Software Engineering

Rogério de Lemos

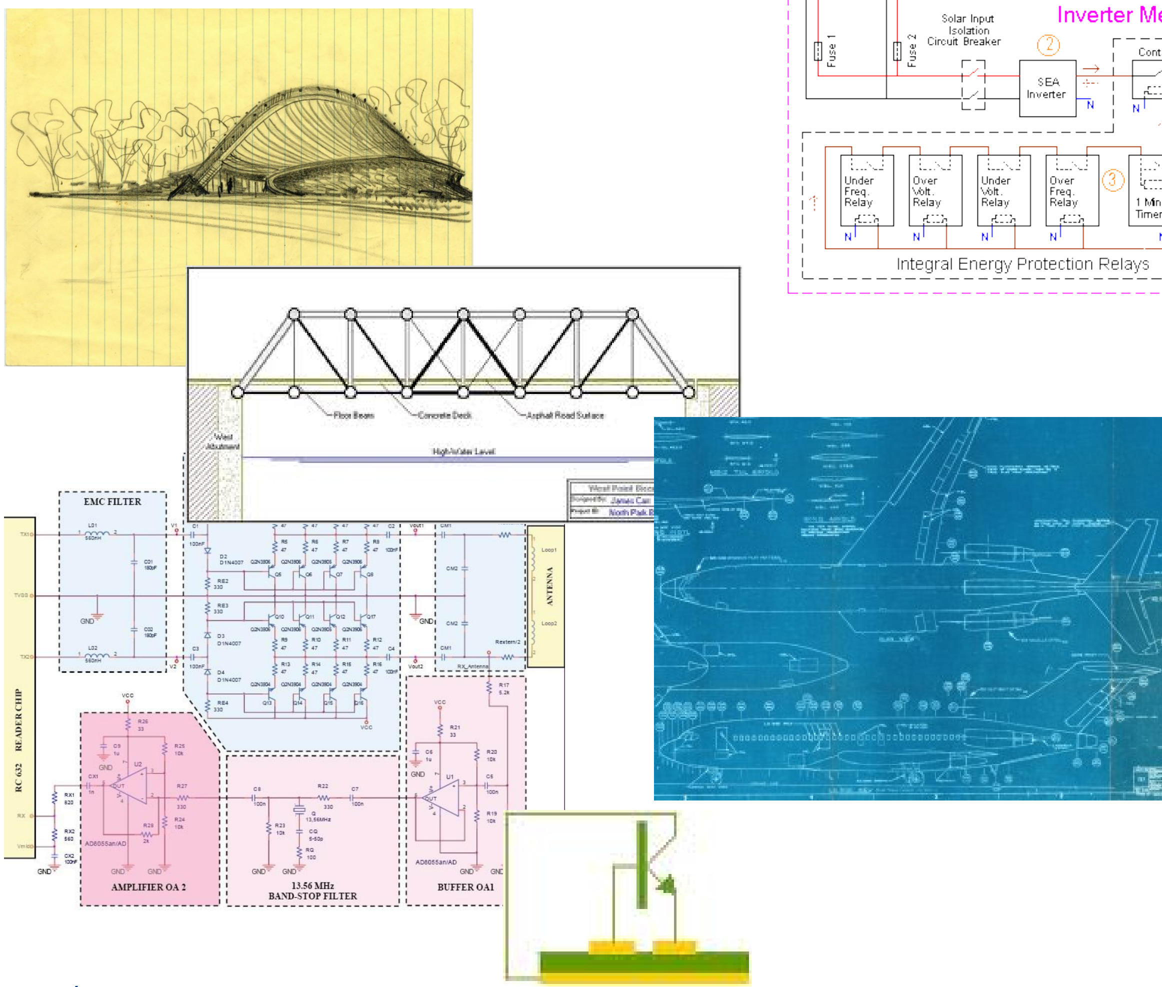
- System models

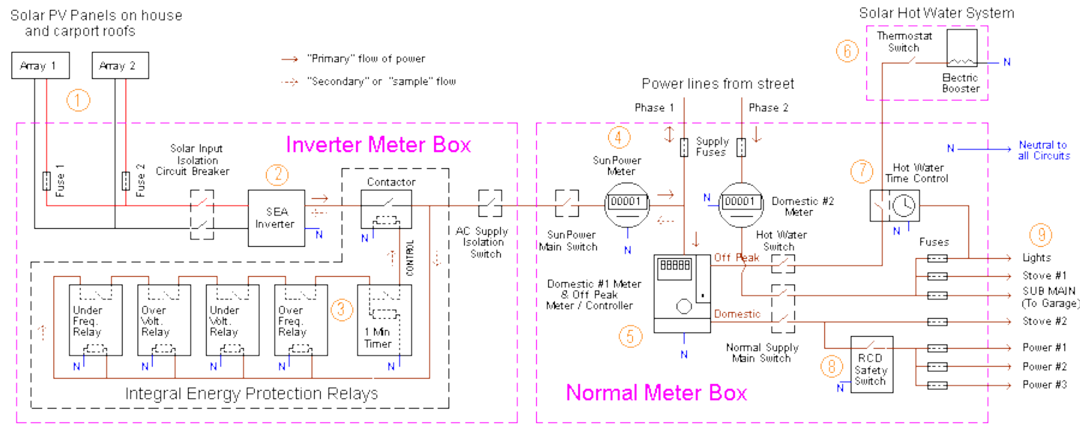
Lecture Outline

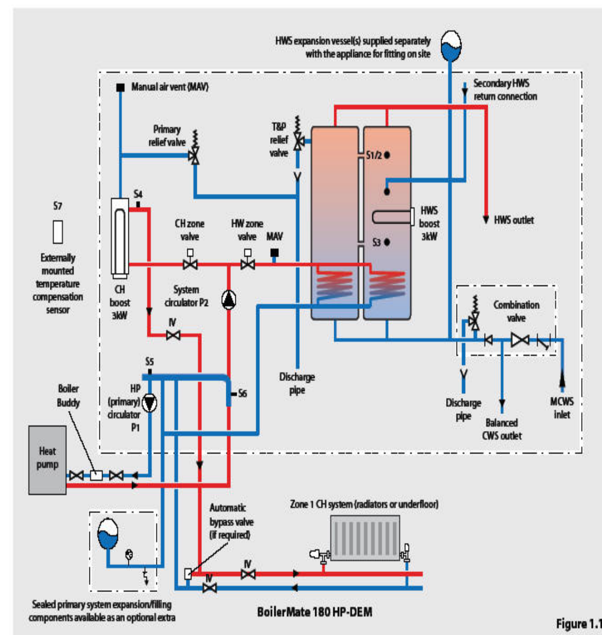
- Motivating the usage of models
- System models
  - context, interaction, structural, behavioral models
- Example of models
- Model driven engineering

