Software Engineering

Dr. Novarun Deb

System Modeling

Topics to be covered

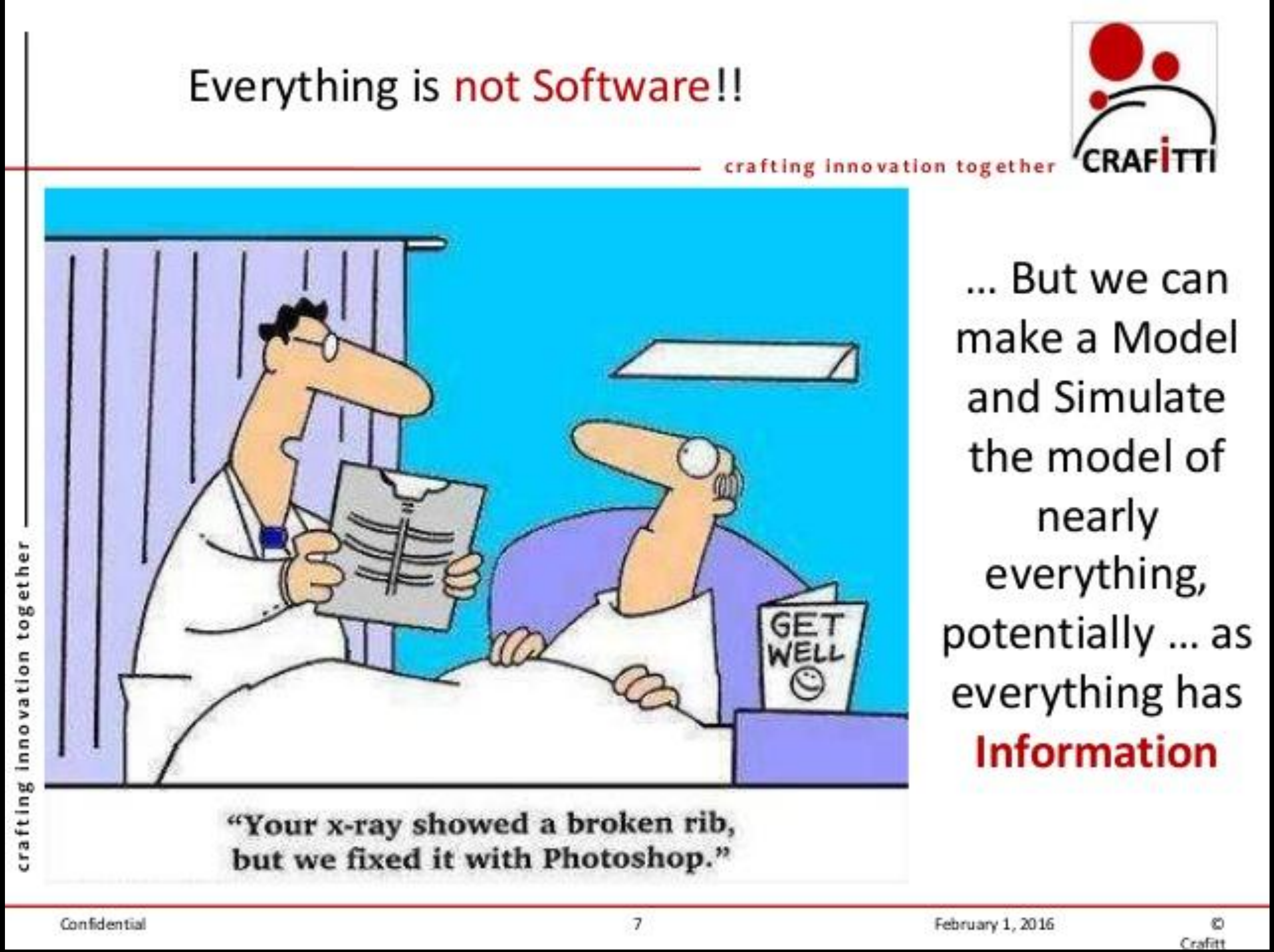
Chapter 5: Software Engineering (Ian Sommerville)

- Context models
- Interaction models
- Structural models
- Behavioral models
- Model-driven engineering

System Modeling

What is it.

- System modeling is the process of developing abstract models of a system.
  - Each model presenting a different view or perspective of that system.
- System modeling has now come to mean representing a system using some kind of graphical notation.
  - Based on notations in the Unified Modeling Language (UML).
- System modelling helps the analyst to understand the functionality of the system.
  - Models are used to communicate with customers.



Everything is not Software!!

crafting innovation together CRAFiTTi

"Your x-ray showed a broken rib, but we fixed it with Photoshop."

... But we can make a Model and Simulate the model of nearly everything, potentially ... as everything has **Information**

crafting innovation together

Confidential 7 February 1, 2016 © Crafitt

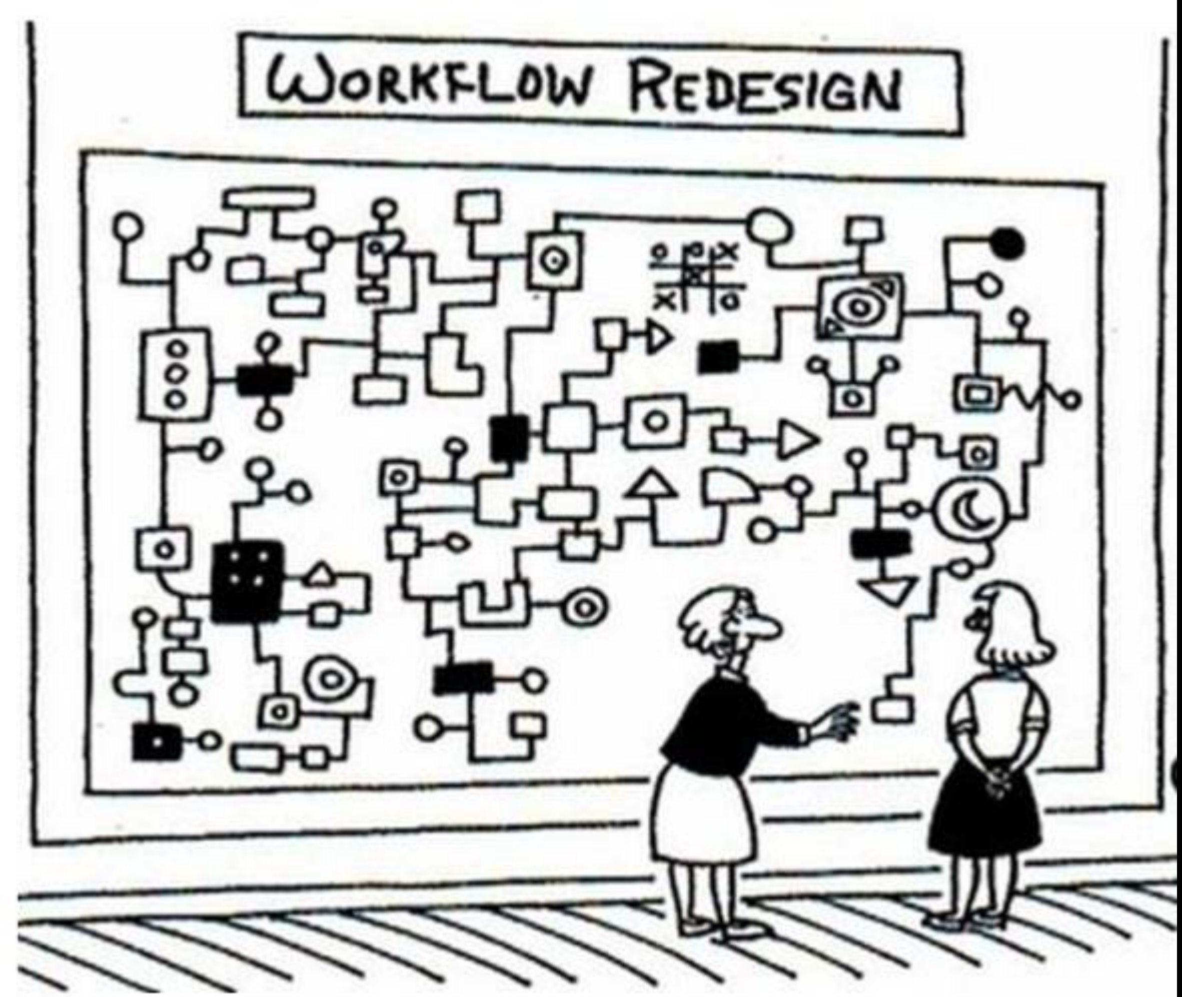
System Modeling

How do they help us?

- Models of the existing system are used during RE.
  - Clarify what the existing system does.
  - Can be used as a basis for discussing its strengths and weaknesses.
  - These then lead to requirements for the new system.

- Models of the new system are used during RE.
  - Help explain the proposed requirements to other system stakeholders.
  - Engineers use these models to discuss design proposals and to document the system for implementation.

- In a model-driven engineering process, it is possible to generate a complete or partial system implementation from the system model.

