrams

Define high-level configurations of the software system in one of its deployed state.

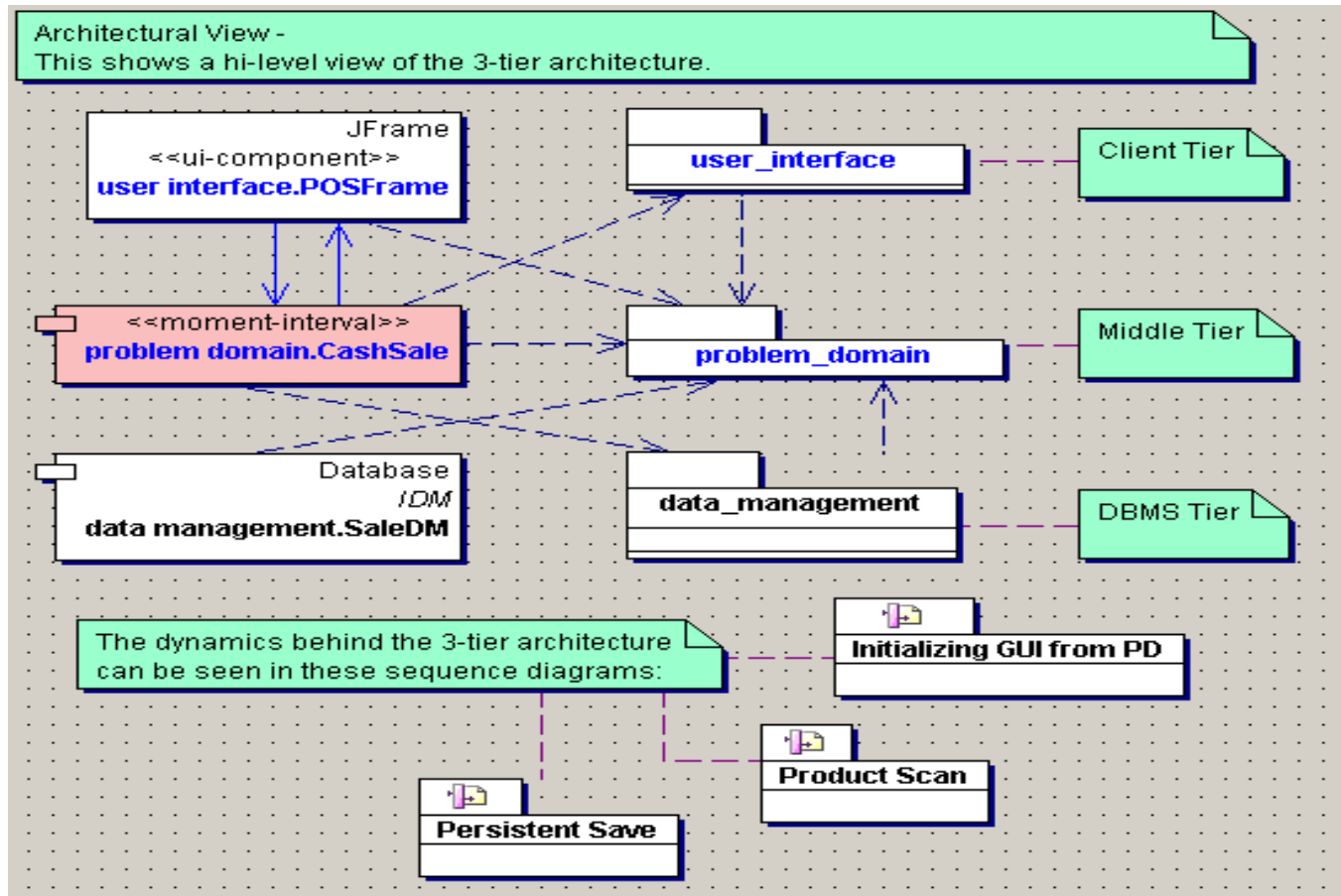**Diagrams from CashSales sample project of Together Architect®**

UML compliance: Component & deployment diagrams

Define high-level configurations of the software system in one of its deployed state.