Other Engineering Disciplines

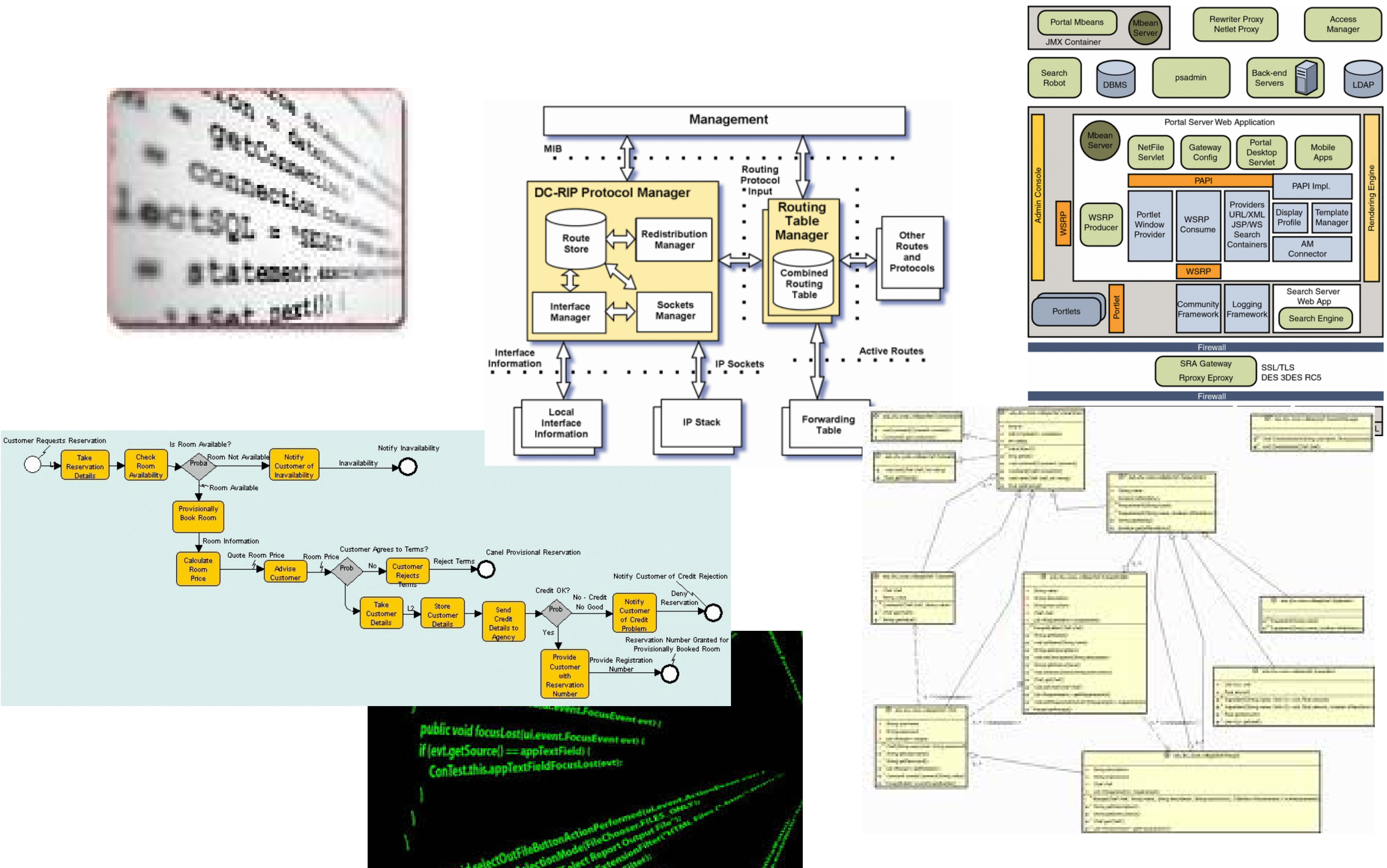
Rely heavily on models





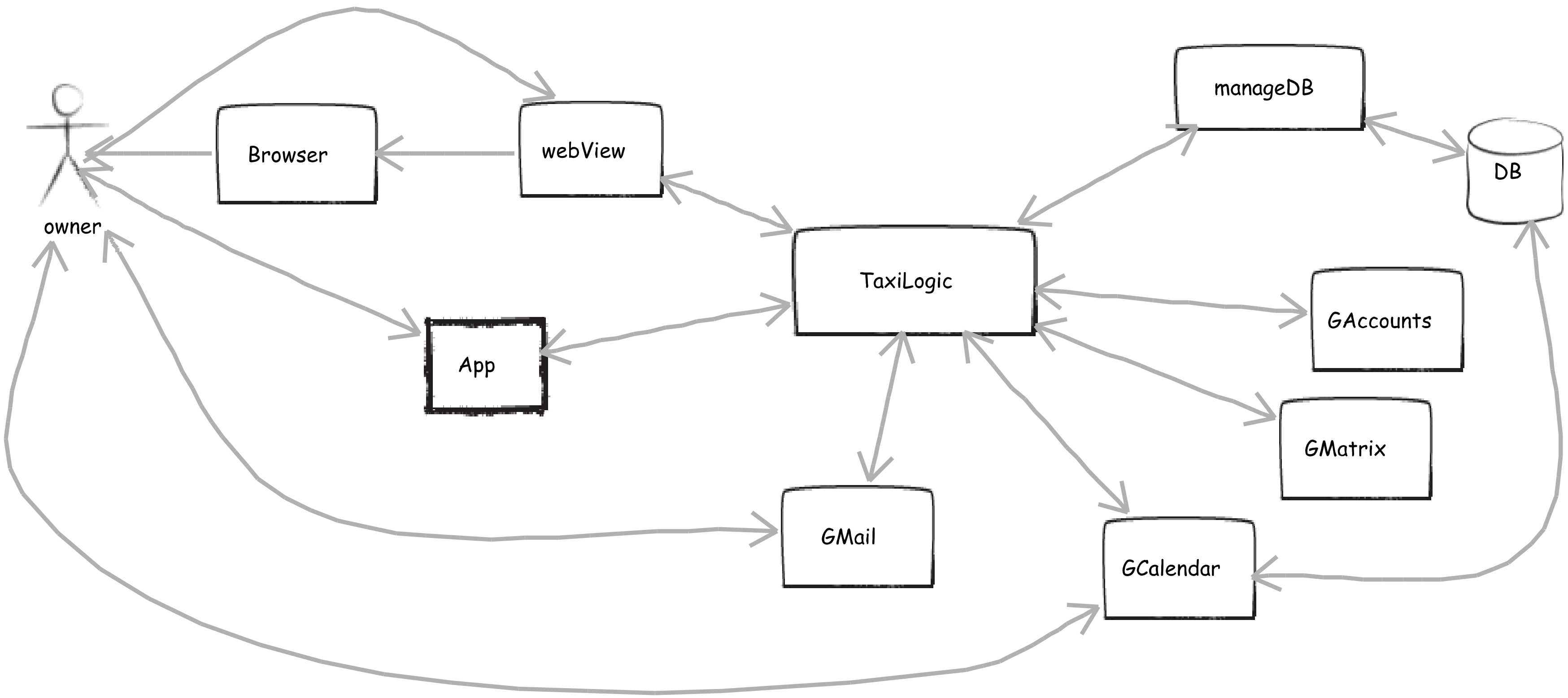


Software


Management
MIB
Routing Protocol Input
DC-RIP Protocol Manager
Route Store
Redistribution Manager
Interface Manager
Sockets Manager
Routing Table Manager
Combined Routing Table
Other Routes and Protocols
Interface Information
IP Sockets
Active Routes
Local Interface Information
IP Stack
Forwarding Table
Portal Mbeans
Mbean Server
JMX Container
Rewriter Proxy Netlet Proxy
Access Manager
Search Robot
DBMS
psadmin
Back-end Servers
LDAP
Portal Server Web Application
Admin Console
Rendering Engine
NetFile Servlet
Gateway Config
Portal Desktop Servlet
Mobile Apps
PAPI
PAPI Impl.
WSRP
WSRP Producer
Portlet Window Provider
WSRP Consume
Providers URL/XML JSP/WS Search Containers
Display Profile
Template Manager
AM Connector
Portlets
Portlet
Community Framework
Logging Framework
Search Server Web App
Search Engine
Firewall
SRA Gateway
Rproxy Eproxy
SSL/TLS DES 3DES RC5
Customer Requests Reservation
Take Reservation Details
Check Room Availability
Is Room Available?
Room Not Available
Notify Customer of Inavailability
Notify Inavailability
Inavailability
Room Available
Provisionally Book Room
Room Information
Calculate Room Price
Quote Room Price
Advise Customer
Room Price
Customer Agrees to Terms?
Customer Rejects Terms
Reject Terms
Canel Provisional Reservation
Take Customer Details