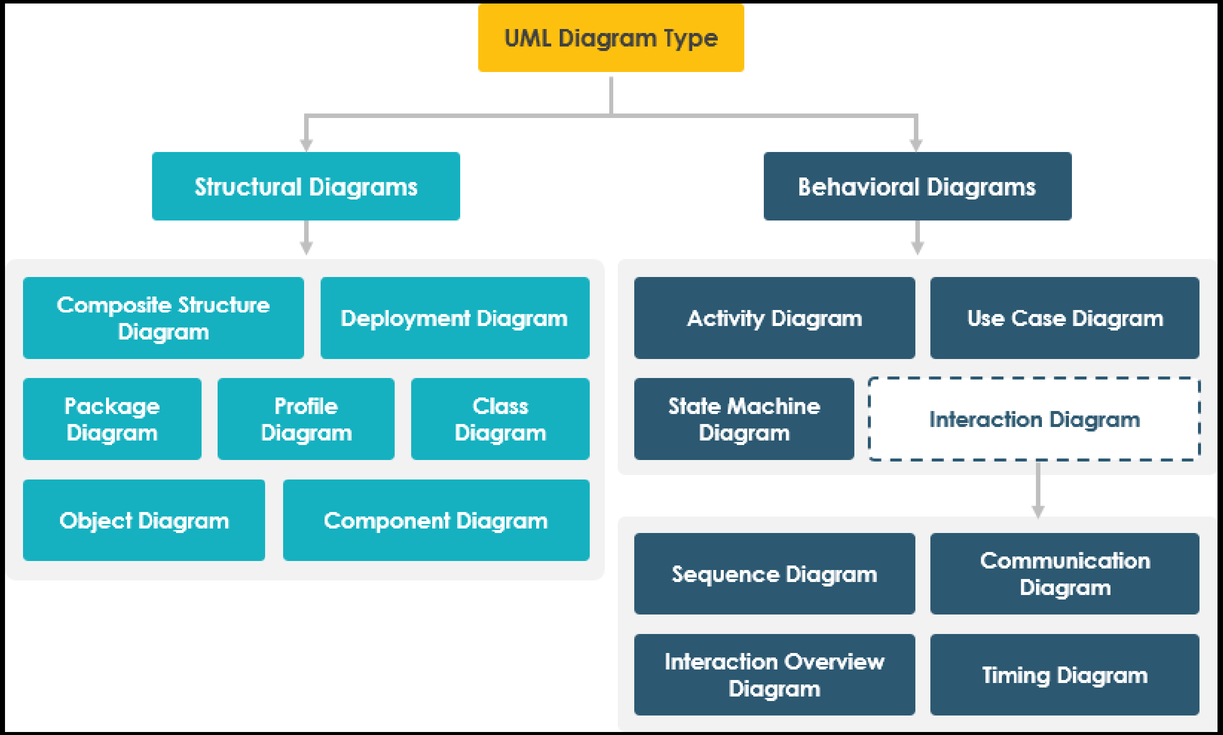


"And this is where our ED workflow redesign team went insane."

System Perspectives

What do they represent?

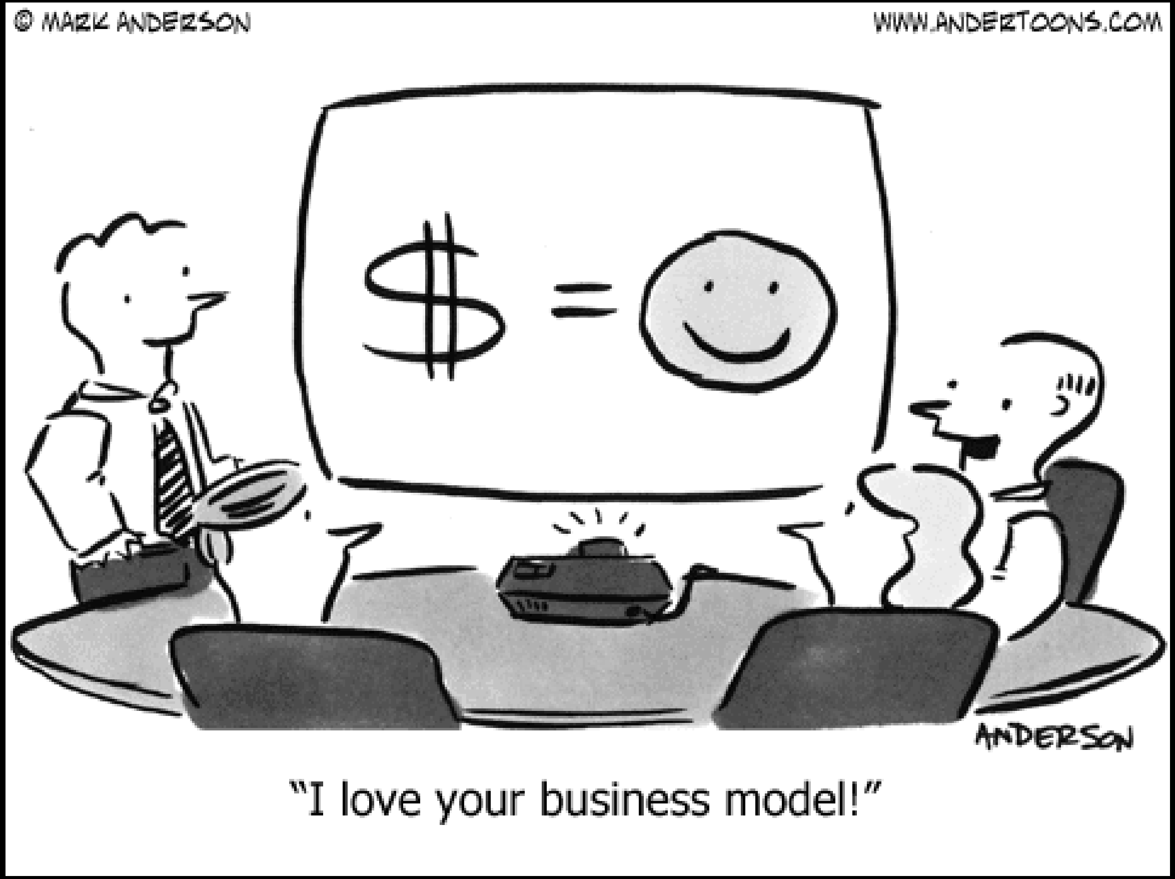
- An external perspective, where you model the context or environment of the system.
- An interaction perspective, where you model the interactions between a system and its environment, or between the components of a system.
- A structural perspective, where you model the organization of a system or the structure of the data that is processed by the system.
- A behavioral perspective, where you model the dynamic behavior of the system and how it responds to events.



Use of Graphical Models

Motivation

- As a means of facilitating discussion about an existing or proposed system
  - Incomplete and incorrect models are OK as their role is to support discussion.
- As a way of documenting an existing system
  - Models should be an accurate representation of the system but need not be complete.
- As a detailed system description that can be used to generate a system implementation
  - Models have to be both correct and complete.

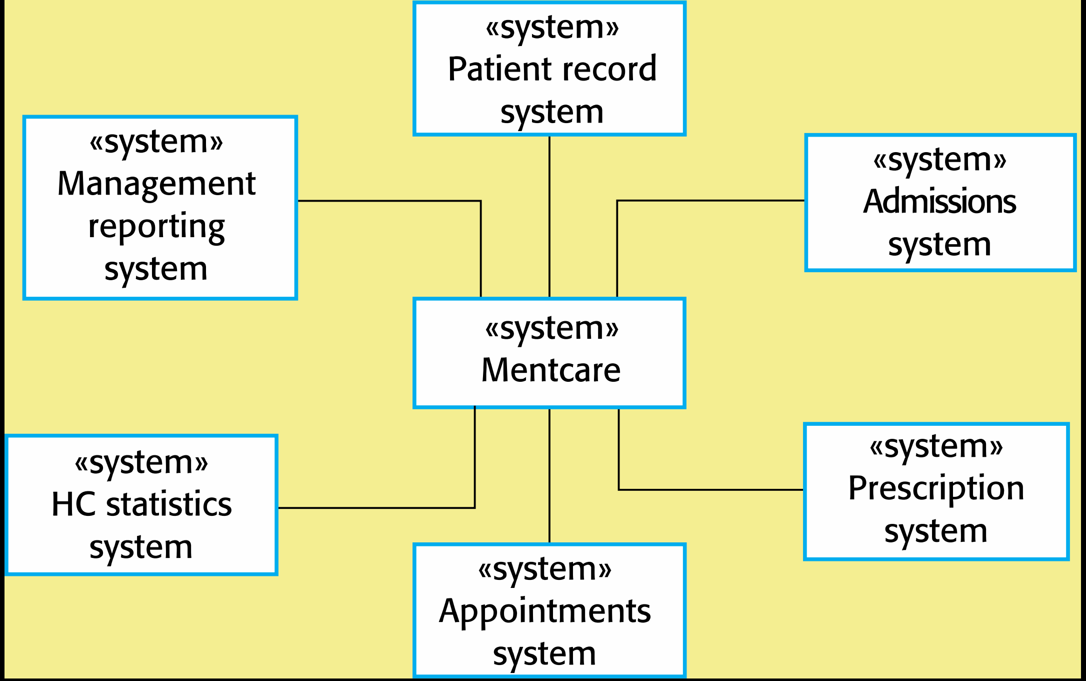


Context Models

Context Models

Motivation

- Context models are used to illustrate the operational context of a system
  - They show what lies outside the system boundaries.
- Social and organisational concerns may affect the decision on where to position system boundaries.
- System boundaries are established to define what is inside and what is outside the system.
  - They show other systems that are used or depend on the system being developed.
  - The position of the system boundary has a profound effect on the system requirements.
  - There may be pressures to develop system boundaries that increase / decrease the influence or workload of different parts of an organization.