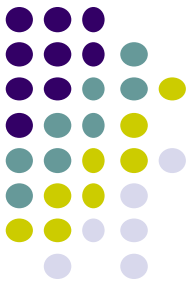**Diagrams from CashSales sample project of Together Architect®**

UML compliance: Package diagrams

Diagrams from CashSales sample project of Together Architect®

Organize groups of classes or use cases and describe overall behavior of the system.
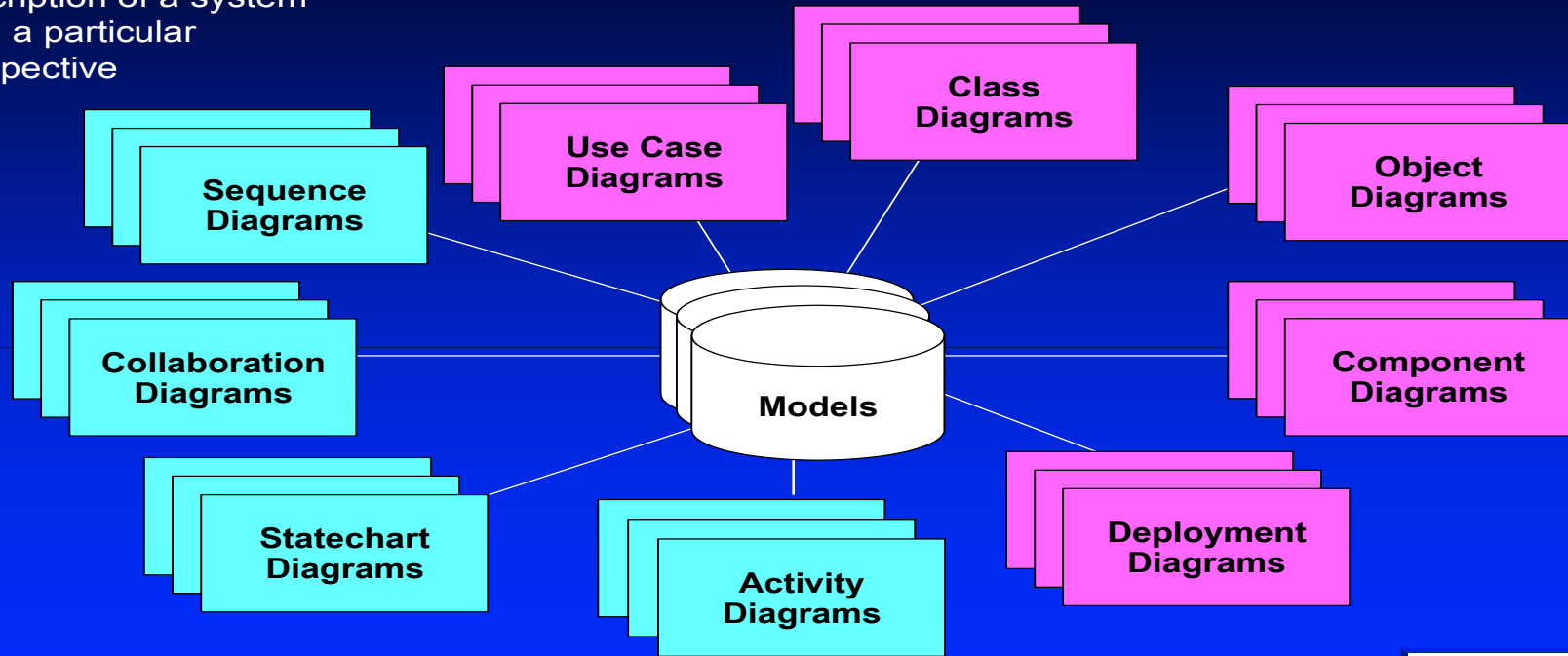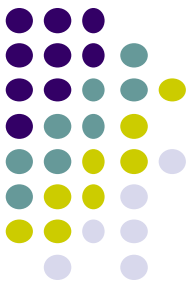
UML 1.5 models/diagrams