Store Customer Details
Send Credit Details to Agency
Credit OK?
No - Credit No Good
Notify Customer of Credit Problem
Deny Reservation
Notify Customer of Credit Rejection
Yes
Provide Customer with Reservation Number
Provide Registration Number
Reservation Number Granted for Provisionally Booked Room
public void focusLost(ui.event.FocusEvent evt) {
if (evt.getSource() == appTextField) {
ConTest.this.appTextFieldFocusLost(evt);
}
}
statement

Component Diagram Sketch

Architectural sketch of MVC based application



Challenges

- complexity of the problem domain and software
- masking its complexity

  “give illusion of simplicity”
- facing changes

  birth of new technology

  change of conditions or process

  change of requirements

  ....
- fragile

  a bit or byte out of place can cause problem

  we have to be careful while making changes

Challenges

- a large software product is a capital investment

  cannot afford scrap it,

  software maintenance/evolution,

  we write less code, automate?

  reuse the existing code

- communication between team members

  e.g., “million lines of code”

System Modeling

System modeling is the process of developing abstract models of a system

- each model presenting a different view or perspective of that system

System modelling

- helps the analyst to understand the structure and behaviour of the system
  - structure of the system in terms of its components
  - interaction between the components
- are used to communicate with customers

Existing and Planned System Models

Models of the existing system are used during requirements engineering

- clarify what the existing system does and can be used as a basis for discussing its strengths and weaknesses
- lead to requirements for the new system

Models of the new system are used during requirements engineering to help explain the proposed requirements to other system stakeholders

- these models are used 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

System Perspectives

- **external perspective**
  - models the context or environment of the system
- **interaction perspective**
  - models the interactions
    - between a system and its environment
    - between the components of a system
- **structural perspective**
  - models the organization of a system or the structure of the data that is processed by the system
- **behavioral perspective**
  - models the dynamic behavior of the system and how it responds to events

Context Models

- Context models
  - 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.
- Architectural models show the system and its relationship with other systems

System Boundaries

System boundaries

- define what is inside and what is outside the system
- 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

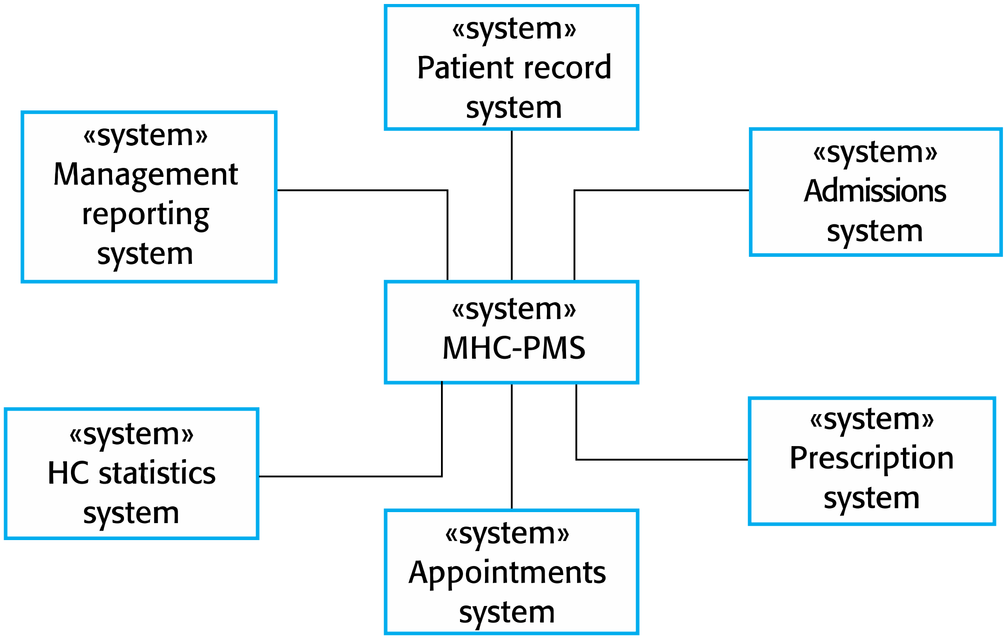
Defining a system boundary is a political judgment

