

4+1 view model

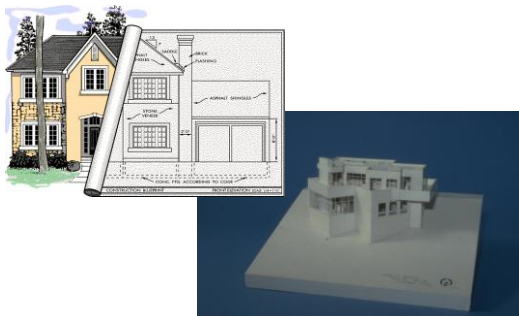
Topic # 17

Chapter 39 – Craig Larman

What is a Model?

- Models are often built in the context of business and IT systems in order to better understand existing or future systems.
- A **model** is an abstraction describing a subset of a system
- A useful model has the right level of detail and represents only what is important for the task in hand.
- However, a model never fully corresponds to reality.
- Modeling always means *emphasizing and omitting*:
 - emphasizing essential details and
 - omitting irrelevant ones.
 - Many things can be modelled: bridges, traffic flow, buildings, economic policy

Modeling a House



Why do we Need Models?

- **Communication** between all involved parties:
 - In order to build the right system, all parties think along the same lines. Important that everyone understands the same requirements, that developers understand these requirements, and that the decisions made can still be understood months later.
- **Visualization** of all facts:
 - All accumulated facts relevant to the system need to be presented in such a way that everyone concerned can understand them.
- **Verification** of facts in terms of completeness, consistency, and correctness: In particular, the clear depiction of interrelationships makes it possible to ask specific questions, and to answer them.

View

- The more information a model gives, the more complex and difficult it becomes.
- A system has many different aspects:
- **functional** (its static structure and dynamic interactions),
- **nonfunctional** (timing requirements, reliability, deployment, and so on),
- **organizational aspects** (work organization, mapping to code modules, and so on).
- A system description requires a number of views, where each view represents a projection of the complete system that shows a particular aspect.
- *Different views are formed of the objects. These views are interconnected in many ways.*

UML Views

- Structural classification
- Dynamic behaviour
- Physical layout
- Model management

UML views and diagrams

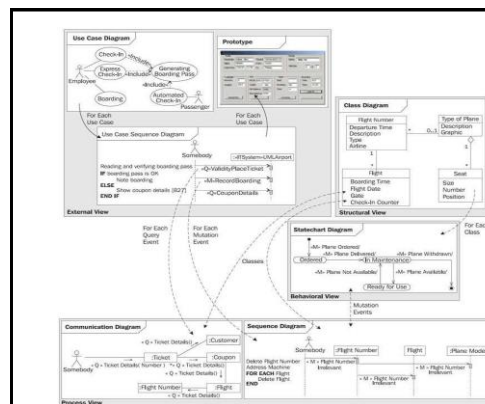
Major View	Diagram	Concepts
structural	class diagram	association, class, dependency, generalization, interface, realization
	internal structure	connector, interface, part, port, provided interface, role, required interface
	collaboration diagram	connector, collaboration, collaboration use, role
	component diagram	component, dependency, port, provided interface, realization, required interface, subsystem
	use case diagram	actor, association, extend, include, use case, generalization

UML views and diagrams cont.

Major View	Diagram	Concepts
dynamic	state machine diagram	completion transition, do activity, effect, event, region, state, transition, trigger
	activity diagram	action, activity, control flow, control node, data flow, exception, expansion region, fork, join, object node, pin
	sequence diagram	occurrence specification, execution specification, interaction, lifeline, message, signal
	communication diagram	collaboration, guard condition, message, role, sequence number

UML views and diagrams cont.

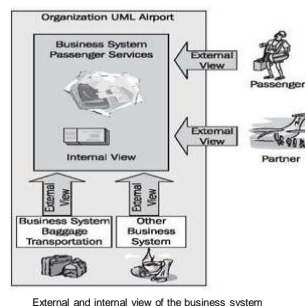
Major View	Diagram	Concepts
physical	deployment diagram	artifact, dependency, manifestation, node
model management	package diagram	import, model, package
	package diagram	constraint, profile, stereotype, tagged value