Models and Diagrams

A *model* is a complete description of a system from a particular perspective

- Sequence Diagrams
- Use Case Diagrams
- Class Diagrams
- Object Diagrams
- Collaboration Diagrams
- Models
- Component Diagrams
- Statechart Diagrams
- Activity Diagrams
- Deployment Diagrams

RATIONAL
SOFTWARE

UML Architectural Views

- ***user model view*** (use cases, scenario view)
- ***structural model view*** (static – class diagram)
- ***behavioral model view:*** dynamic – statechart/activity/interactions (sequence, collaboration) diagrams
- ***implementation model view*** (realization structure – component diagram)
- **environment model view** (deployment)

UML artifacts – packages, diagrams
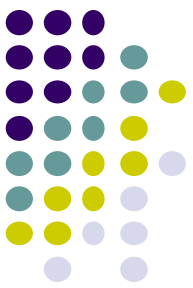
Models and views in UML 1.5

|	Use case view	Logical view	Implementation view	Process view	Deployment view
Use case diagram	YES				
Class diagram		YES			
Sequence diagram	YES	YES			
Collaboration diagram	YES	YES		YES	
Statechart diagram	YES	YES		YES	
Component diagram			YES	YES	YES
Deployment diagram					YES

Upgrading UML to Version 2.*
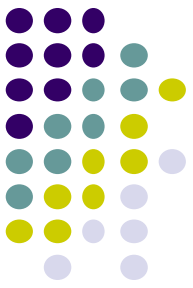
Consists of four parts:

- **UML 2.* Superstructure** - defines the six structure diagrams, three behavior diagrams, four interaction diagrams, and the elements that comprise them.

- **UML 2.* Infrastructure** - defines base classes that form the foundation not only for the UML 2.* superstructure.

- **UML 2.* Object Constraint Language (OCL)** - allows setting of pre- and post-conditions, invariants, and other conditions.

- **UML 2.* Diagram Interchange** - extends the UML metamodel with a supplementary package for graph-oriented information, allowing models to be exchanged or stored/retrieved and then displayed as they were originally.
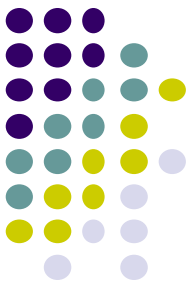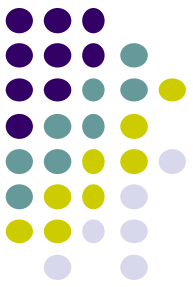
UML 2.* Changes

- **Highlights of the UML 2.* RFP:**
  - UML 1.x notions of interface and architecture enhanced to support and simplify support for standard component frameworks and architectures
  - Data flow modeling added
  - Many of the semantics of relationships clarified
  - In UML 1.x, sequence diagrams are too limited in their expressiveness and semantics and must be enhanced
  - Activity diagrams semantically separated from state machines
  - Clean up inconsistencies and errors in the UML 1.x specifications
  - Superstructure requirements to improve the ability and utility of the UML with the respect to architecture and scalability
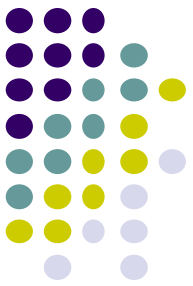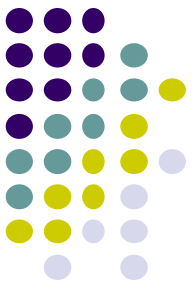
UML 2.* XMI Changes