- there may be pressures to develop system boundaries that increase / decrease the influence or workload of different parts of an organization

The Context of the MHC-PMS

Example

- Mental Health Care Patient Management System (MHC-PMS)



Process Perspective

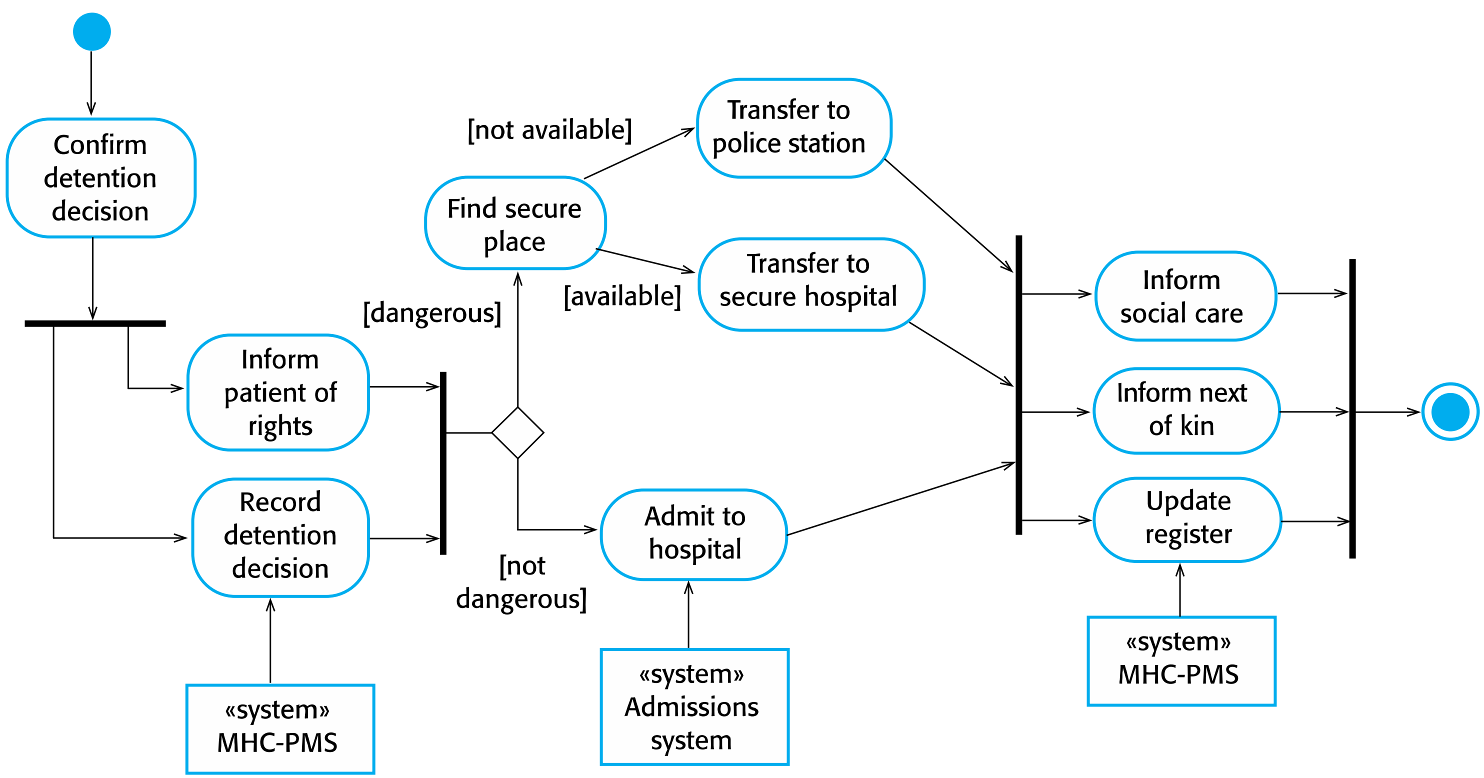
Context models simply show the other systems in the environment, not how the system being developed is used in that environment