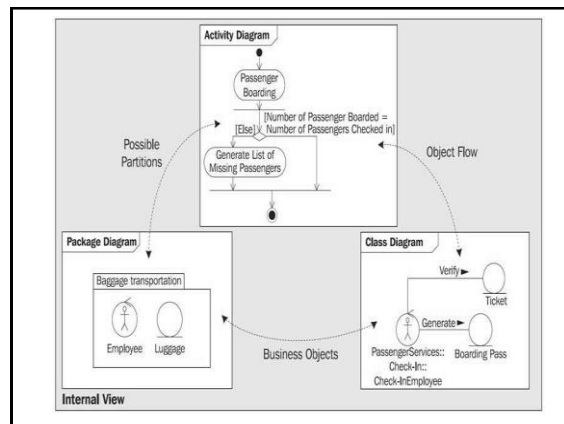
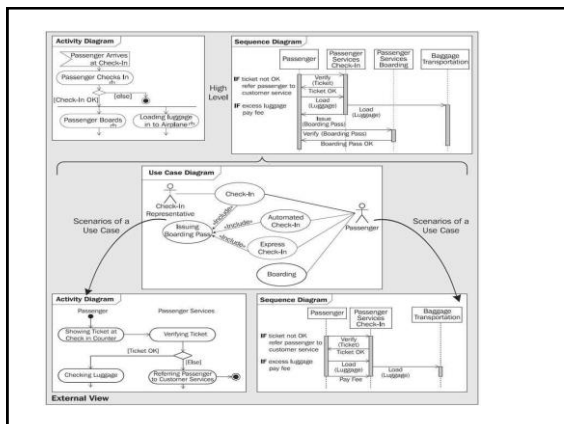


One Model—Two Views

- External View
- Internal View

External and Internal View





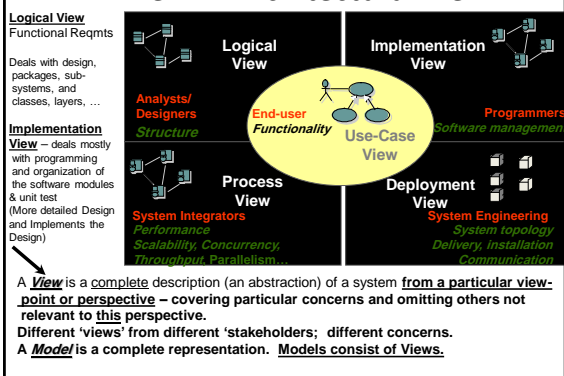
What is View Model ?

- A view model in systems engineering or software engineering is a framework.
- It defines a coherent **set of views** to be used in the construction of a system architecture or software architecture.
- A view is a representation of a whole system from the perspective of a related set of concerns.
- Viewpoint modeling has become an effective approach for dealing with the inherent complexity of large distributed systems.

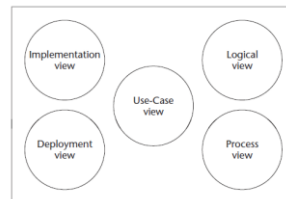
Intent of 4+1 view model

- To come up with a mechanism to separate the different aspects of a software system into different views of the system.
- But why???? -> Different stakeholders always have different interest in a software system.
- DEVELOPERS – Aspects of Systems like classes
- SYSTEM ADMINISTRATOR – Deployment, hardware and network configuration.
- Similar points can be made for Testers, Project Managers and Customers.

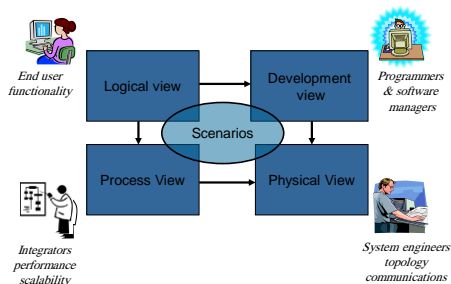
**The 4+1 Architectural View



The Views in UML



4+1 View Model



Diagrams

- Each view requires a number of diagrams that contain information emphasizing a particular aspect of the system. A slight overlap does exist, so a diagram can actually be a part of more than one view.
- By looking at the system from different views, it is possible to concentrate on one aspect of the system at a time.
- A diagram in a particular view needs to be simple enough to communicate information clearly, yet coherent with the other diagrams and views so that the complete picture of the system is described by all the views put together.

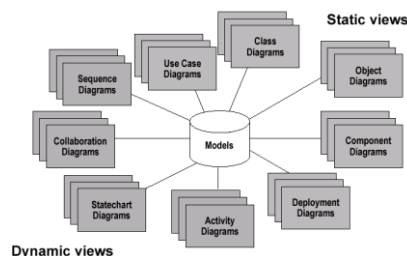
Diagrams

- A system model typically has several diagrams of varying types, depending on the goal for the model.
- A diagram is part of a specific view, and when it is drawn, it is usually allocated to a view.
- Some diagram types can be part of several views, depending on the contents of the diagram.

