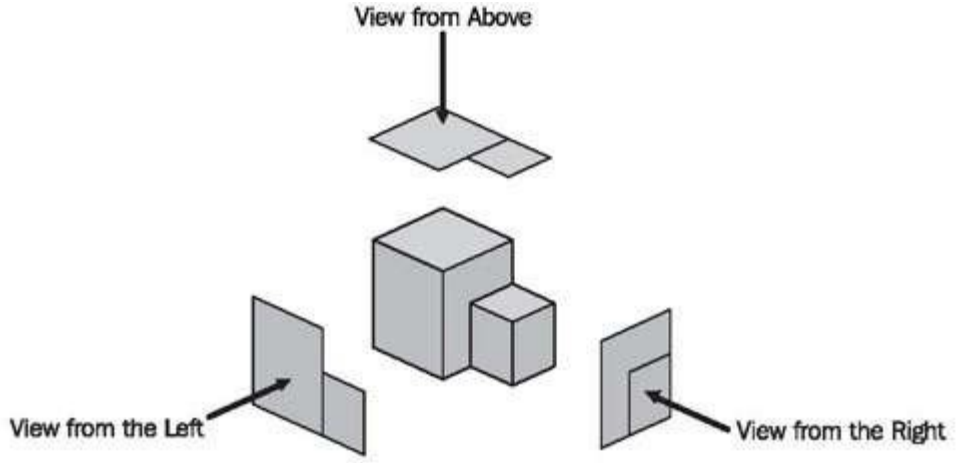
What is a Model?

- Models are often built in the context of business and IT systems in order to better understand existing or future systems.
- However, a model never fully corresponds to reality.
- Modeling always means *emphasizing* and *omitting*: emphasizing essential details and omitting irrelevant ones.