- 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
  - WS-BPEL: Web Services Business Process Execution Language

Process Model of involuntary detention

Example

- MHC-PMS involuntary detention



Interaction Models

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

Structural Models

Structural models display the organization of a system in terms of

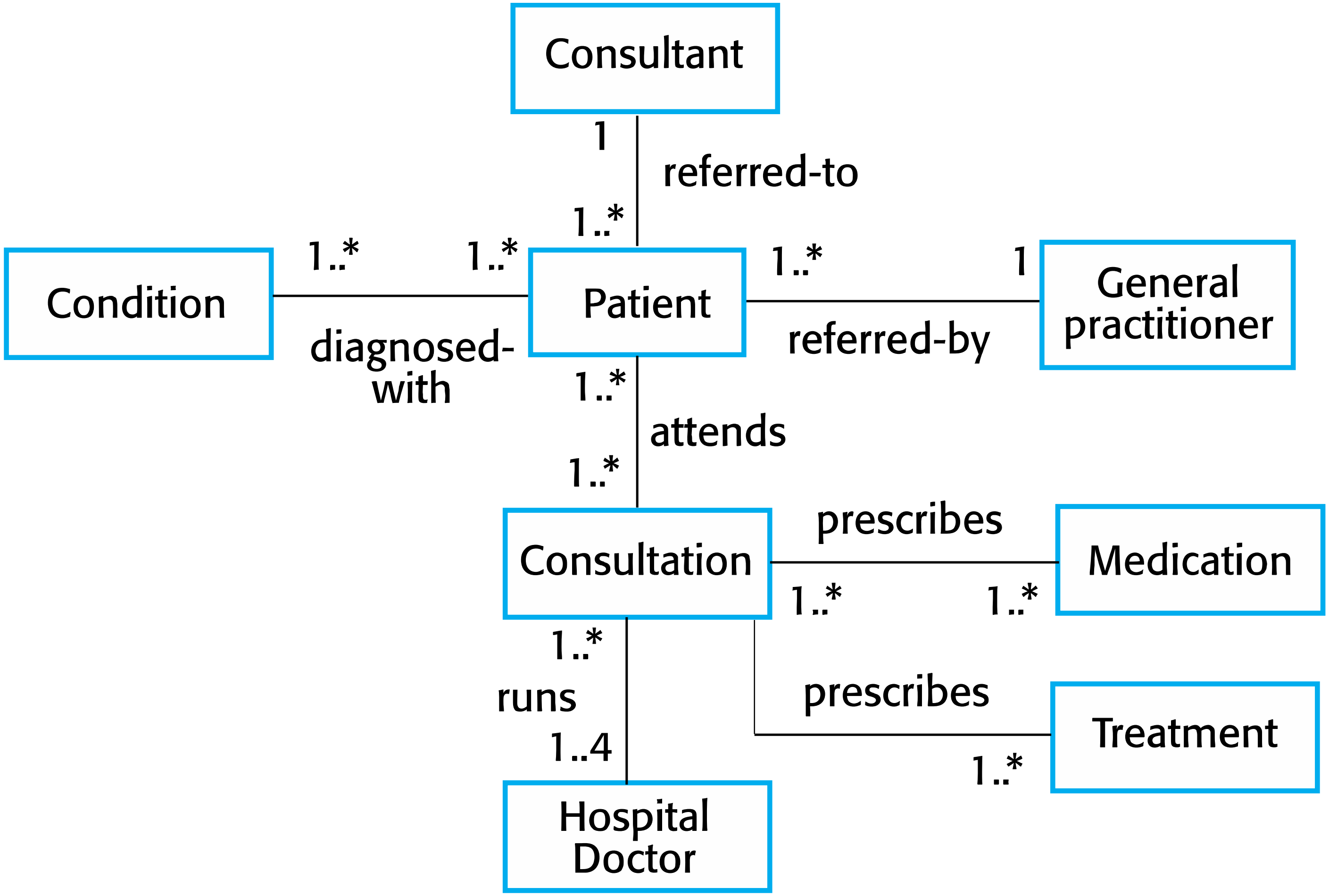
- components that make up that system
- their relationships

Structural models may be

- static models, which show the structure of the system design
- dynamic models, which show the organization of the system when it is executing

Structural models of a system are created when discussing and designing the system architecture

UML Class Diagram of MHC-PMS



Behavioral Models

Behavioral models are models of the dynamic behavior of a system as it is executing