«system» Patient record system
«system» Management reporting system
«system» Admissions system
«system» Mentcare
«system» HC statistics system
«system» Prescription system
«system» Appointments system

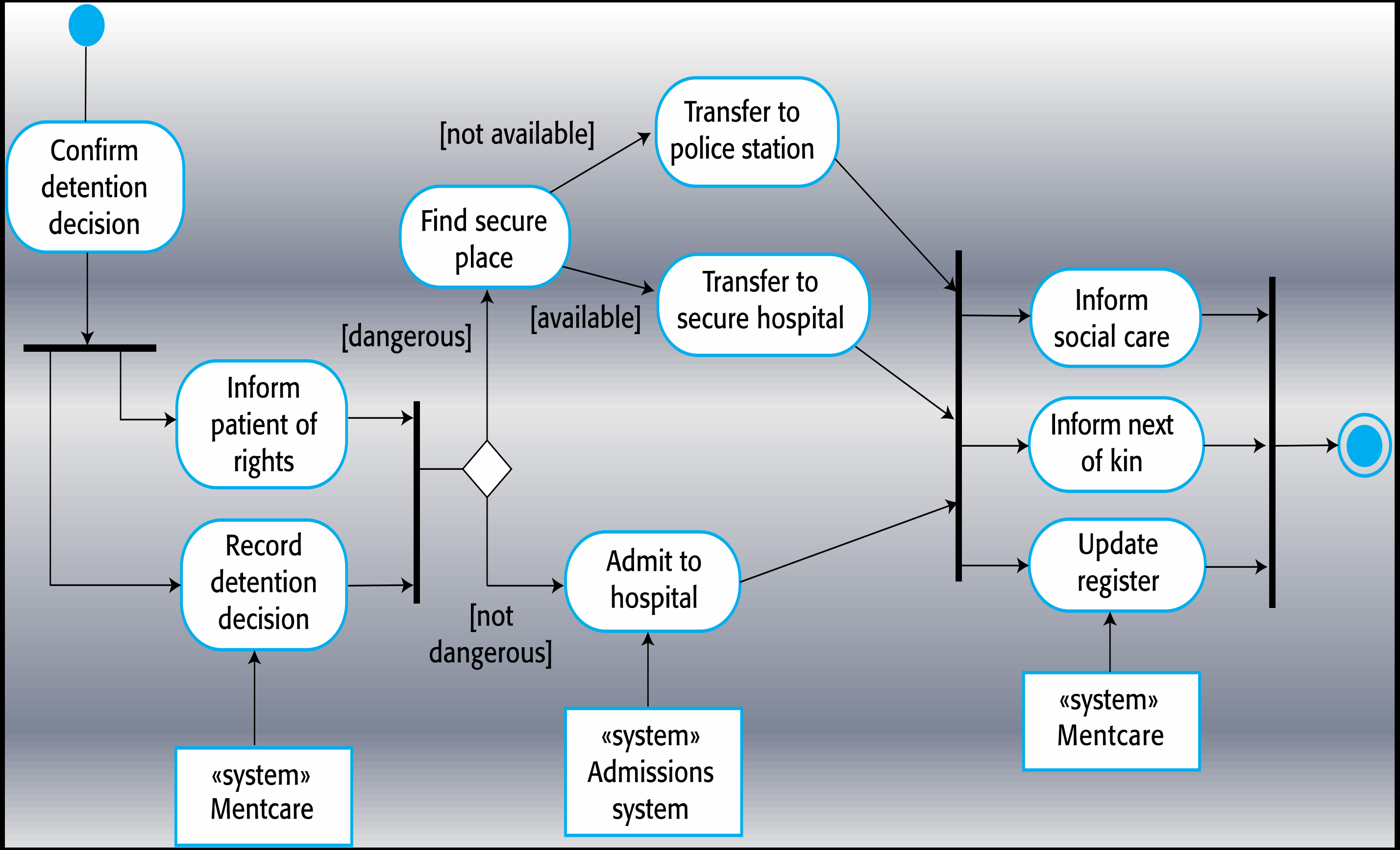
Process Perspective

Motivation

- Context models simply show the other systems in the environment.
  - Not how the system being developed is used in that environment.

- Process models reveal how the system being developed is used in broader business processes.

- UML activity diagrams may be used to define business process models.

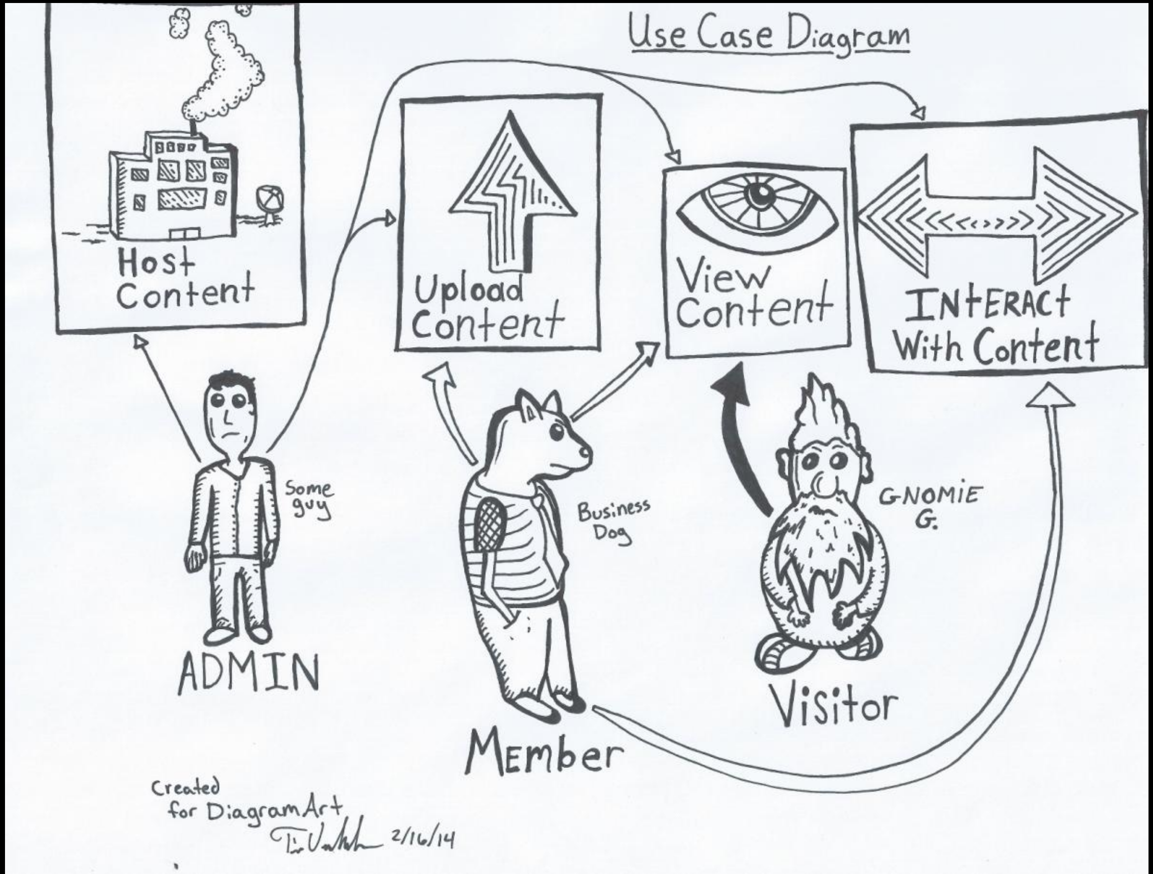


Interaction Models

Interaction Models

Motivation

- Modeling user interaction is important as it helps to identify user requirements.
- Modeling system-to-system interaction highlights the communication problems that may arise.
- Modeling component interaction helps us understand if a proposed system structure is likely to deliver the required system performance and dependability.
- Use case diagrams and Sequence diagrams may be used for interaction modelling.