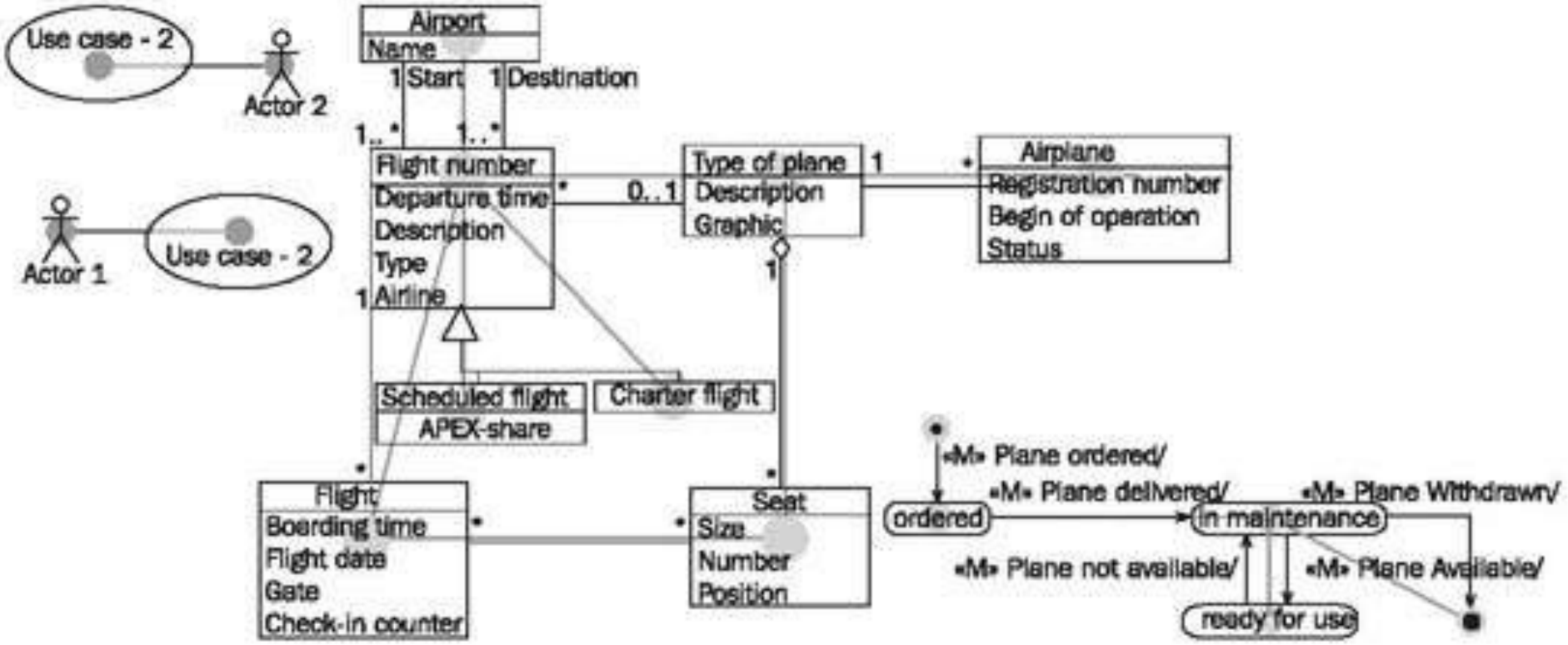
Views

- The more information a model gives, the more complex and difficult it becomes.
- A map of Europe, for example, that simultaneously contains political, geological, demographic, and transportation-related information is hardly legible. The solution to this problem