- 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

Data-processing systems that are primarily driven by data

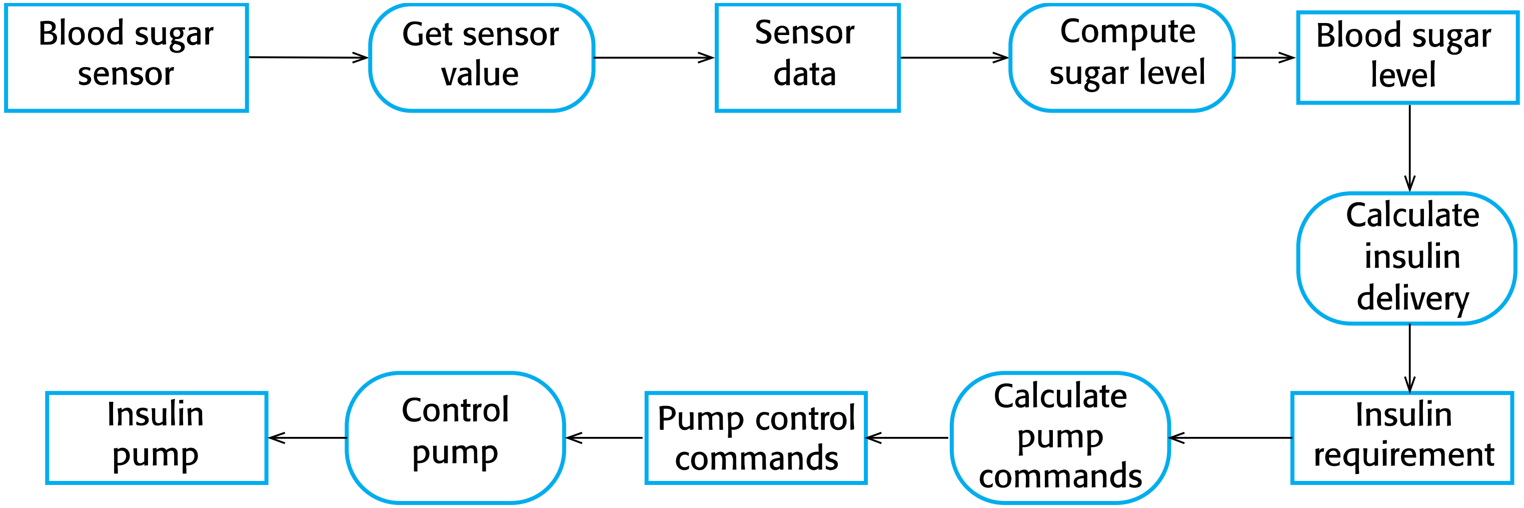
- controlled by the data input to the system, with relatively little external event processing
- e.g., business systems

Data-driven models represent

- sequence of actions involved in processing input data
- generating an associated output
- useful during the analysis of requirements as they can be used to show end-to-end processing in a system

Data-driven Modeling: Example

An activity model of the insulin pump's operation



Event-driven Modeling

In event-driven systems there is minimal data processing

- e.g., real-time systems
  - a landline phone switching system responds to events such as ‘receiver off hook’ by generating a dial tone

Event-driven modeling shows

- how a system responds to external and internal events
- based on the assumption that a system has a finite number of states
  - events (stimuli) may cause a transition from one state to another

State Machine Models

State machines

- model the behaviour of the system in response to external and internal events
- show the system’s responses to stimuli