- In UML 1.x, _XMI (XML Metadata Interchange)_ is a mechanism for exchanging UML models
  - This mechanism did not fully fulfill the goal of model interchange
- The UML 2.* solution extends the UML meta-model by a _supplementary package or graphic-oriented information_ while leaving the current UML meta-model fully intact.
- See the UML 2.5.1 Diagram Interchange spec at http://www.omg.org/technology/documents/modeling_spec_catalog.htm for additional information.
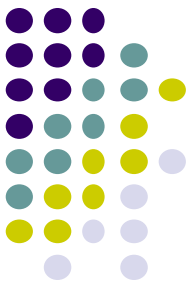
UML 2.* Highlight of Changes

- Introduced new concept of **Ports**

- **Composite Structure Classes & Diagrams** introduced

- **Class Diagrams** – the least changed

- Collaboration Diagram – renamed to **Communication Diagram**

- **Sequence Diagram –** nesting options

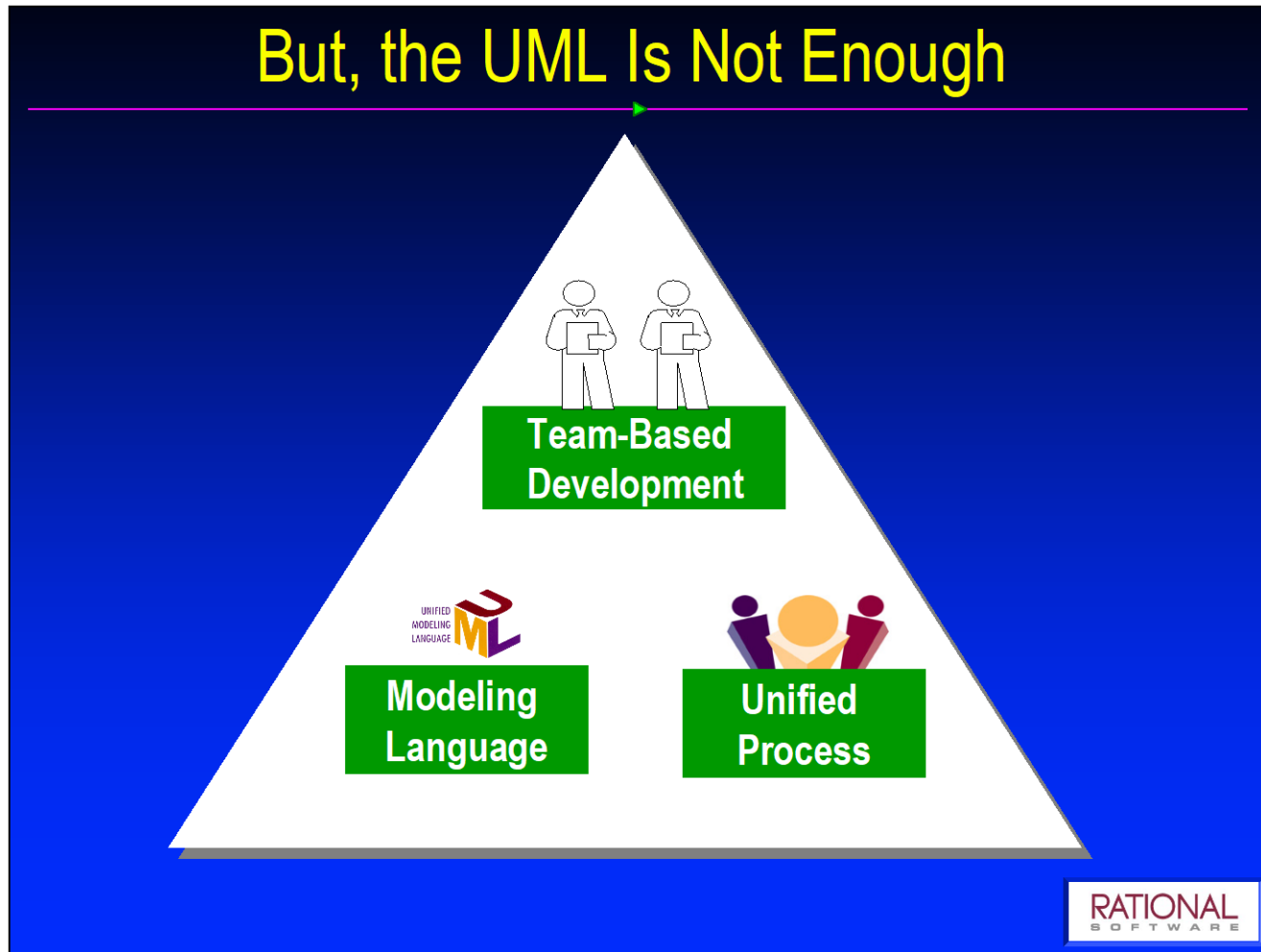- New diagram introduced **– Timing Diagram**