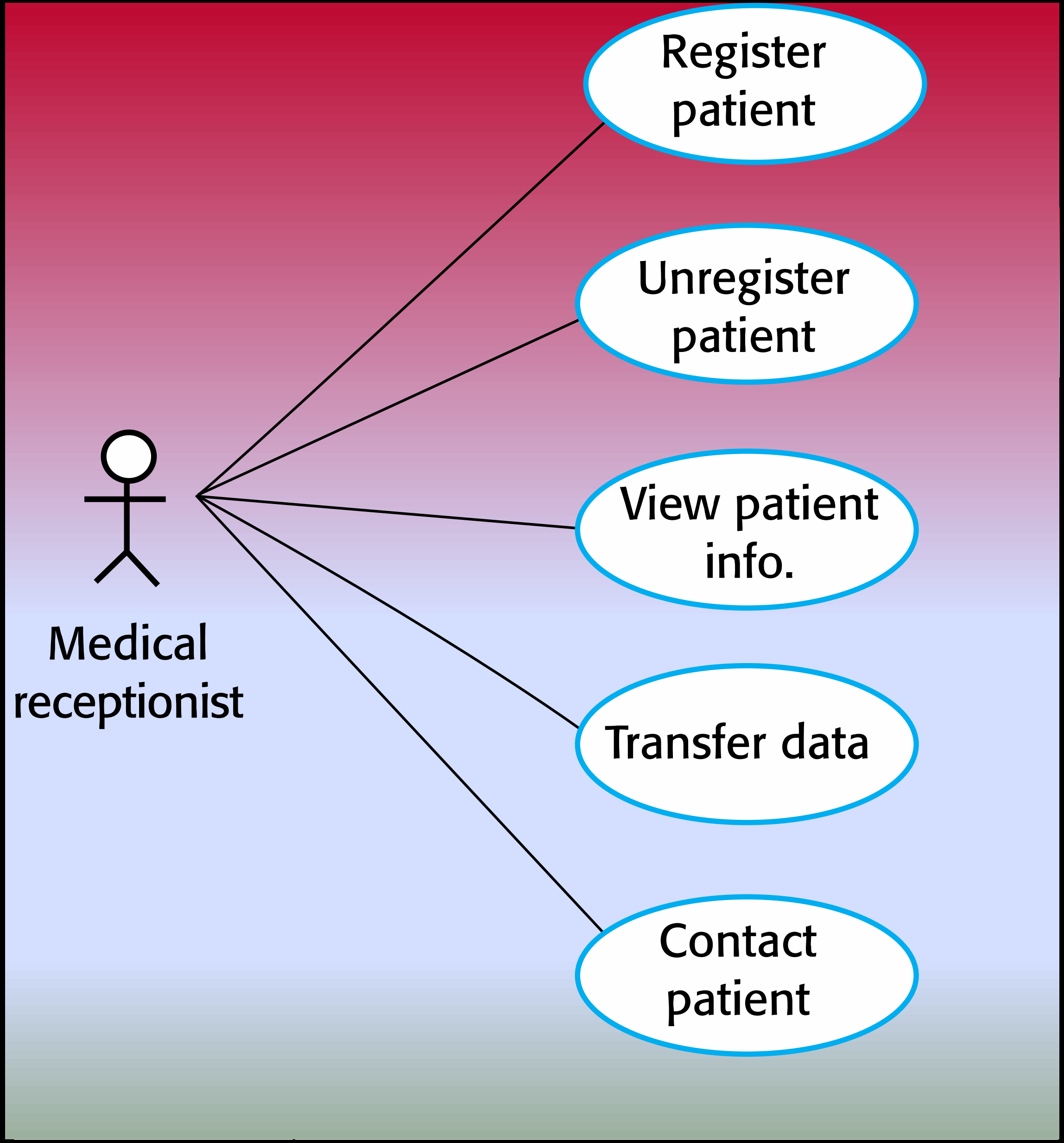
Use Case Modeling

- Use cases were developed originally to support requirements elicitation and now incorporated into the UML.

- Each use case represents a discrete task that involves external interaction with a system.

- Actors in a use case may be people or other systems.

- Represented diagrammatically to provide an overview of the use case and in a more detailed textual form.



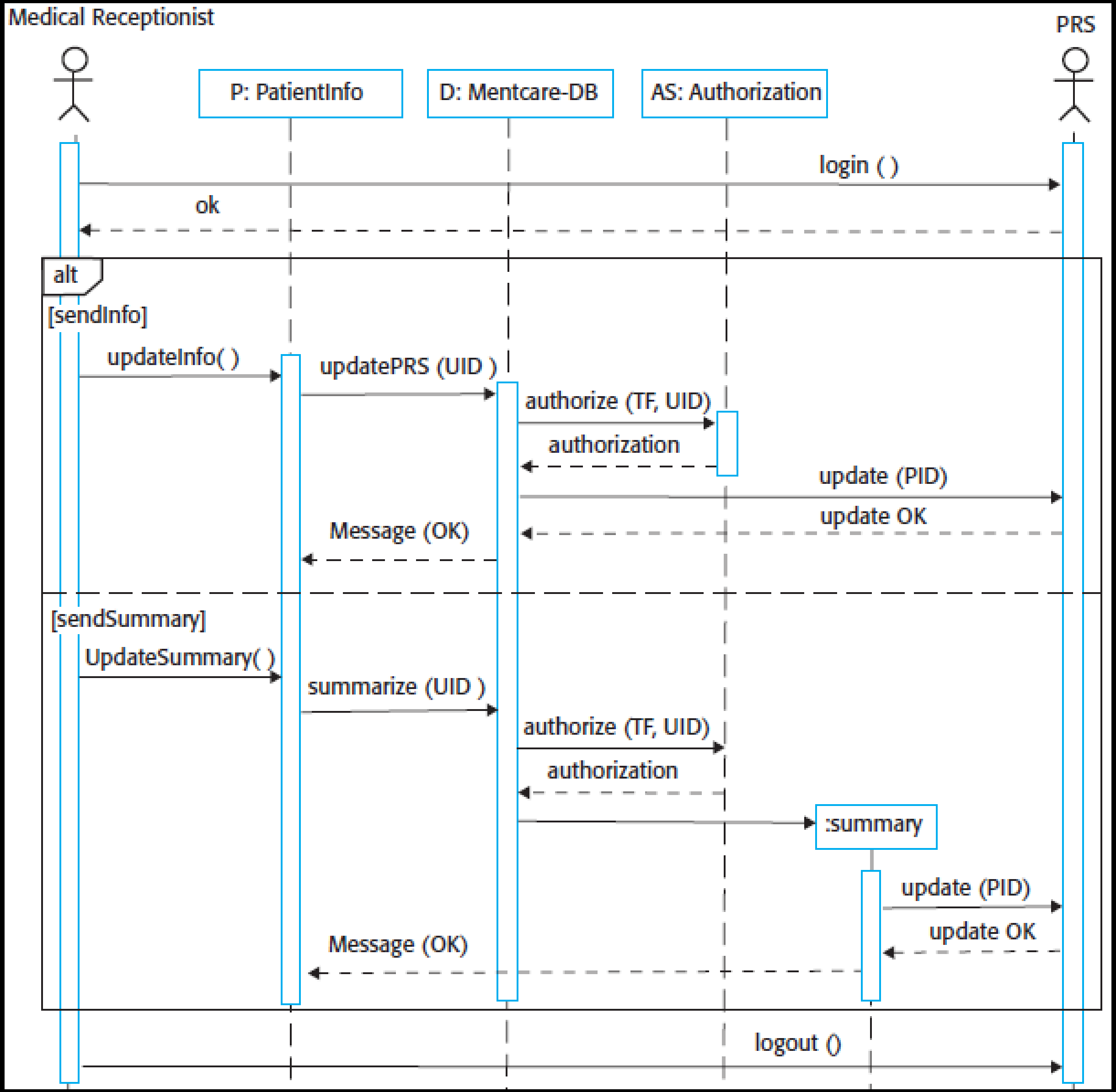
Sequence Diagrams

- Sequence diagrams are part of the UML and are used to model the interactions between the actors and the objects within a system.

- A sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance.

- The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these.

- Interactions between objects are indicated by annotated arrows.

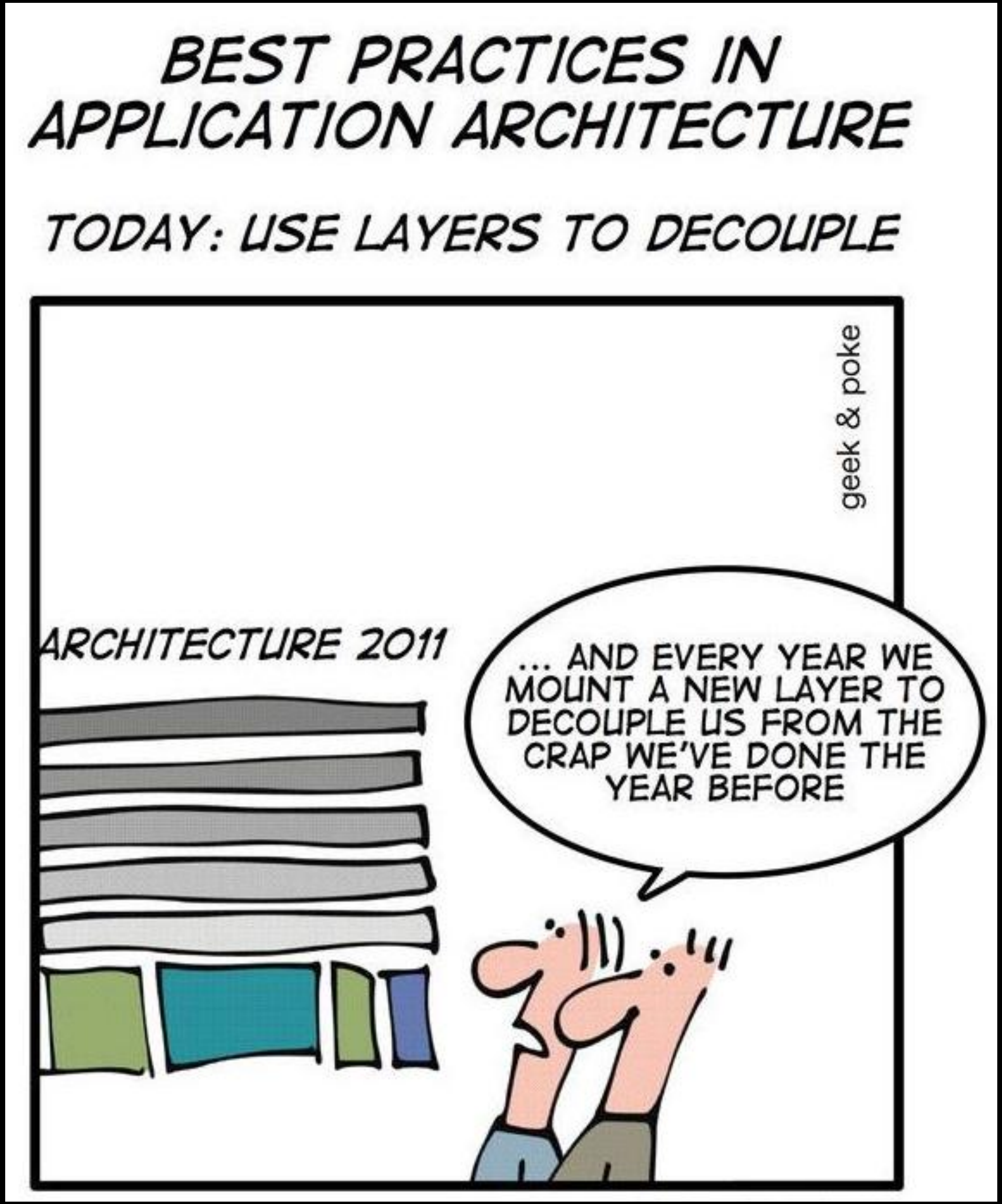


Structural Models

Structural Models

Motivation

- Structural models of software display the organization of a system in terms of the components that make up that system and their relationships.
- Structural models may be static models, which show the structure of the system design, or dynamic models, which show the organization of the system when it is executing.
- You create structural models of a system when you are discussing and designing the system architecture.