is to convey the different types of information on individual maps.
- *Different views* are formed of the objects. These views are interconnected in many ways.



Diagrams as Views

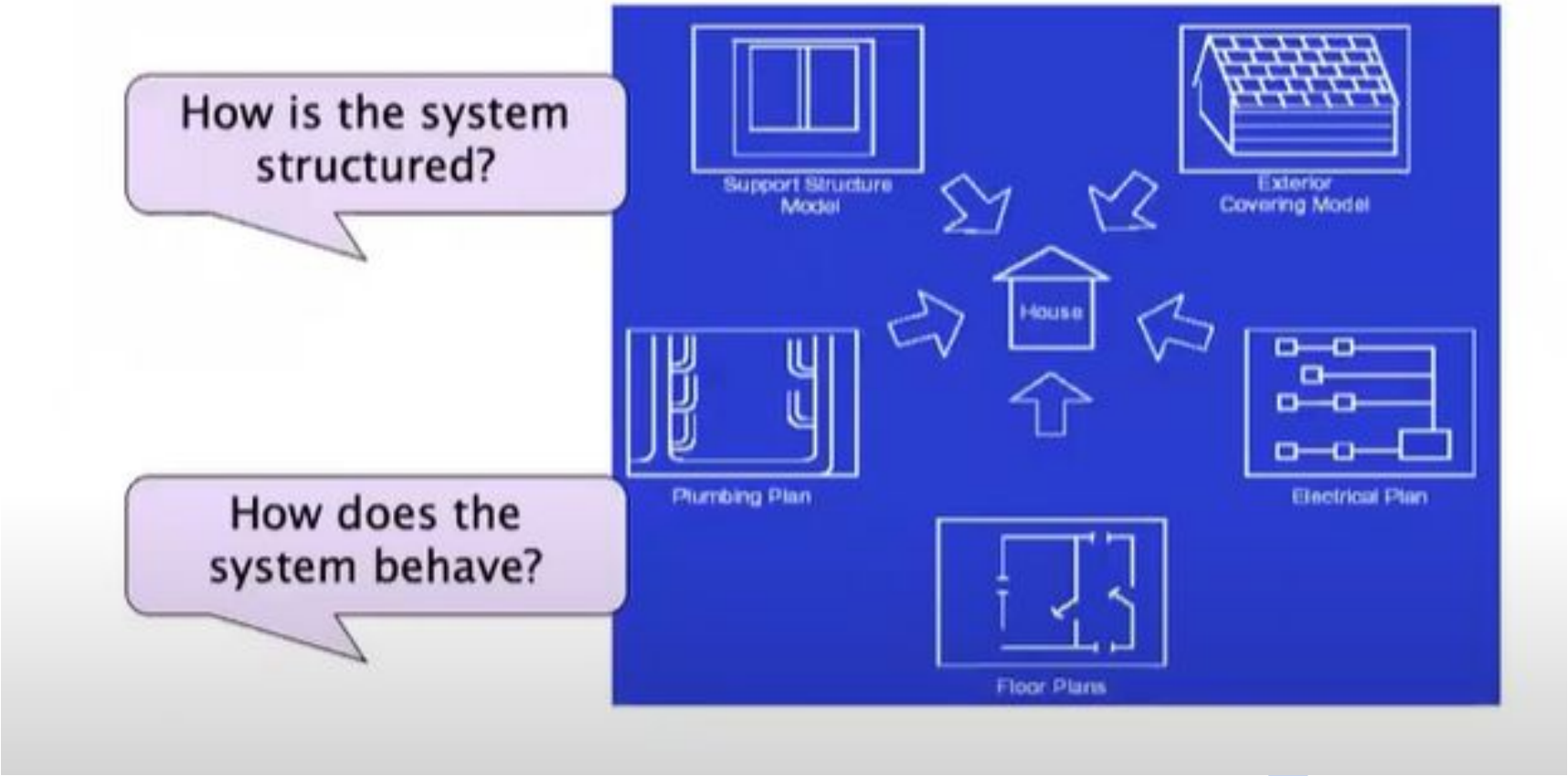
- Each particular UML diagram corresponds to one **view** of a model of a system.