UML Models, Views, Diagrams

- UML is a multi-diagrammatic language
 - Each diagram is a view into a model
 - Diagram presented from the aspect of a particular stakeholder
 - Provides a partial representation of the system
 - Is semantically consistent with other views

Models, Views, Diagrams



Use Case View

- The use-case view describes the functionality the system should deliver, as perceived by external actors.
- An actor interacts with the system; the actor can be a user or another system.
- The use-case view is used by customers, designers, developers, and testers; it is described in use-case diagrams, sometimes with support from activity diagrams.
- The desired usage of the system is described as a number of use cases in the use-case view, where a use case is a generic description of a function requested.

Central View

- The use-case view is central, because its contents drive the development of the other views.
- The final goal of the system is to provide the functionality described in this view—along with some nonfunctional properties.
- Hence, this view affects all the others.
- This view is also used to validate the system and finally to verify the functioning of the system by testing the use-case view with the customers (asking, “Is this what you want?”) and against the finished system (asking, “Does the system work as specified?”).

Logical View

- The logical view describes how the system’s functionality is provided.
- It is mainly for designers and developers. In contrast to the use-case view, the logical view looks inside the system. It describes both the static structure (classes, objects, and relationships) and the dynamic collaborations that occur when the objects. The static structure is described in class and object diagrams.
- The dynamic modeling is described in state machines, and interaction and activity diagrams.

Implementation view

- The implementation view describes the main modules and their dependencies.
- It is mainly for developers and consists of the main software artifacts. The artifacts include different types of code modules shown with their structure and dependencies.
- Additional information about the components, such as resource allocation (responsibility for a component) or other administrative information, such as a progress report for the development work, can also be added.
- The implementation view will likely require the use of extensions for a specific execution environment.

Process View

- The process view deals with the division of the system into processes and processors. This aspect allows for efficient resource usage, parallel execution, and the handling of asynchronous events from the environment.
- This view must also deal with the communication and synchronization of these threads.
- This view provides critical information for developers and integrators of the system.
- The view consists of dynamic diagrams (state machines, and interaction and activity diagrams) and implementation diagrams (interaction and deployment diagrams). A timing diagram also provides a specialized tool for the process view.
- A timing diagram provides a way to show the current status of an object in terms of time.

Deployment view

- Finally, the deployment view shows the physical deployment of the system, such as the computers and devices (nodes) and how they connect to each other.
- The various execution environments within the processors can be specified as well.
- The deployment view is used by developers, integrators, and testers and is represented by the deployment diagram.
- This view also includes a mapping that shows how the artifacts are deployed in the physical architecture, for example, which programs or objects execute on each respective computer.

4+1 View model came before UML

- It's important to remember the 4 + 1 approach was put forward two years before the first introduction of UML.
- UML is how most enterprise architectures are modeled and the 4 + 1 approach still plays a relevance to UML today.
- UML has 13 different types of diagrams - each diagram type can be categorized into one of the 4 + 1 views.
- UML is 4 + 1 friendly!

Is it important?

- **It makes modeling easier.**
- **Better organization with better separation of concern.**
- **The 4 + 1 approach provides a way for architects to be able to prioritize modeling concerns.**
- **The 4 + 1 approach makes it possible for stakeholders to get the parts of the model that are relevant to them.**

Views

Use Case View

- Use Case Analysis is a technique to capture business process from user's perspective.
- Static aspects in use case diagrams; Dynamic aspects in interaction (statechart and activity) diagrams.

Design View

- Encompasses classes, interfaces, and collaborations that define the vocabulary of a system. Supports functional requirements of the system.
- Static aspects in class and object diagrams; Dynamic aspects in interaction diagrams.

Process View

- Encompasses the threads and processes defining concurrency and synchronization. Addresses performance, scalability, and throughput.
- Static and dynamic aspects captured as in design view; emphasis on active classes.

Implementation View

- Encompasses components and files used to assemble and release a physical system.
- Addresses configuration management.
- Static aspects in component diagrams; Dynamic aspects in interaction diagrams.

Deployment View

- Encompasses the nodes that form the system hardware topology. Addresses distribution, delivery, and installation.
- Static aspects in deployment diagrams; Dynamic aspects in interaction diagrams..

Book Reference and Reading

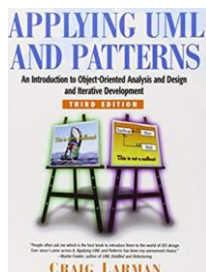
- Chapter 39 – Applying UML & Patterns – Craig Larman

Chapter 39. Documenting Architecture: UML & the N+1 View Model

"They have computers, and they may have other weapons of mass destruction.
USA government official"

Objectives

- Create useful architecture documentation based on the N+1 (or 4+1) view model.
- Apply various UML diagram types.



END OF TOPIC 16

-COMING UP!!!!!!

- Homogenization of system
- Implementation Diagrams