BEST PRACTICES IN APPLICATION ARCHITECTURE
TODAY: USE LAYERS TO DECOUPLE
geek & poke
ARCHITECTURE 2011
... AND EVERY YEAR WE MOUNT A NEW LAYER TO DECOUPLE US FROM THE CRAP WE'VE DONE THE YEAR BEFORE

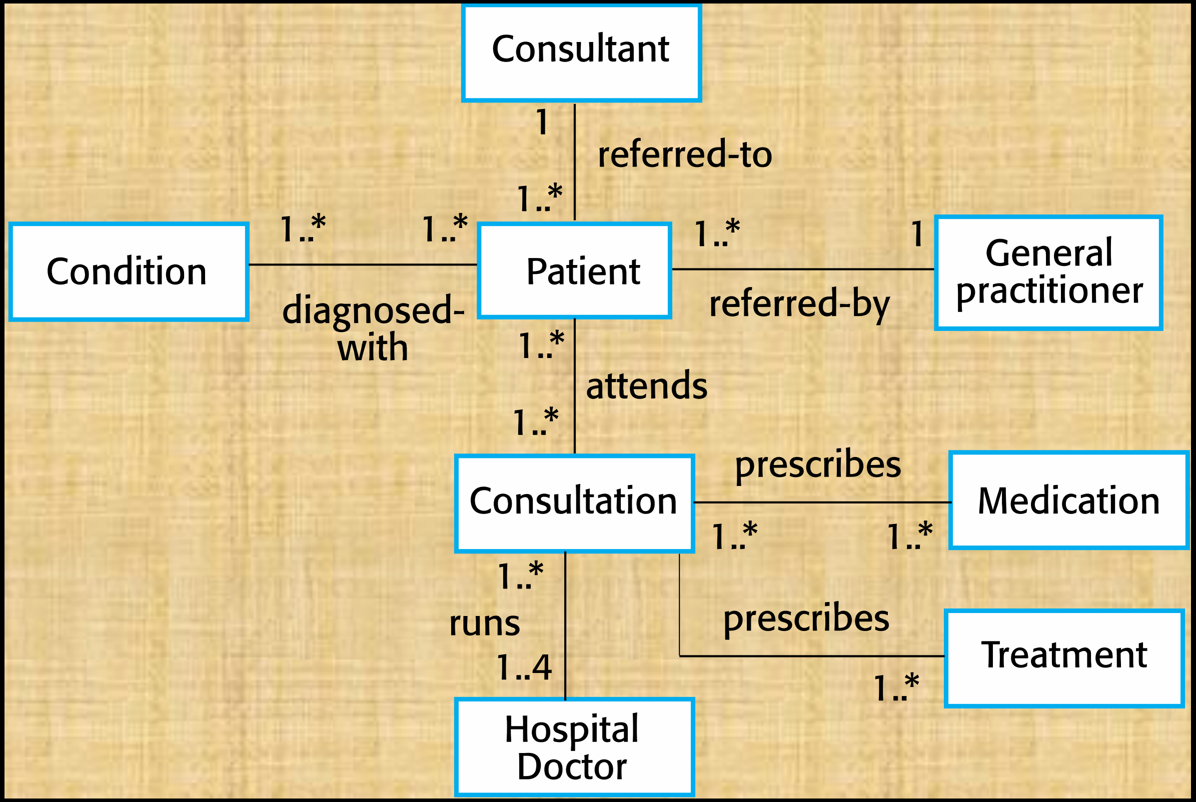
Class Diagrams

- Class diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between these classes.

- An object class can be thought of as a general definition of one kind of system object.

- An association is a link between classes that indicates that there is some relationship between these classes.

- When you are developing models during the early stages of the software engineering process, objects represent something in the real world, such as a patient, a prescription, doctor, etc.



Class Diagrams

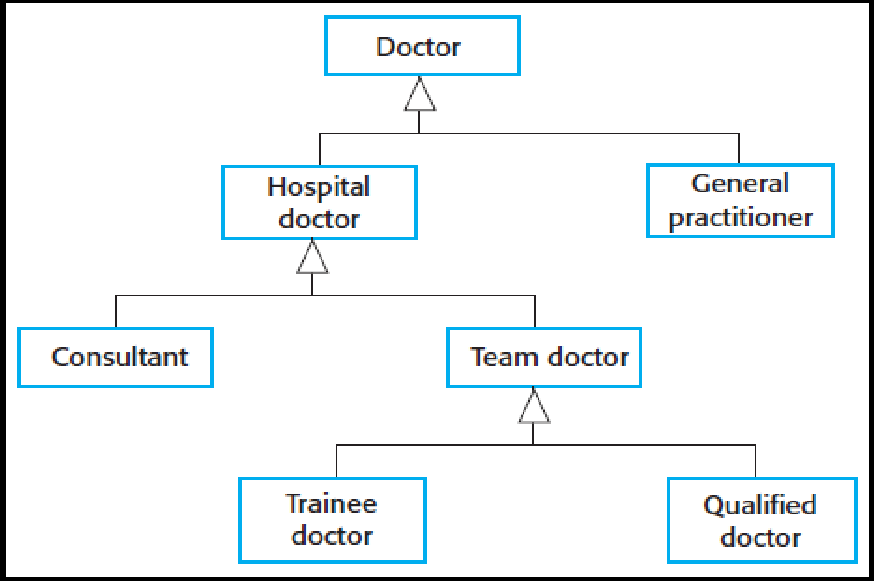
Generalizations

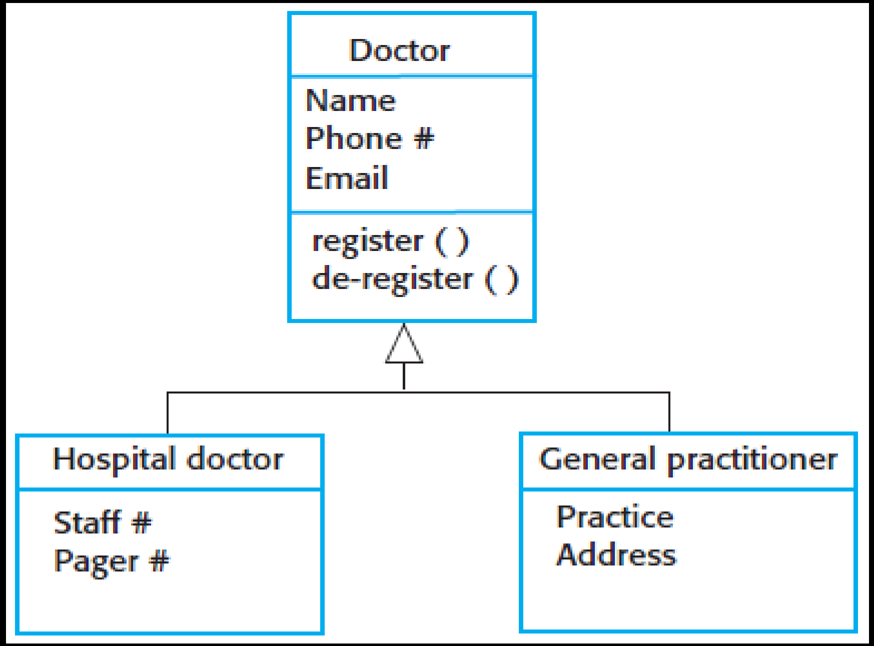
- Always check scope for generalization.
  - If changes are proposed, then you do not have to look at all classes in the system to see if they are affected by the change.

- Implemented using the class inheritance mechanisms.

- The attributes and operations associated with higher-level classes are also associated with the lower-level classes.

- The lower-level classes are subclasses inherit the attributes and operations from their superclasses.
  - These lower-level classes then add more specific attributes and operations.