Introduction to Views

- Dictionary Meaning ~ manner of looking at something
- Why (multiple) view ?
  - For better understanding and managing , multi dimensional view must be taken for any complex entity , multiple parameters / specialties involved. (bcos of its complex nature, it can't be described in 1 dimensional view)
- For example, In civil what are the views of a building…
  - Room layout; Front Elevation drawing; 3D view of building / room; Electrical diagram; Plumbing diagram; Security alarm diagram; AC duct diagram etc…etc…

Systems have multiple aspects



Definition of SW View

- Software architecture descriptions are commonly organized into views,
  - Each view addresses a set of system concerns, following the conventions of its *viewpoint*.
    - Viewpoint - A position or direction from which something is observed or considered;
    - View – Details or full specification considered from that viewpoint

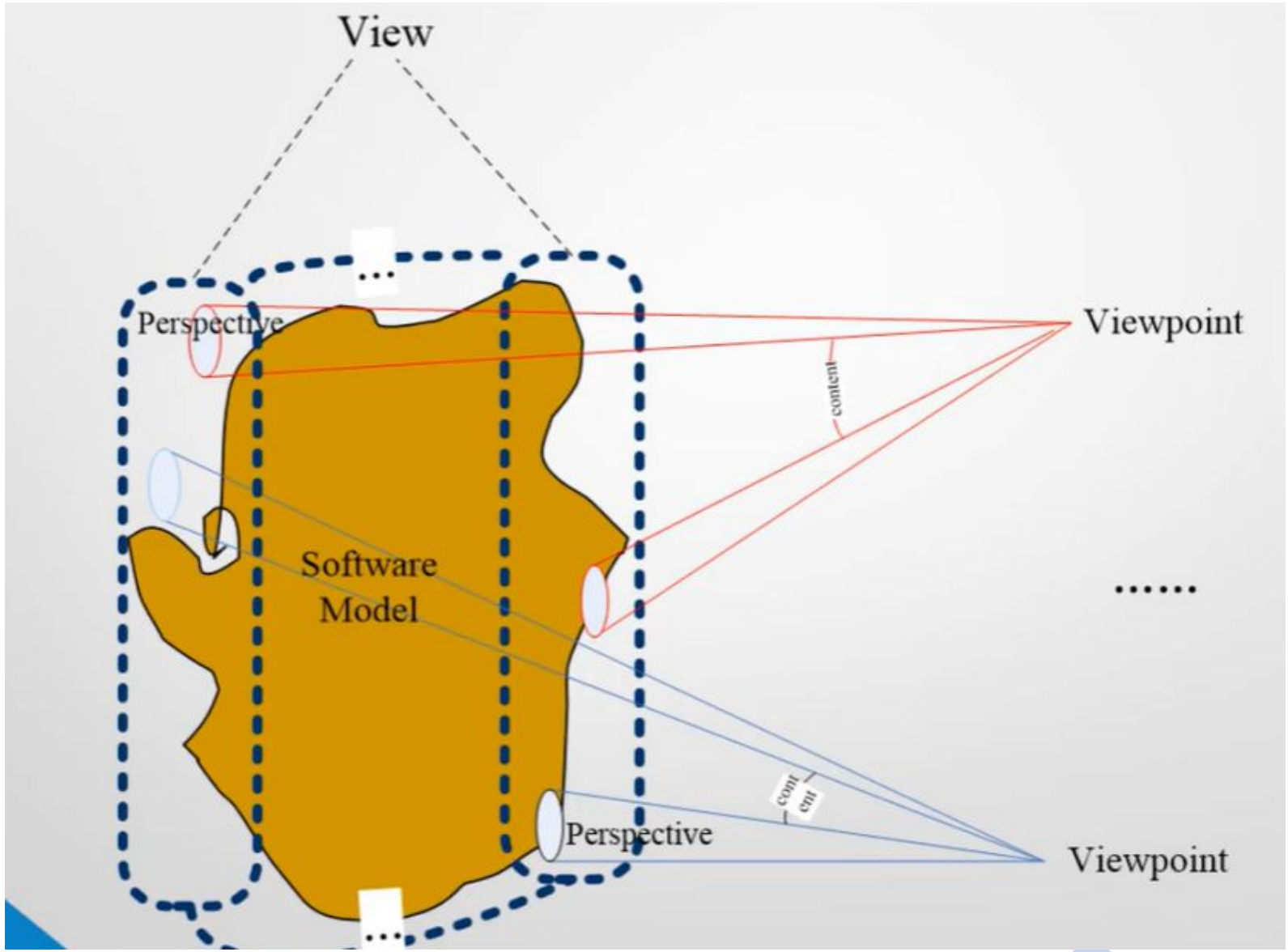
So, a view of a system is a representation of the system from the perspective of a viewpoint.

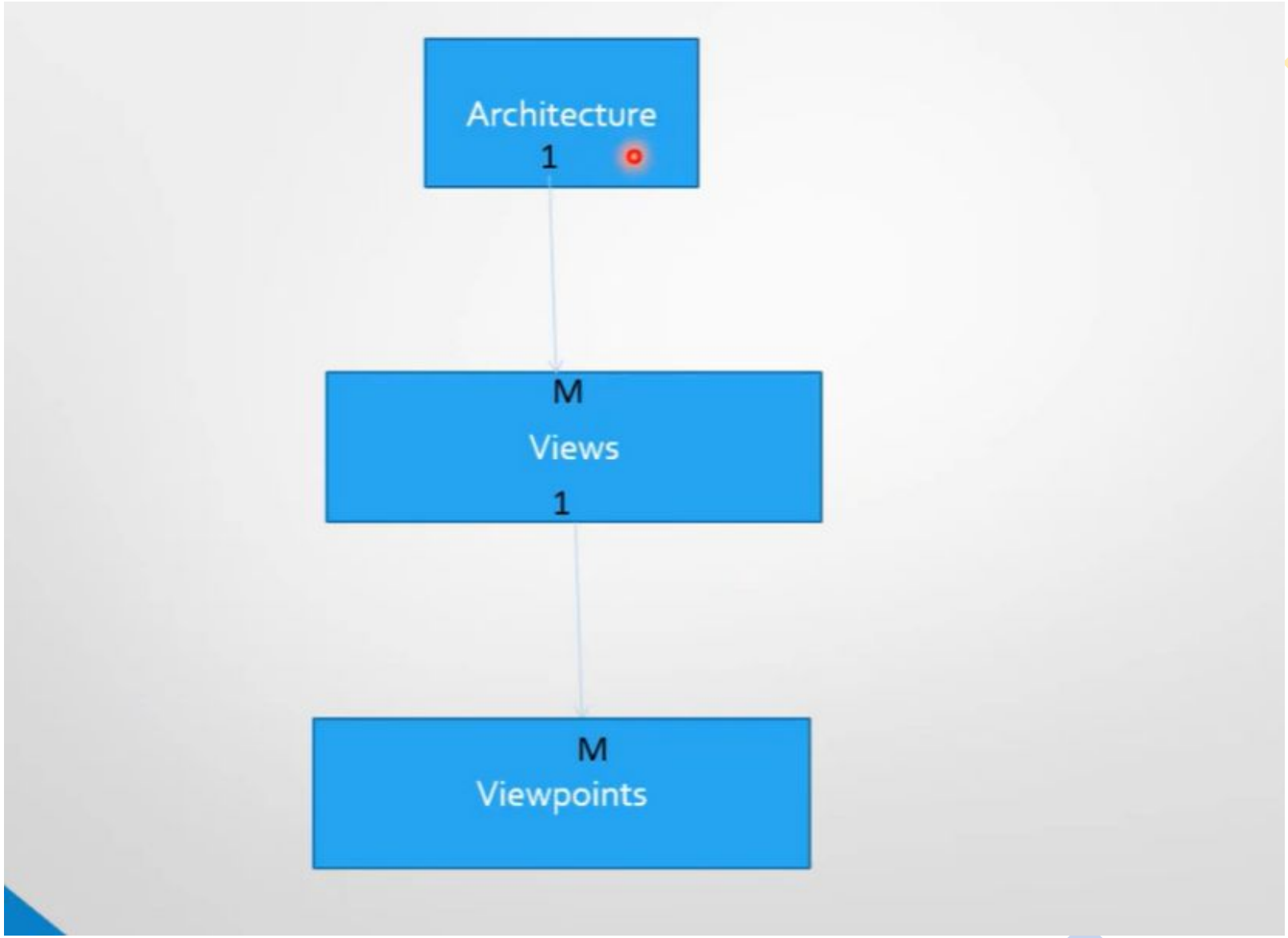
But why????