Cont.

- **Activity Diagrams** have the greatest number of changes of any of the UML diagrams

- **Use Case Diagrams** – added multiplicity and changes with extension points.

- **Package Diagram** now an official UML diagram.

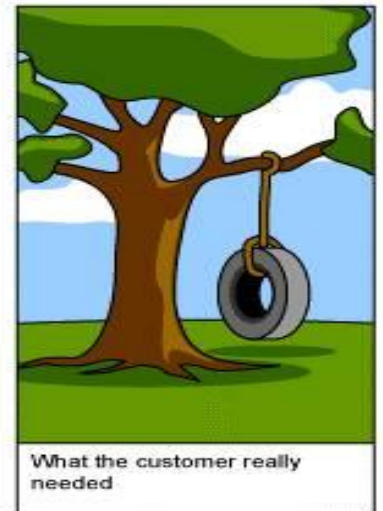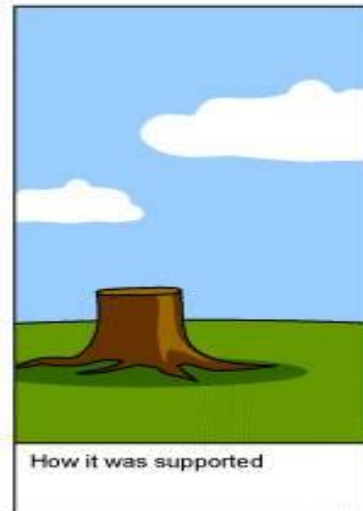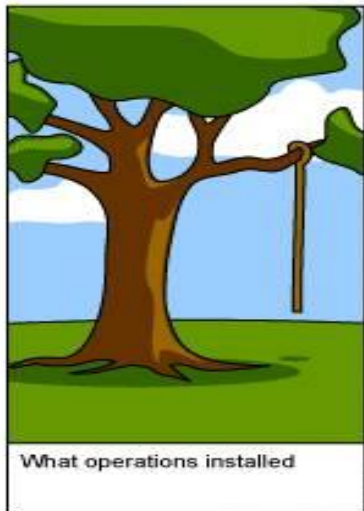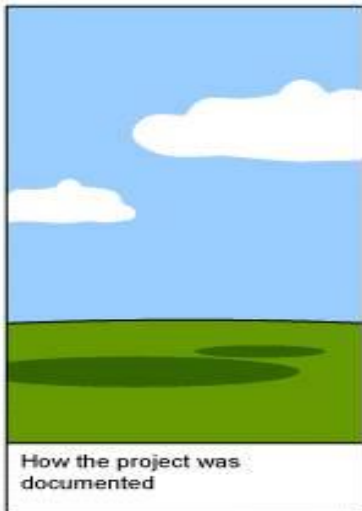- **Timing diagrams** – a new type of diagrams in UML 2.0

The UML triangle

But, the UML Is Not Enough

Team-Based Development

Modeling Language

Unified Process

RATIONAL
SOFTWARE

Instead UML:

How the customer explained it

How the Project Leader understood it

How the Analyst designed it

How the Programmer wrote it

How the Business Consultant described it

How the project was documented

What operations installed

How the customer was billed

How it was supported

What the customer really needed

Q & A

2. Object-Oriented Modelling with UML