Class Diagrams

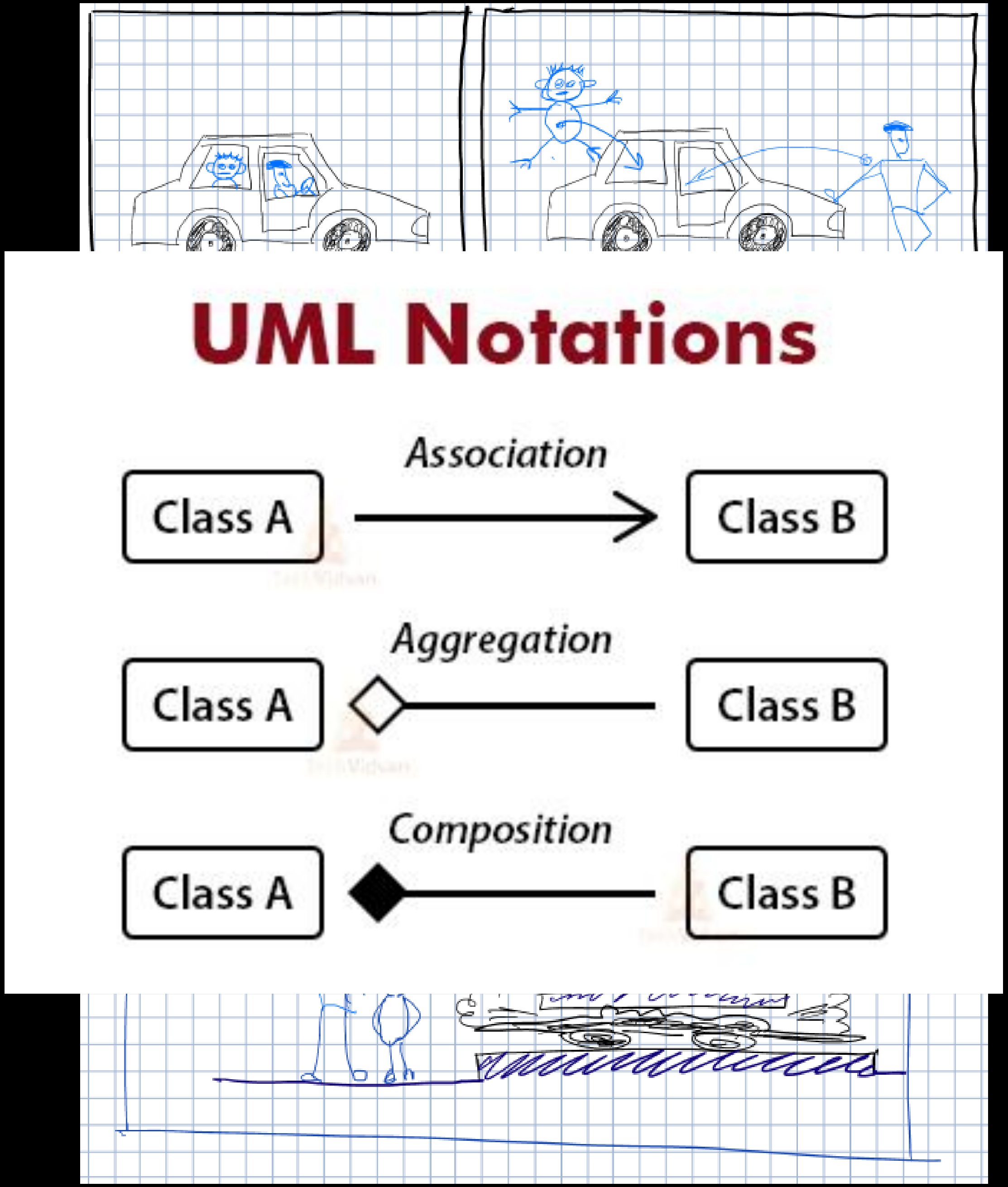
Associations - Aggregations - Compositions

- An aggregation model shows how classes that are collections are composed of other classes.

- Aggregation models are similar to the part-of relationship in semantic data models.

- Objects in the real world are often made up of different parts.

- The UML provides a special type of association between classes such that one object (the whole) is composed of other objects (the parts).
  - To define aggregation, a diamond shape is added to the link next to the class that represents the whole.

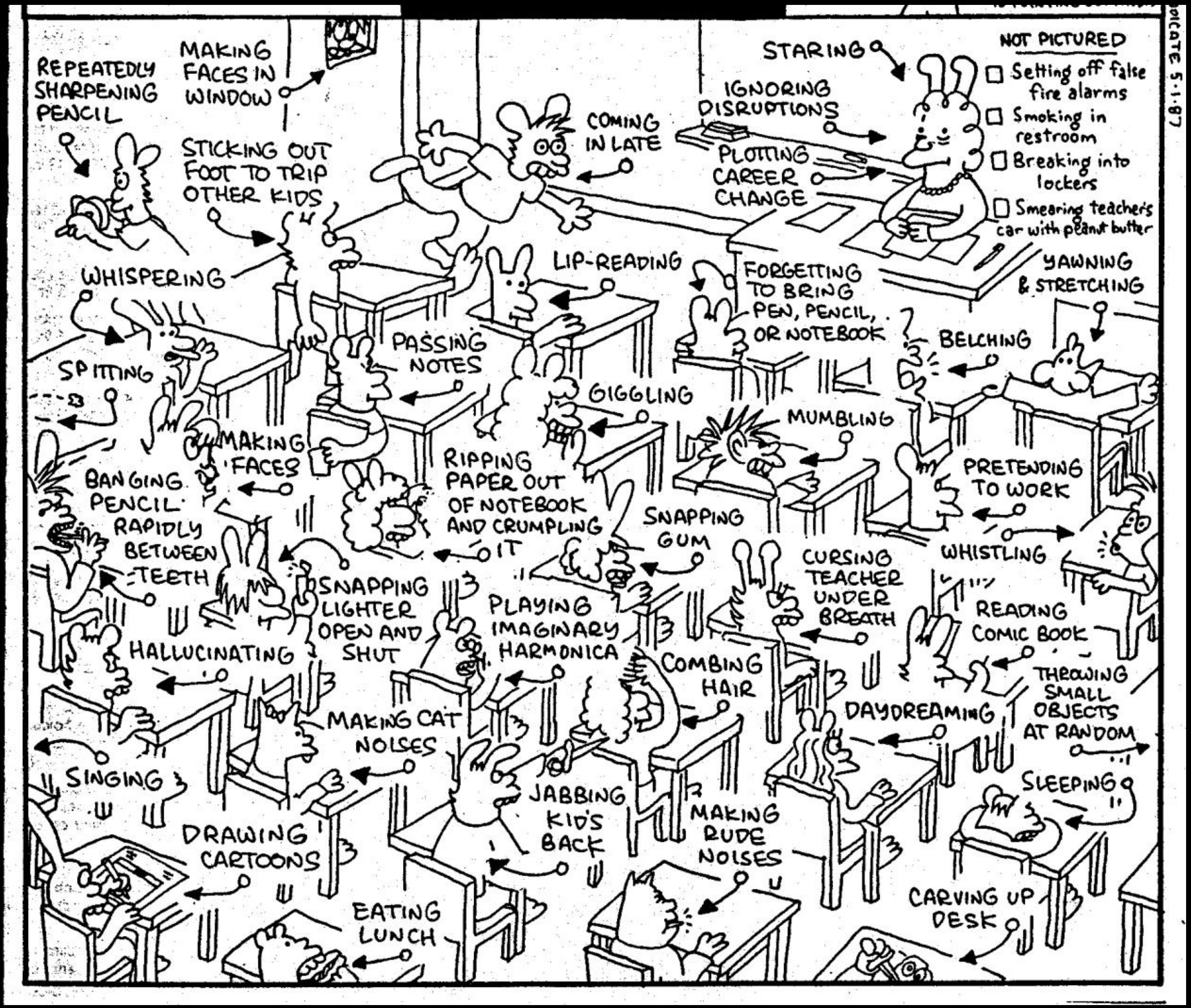


Behavioral Models

Behavioral Models

Motivation

- Behavioral models are models of the dynamic behavior of a system as it is executing.
- They show what happens or what is supposed to happen when a system responds to a stimulus from its environment.
- You can think of these stimuli as being of two types:
  - Data: Some data arrives that has to be processed by the system.
  - Events: Some event happens that triggers system processing. Events may have associated data, although this is not always the case.