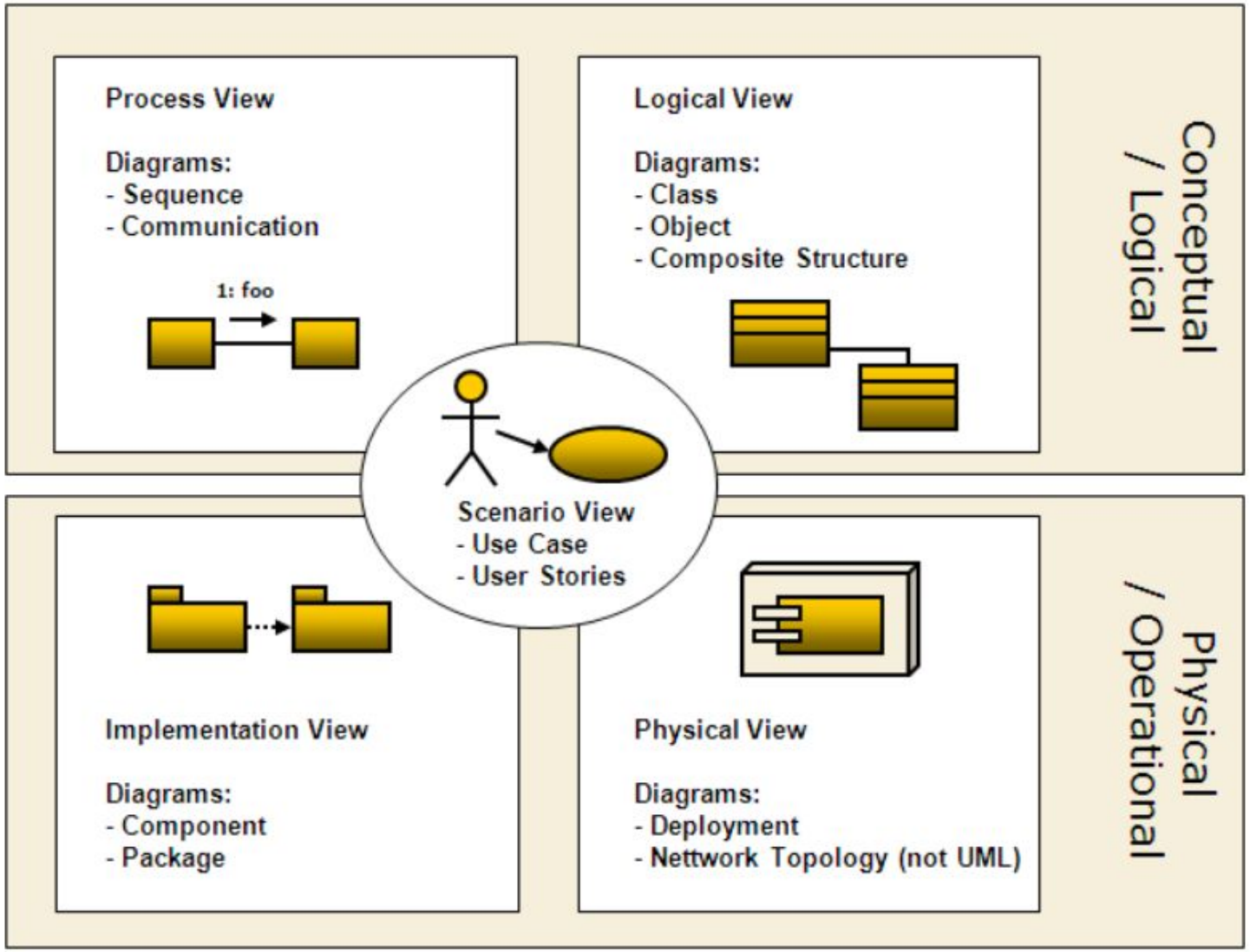
Different stakeholders always have different interest in a software system

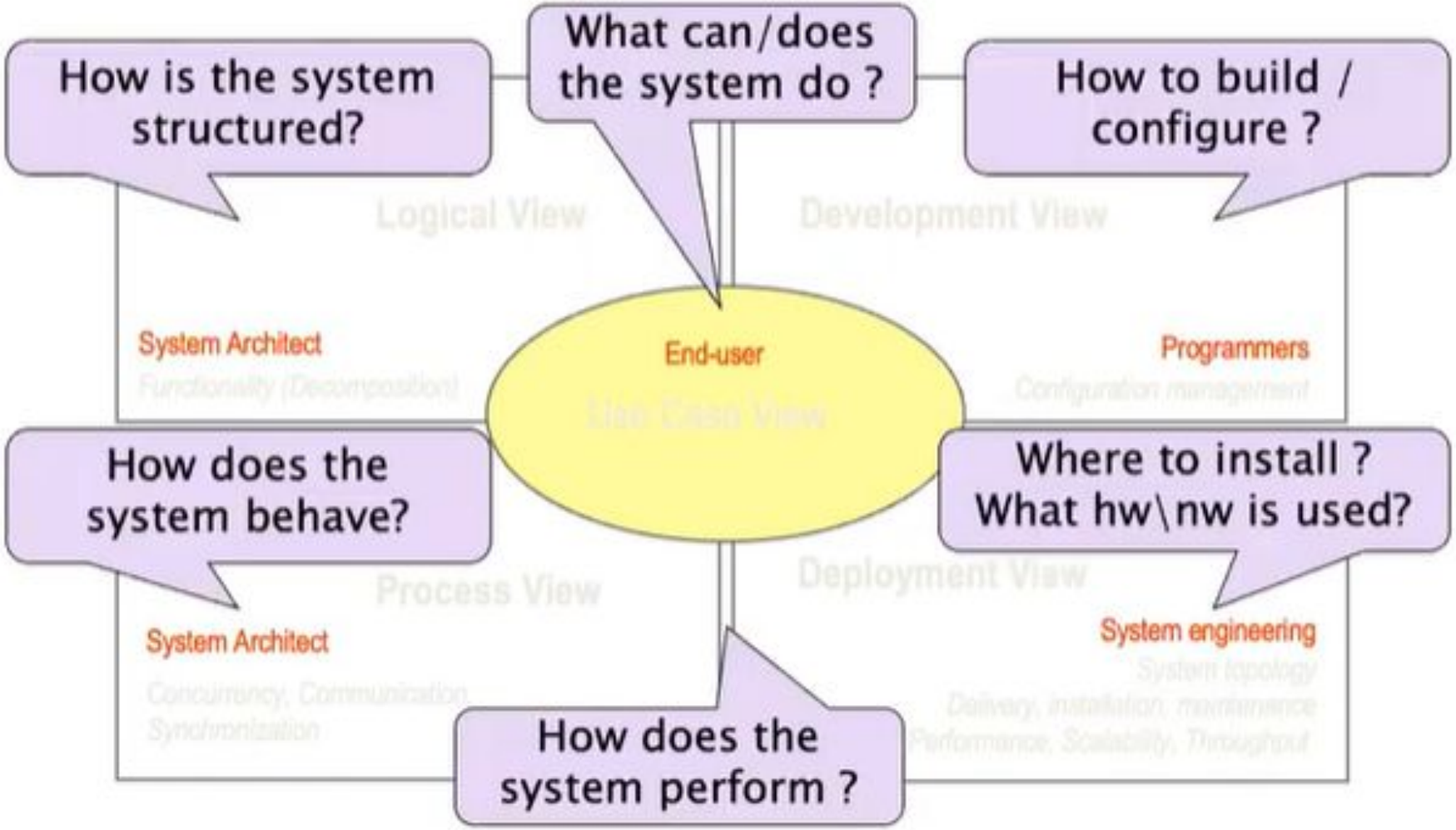
4+1 view model

- It was introduced by Philippe Kruchten in 1995
- The views are used to describe the system from the viewpoint of different stakeholders , such as end users , developers and project managers


View
Perspective
Viewpoint
content
Software
Model
.....
Perspective
Viewpoint


Architecture
1
M
Views
1
M
Viewpoints


Process View
Diagrams:
- Sequence
- Communication
1: foo
Logical View
Diagrams:
- Class
- Object
- Composite Structure
Conceptual / Logical
Scenario View
- Use Case
- User Stories
Implementation View
Diagrams:
- Component
- Package
Physical View
Diagrams:
- Deployment
- Nettwork Topology (not UML)
Physical / Operational


How is the system structured?
What can/does the system do ?
How to build / configure ?
Logical View
Development View
System Architect
End-user
Programmers
How does the system behave?
Where to install ? What hw\nw is used?
Process View
Deployment View
System Architect
System engineering
How does the system perform ?

The Logical View/Structural view

- (Object-Oriented Decomposition)
- The system is decomposed into a set of key abstractions, taken (mostly) from the problem domain, in the form of objects or object classes.
- **Viewer / Stake holder**: End user , Analyst , Designer
- **Considers**: supports the functional requirements—what the system should provide in terms of services to its users.
- **UML diagrams**: Class, State, Object, sequence, Communication Diagram

The Process View/ Behavior view

- (Process Decomposition)
- A process is a grouping of tasks that form an executable unit.
- Shows how , at runtime , the system is composed of interacting objects
- Shows the processes and how they communicate with each other
- **Considers :** The process view takes into account some non-functional requirements,
- such as performance and availability. It addresses issues of concurrency and distribution, of system's integrity, of fault-tolerance, and how the main abstractions from the logical view fit within the process architecture—on which thread of control is an operation for an object actually executed.

The Process View

- **Viewer / Stake holder:** Integrators & developers
- **UML diagrams**: Activity Diagram

The Development View/ Implementation view

- Subsystem decomposition
- Shows how the software is decomposed for the development
- It gives a building block view of the system
- **Considers :** The development architecture focuses on the actual software module organization on the software development environment. The software is packaged in small chunks—program libraries, or subsystems— that can be developed by one or a small number of developers.
- The subsystems are organized in a hierarchy of layers, each layer providing a narrow and well-defined interface to the layers above it.