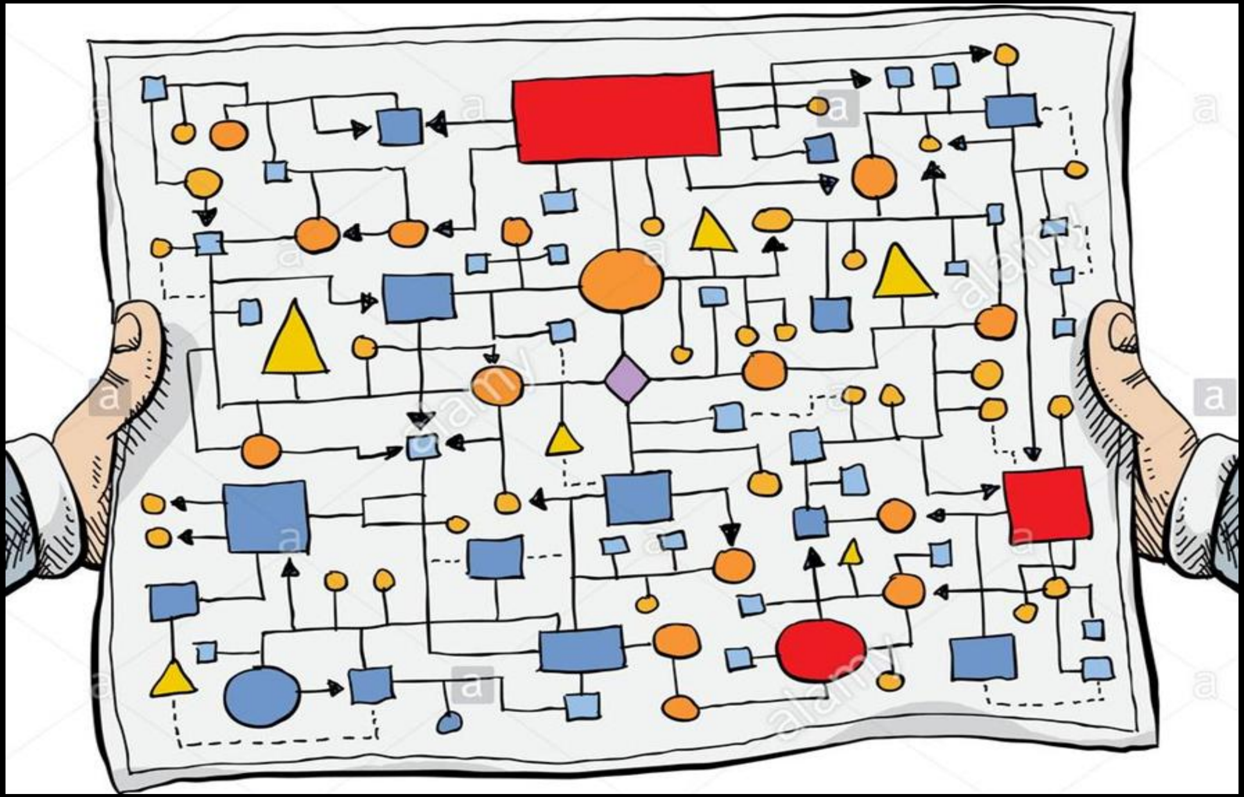
Data-driven Modeling

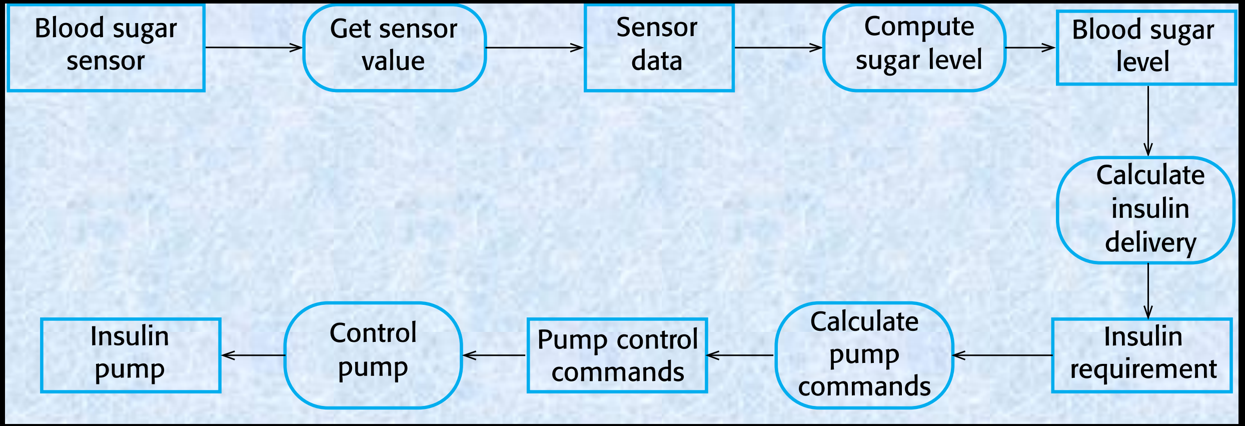
- Many business systems are data-processing systems that are primarily driven by data.

- They are controlled by the data input to the system, with relatively little external event processing.

- Data-driven models show the sequence of actions involved in processing input data and generating an associated output.

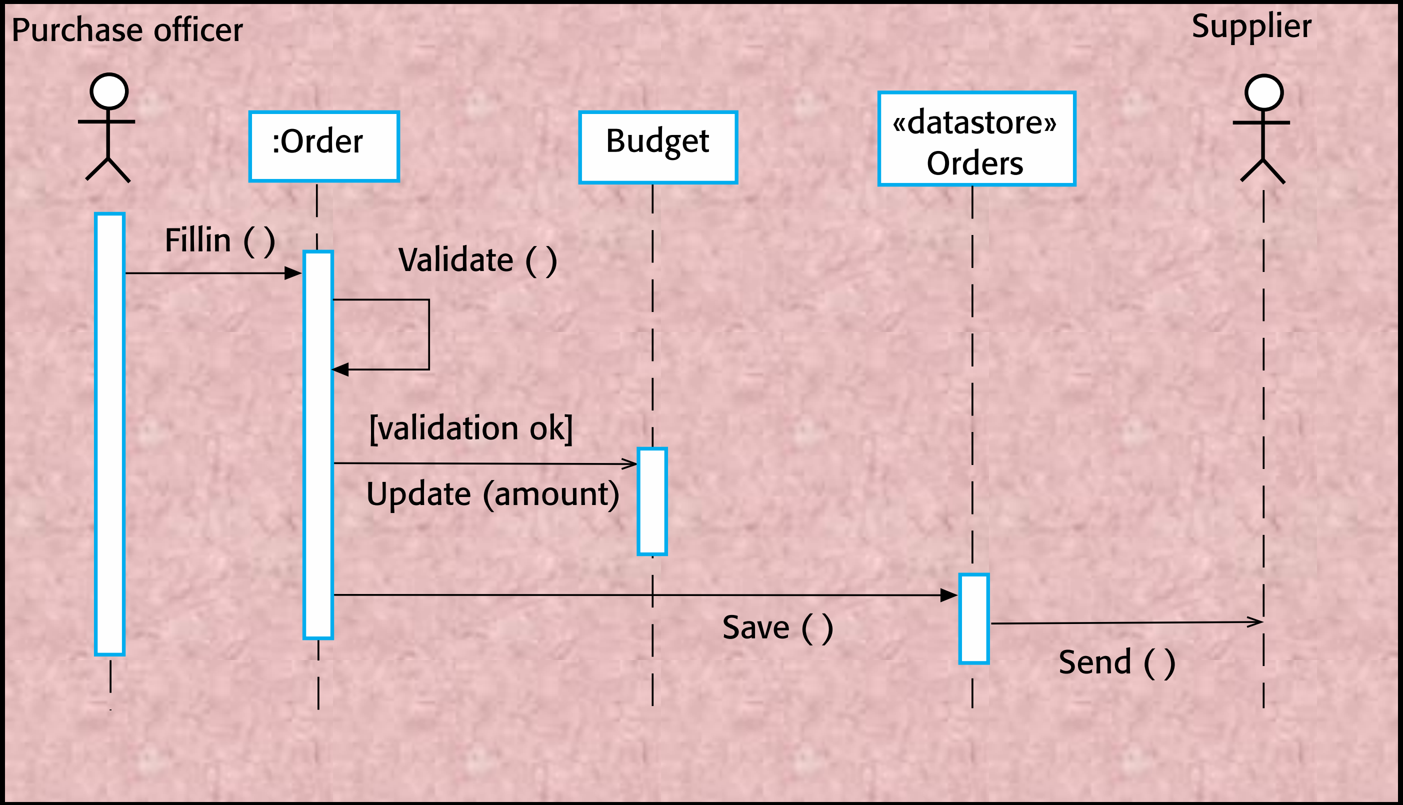
- They are particularly useful during the analysis of requirements as they can be used to show end-to-end processing in a system.





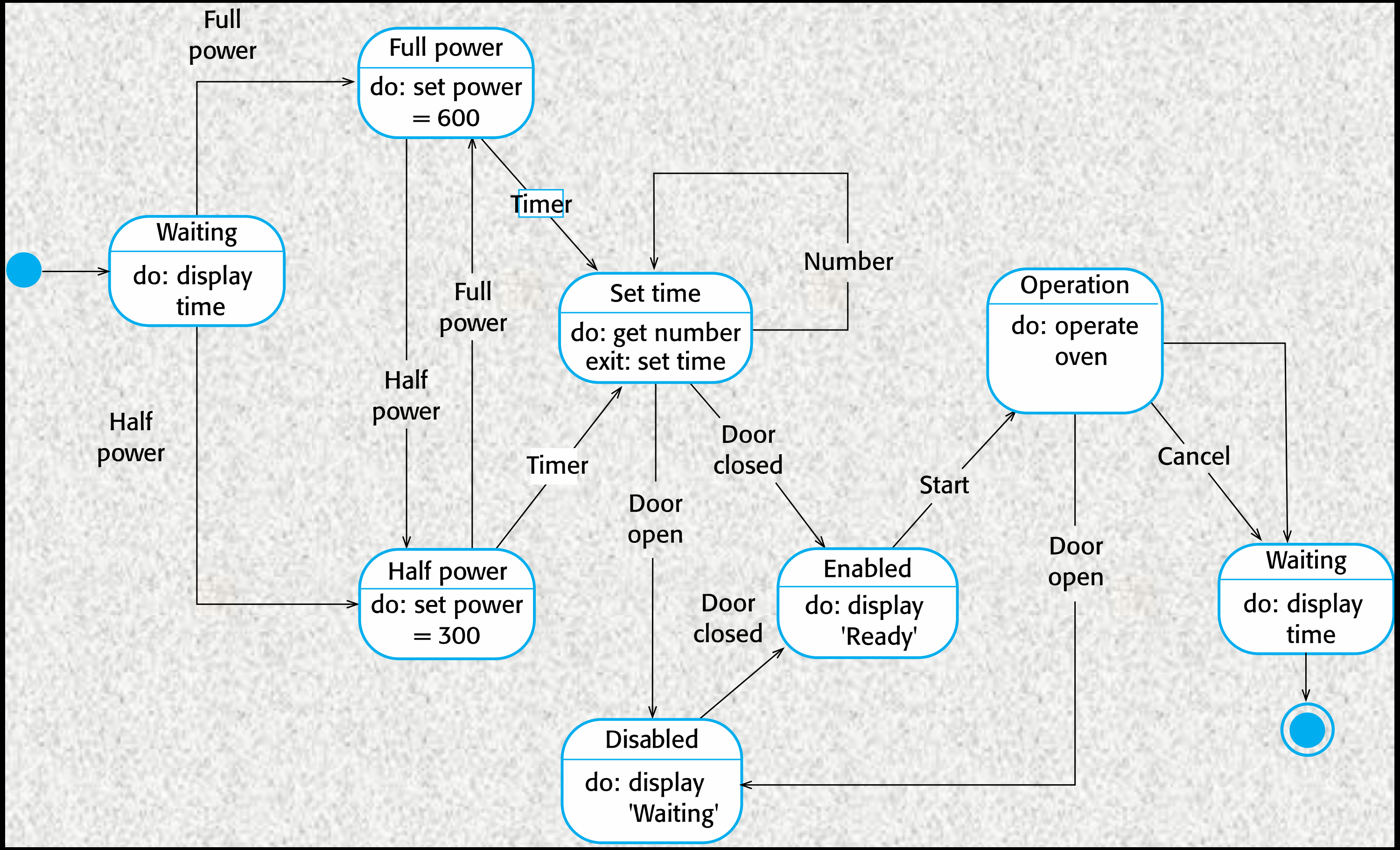
Event-driven Modeling

- Real-time systems are often event-driven, with minimal data processing.
- Event-driven modeling shows how a system responds to external and internal events.
- It is based on the assumption that a system has a finite number of states and that events (stimuli) may cause a transition from one state to another.