The Development View

- **Viewer / Stake holder :** Programmer and software project managers
- **UML diagrams** : Component, Package diagram

The Physical View/Deployment view

- (Mapping the software to the hardware)
- Shows the installation, configuration and deployment of software application
- **Considers :** Nonfunctional requirement regarding to underlying hardware(Topology , communication)
- **Viewer / Stake holder:** System engineer, operators, system administrators and system installers
- **UML diagram :** Deployment diagram

The Physical View

- Components are either processing nodes (servers, virtual machines, docker containers, serverless configurations, etc.) or networking channels (routers, firewalls, proxies, load balancers, etc.) and the diagram should illustrate which nodes host which processes from the process view. This allows us to start considering compute and network capacity, as well as latency and other performance considerations.
- The view shows the system execution environment

Scenarios

- (Putting it all together)
- The scenarios are in some sense an abstraction of the most important requirements.
- This view is redundant with the other ones (hence the “+1”)
- The elements in the four views are shown to work together seamlessly by the use of a small set of important scenarios

Scenarios

- **Considers :** System Consistency and validity
- **Viewer / Stake holder:**All the views of their views and evaluators
- **UML diagram :** Use case diagram

Conceptual / Logical

Physical / Operational

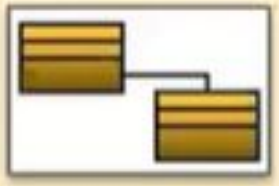
Functional

Non-functional

Logical / Structural view

Perspective: Analysts, Designers
Stage: Requirement analysis
Focus: Object oriented decomposition
Concerns: Functionality
Artefacts:

- Class diagram
- Object diagram
- Composite structure diagram



Implementation / Developer view

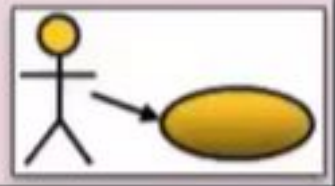
Perspective: Developers, Proj. mngs.
Stage: Design
Focus: Subsystem decomposition
Concerns: Software management
Artefacts:

- Component diagram
- Package diagram

Use Case/Scenario view

Perspective: End users
Stage: Putting it alltogether
Concerns: Understandability, usability
Focus: Feature decomposition
Artefacts:

- Use-case diagram
- User stories



Process / Behaviour view

Perspective: System Integrators
Stage: Design
Focus: Process decomposition
Concerns: Performance, scalability, throughput
Artefacts:

- Sequence diagram
- Communication diagram
- Activity diagram
- State (machine) diagram
- Interaction overview diagram
- Timing diagram

Deployment / Physical view

Perspective: System Engineers
Stage: Design
Focus: Map software to hardware
Concerns: System topology, delivery, installation, communication
Artefacts:

- Deployment diagram
- Network topology (not UML)

Why is it called 4+1 instead of 5?

- The **use case view** has a special significance as it details the high-level requirement of a system while other views details — how those requirements are realized. When all other four views are completed, it’s effectively redundant. However, all other views would not be possible without it.

Correspondence between views

- Views are interconnected.
- Start with Logical view (Req. Doc) and Move to Development or Process view and then finally go to Physical view.

4+1 View Model of Architecture

End user

Programmers & software managers

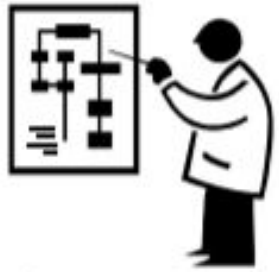
Logical view

Development view

Scenarios

Process View

Physical View



System Engineer

- For any given system, you may not need to document all the views, and there may be other, more relevant views for your system.

One Model—Two Views

- Viewing a business system from the outside, we take on the role of a customer, a business partner, a supplier, or another business system. From
  - this **external view**, only those business processes that involve outsiders are of interest. The external view describes the environment of a business system. The business system itself remains a black box.
- Within the business system, we find *employees* and *tools* that are responsible for fulfilling the demands of the environment, and for handling the necessary business processes. Behind the business processes are *workflows* and *IT systems*. Each individual employee is part of the *organizational structure*. Normally this **internal view** remains hidden to outsiders.

External and Internal view of a business system