State Machine Models

- Model the behaviour of the system in response to external and internal events.
- Often used for modelling real-time systems.
- State machine models show system states as nodes and events as arcs between these nodes.
  - When an event occurs, the system moves from one state to another.
- Statecharts are an integral part of UML models.

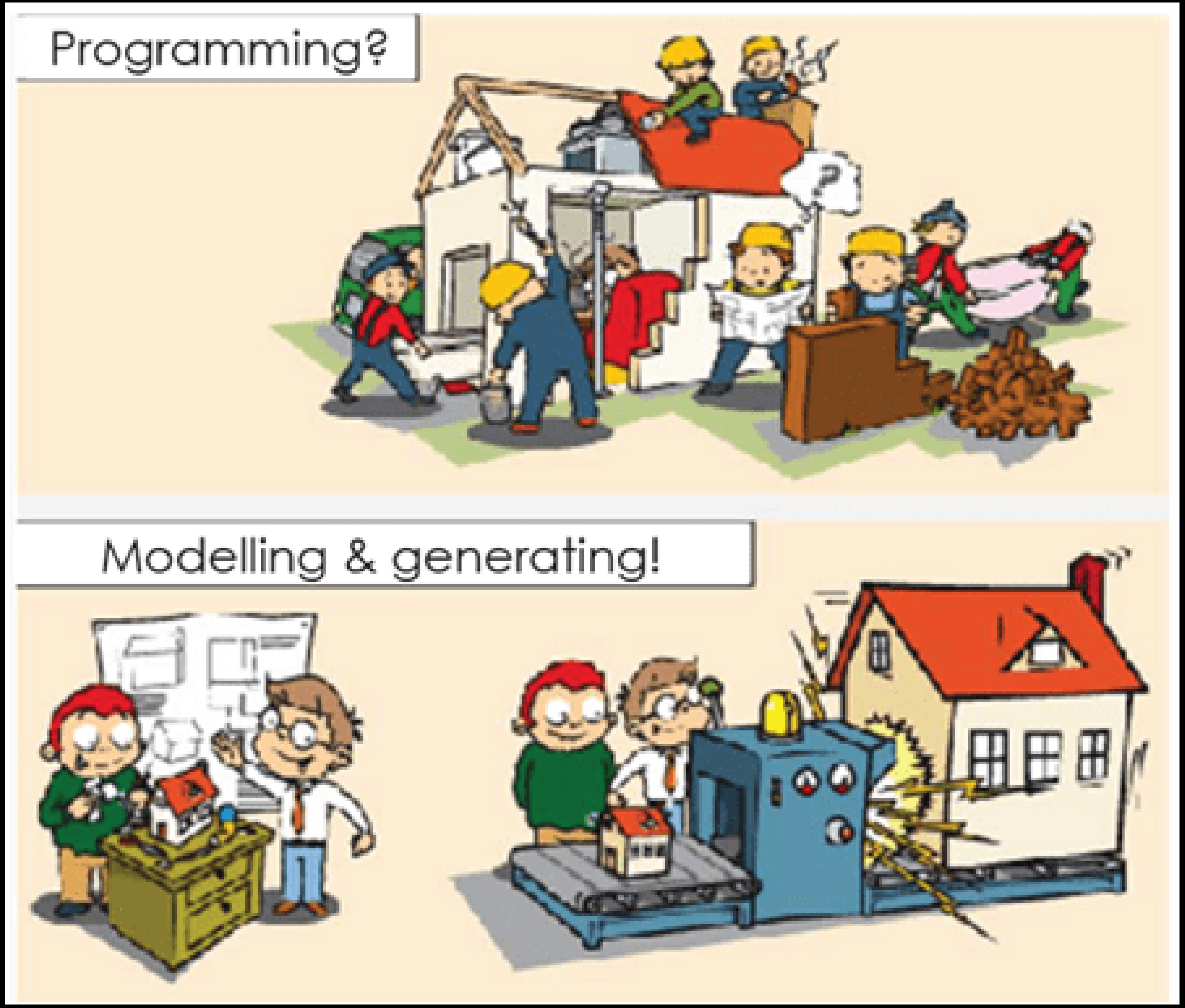


Model – driven Engineering

Model-Driven Engineering

Motivation

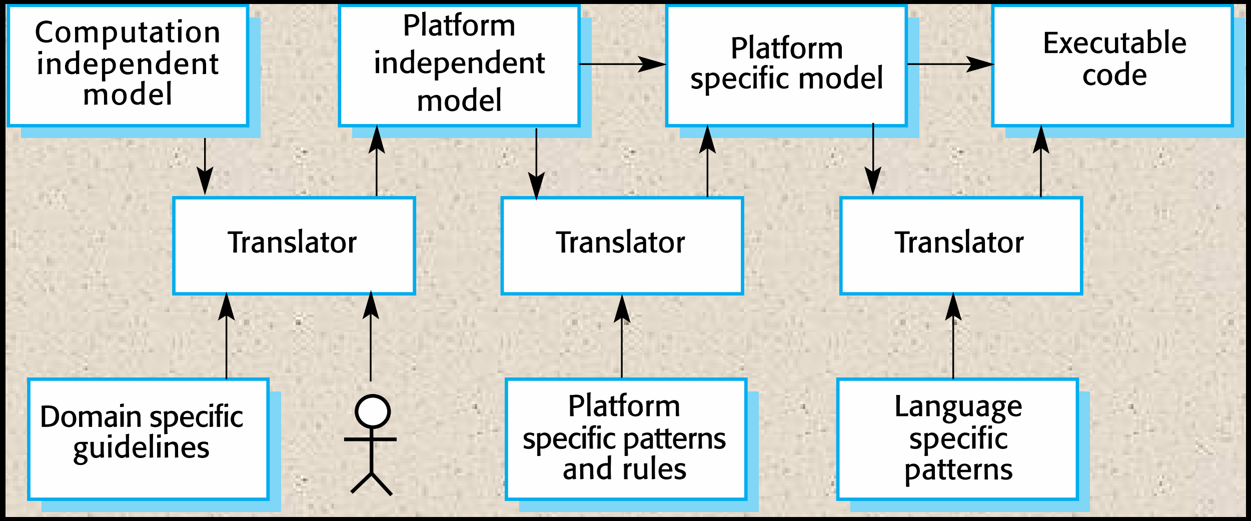
- Model-driven engineering (MDE) is an approach to software development.
  - Models rather than programs are the principal outputs of the development process.
- The programs that execute on a hardware/software platform are then generated automatically from the models.
- Raises the level of abstraction in software engineering.
  - Engineers no longer concerned with programming language details or the specifics of execution platforms.



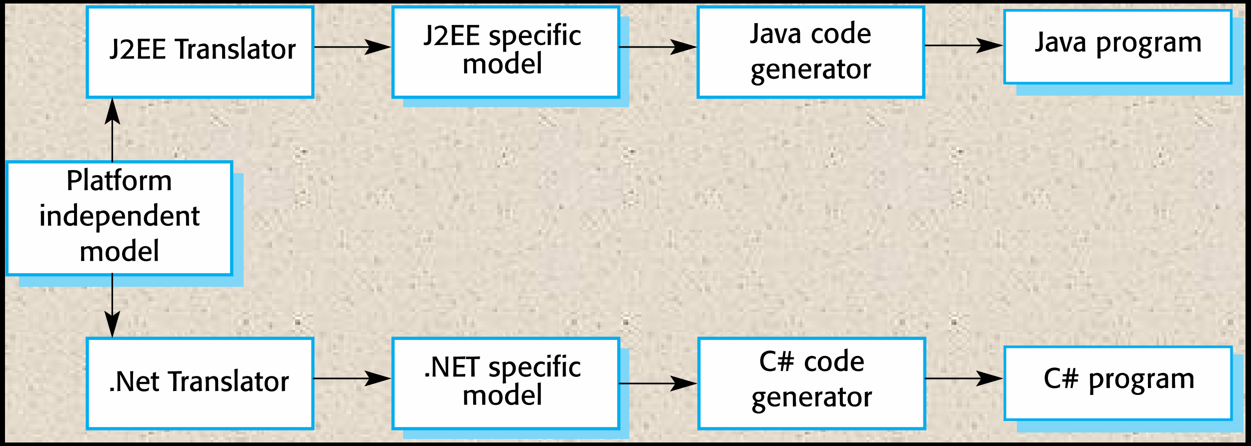
Model-Driven Architecture

Motivation

- Models at different levels of abstraction are created.
- A computation independent model (CIM)
  - These model the important domain abstractions used in a system. CIMs are sometimes called domain models.
- A platform independent model (PIM)
  - These model the operation of the system without reference to its implementation. The PIM is usually described using UML models.
- Platform specific models (PSM)
  - These are transformations of the platform-independent model with a separate PSM for each application platform.



Computation independent model
Platform independent model
Platform specific model
Executable code
Translator
Translator
Translator
Domain specific guidelines
Platform specific patterns and rules
Language specific patterns



J2EE Translator
J2EE specific model
Java code generator
Java program
Platform independent model
.Net Translator
.NET specific model
C# code generator
C# program