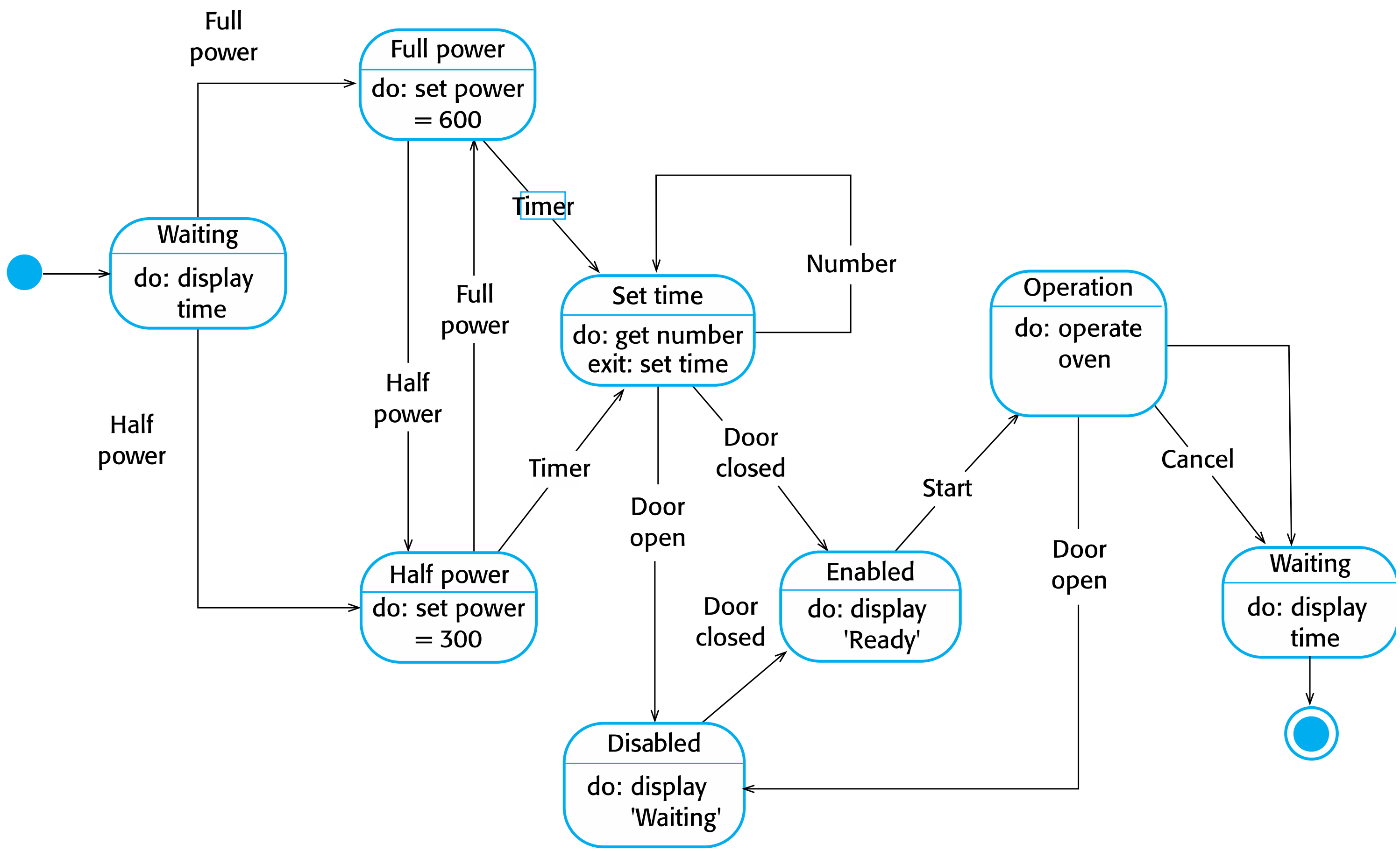
State machine models show

- system states as nodes and events as arcs between these nodes
- when an event occurs, the system moves from one state to another.

UML state diagrams are used to represent state machine models

State Machine Models: Example

State diagram of a microwave oven



Classic Modelling Techniques

Models range from formal (mathematical/precise semantics) to informal (textual description)

- UML are rigorous models (formal semantics) based on diagram and text
  - diagrams can be subjected to all kinds of consistency checks
  - even generate executable code from diagrams

Techniques before and after OO design

- traditional techniques focus on identifying the functions of the system
- object-oriented techniques focus on identifying and interrelating the objects that play a role in the system

Classic Modelling Techniques

- Entity-relationship modelling(ERM)
  - data modelling technique
  - UML classs diagrams are based on ERM
- Finite state machines (FSMs)
  - model states and state transitions
  - e.g., UML state diagrams
- Data flow diagrams (DFD)
  - model a system as a set of processes and data flows that connect these processes
  - result from a top-down decomposition process
  - UML sequence diagrams
- Class—Responsibility-Collaborators (CRC) cards
  - simple requirements elicitation tool

Model-driven Engineering

Model-driven engineering (MDE)

- an approach to software development where models rather than programs are the principal outputs of the development process
- programs are then generated automatically from the models
- this raises the level of abstraction in software engineering
  - engineers no longer have to be concerned with programming language details
  - specifics of execution platforms

Model-driven Engineering

Model-driven engineering is still at an early stage of development

- it is unclear whether or not it will have a significant effect on software engineering practice

Pros

- allows systems to be considered at higher levels of abstraction
- automatic generation of code means that it is cheaper to adapt systems to new platforms

Cons

- models for abstraction and not necessarily right for implementation
- savings from generating code may be outweighed by the costs of developing translators for new platforms

Model-driven Architecture

Model-driven architecture (MDA)

- was the precursor of more general model-driven engineering

MDA is a model-focused approach to software design and implementation

- uses a subset of UML models to describe a system
- models at different levels of abstraction are created
  - from a high-level, platform independent model, it is possible to generate a working program without manual intervention

MDA: Types of model

Computation independent model (CIM)

- model the domain abstractions used in a system
- CIMs are sometimes called domain models

Platform independent model (PIM)

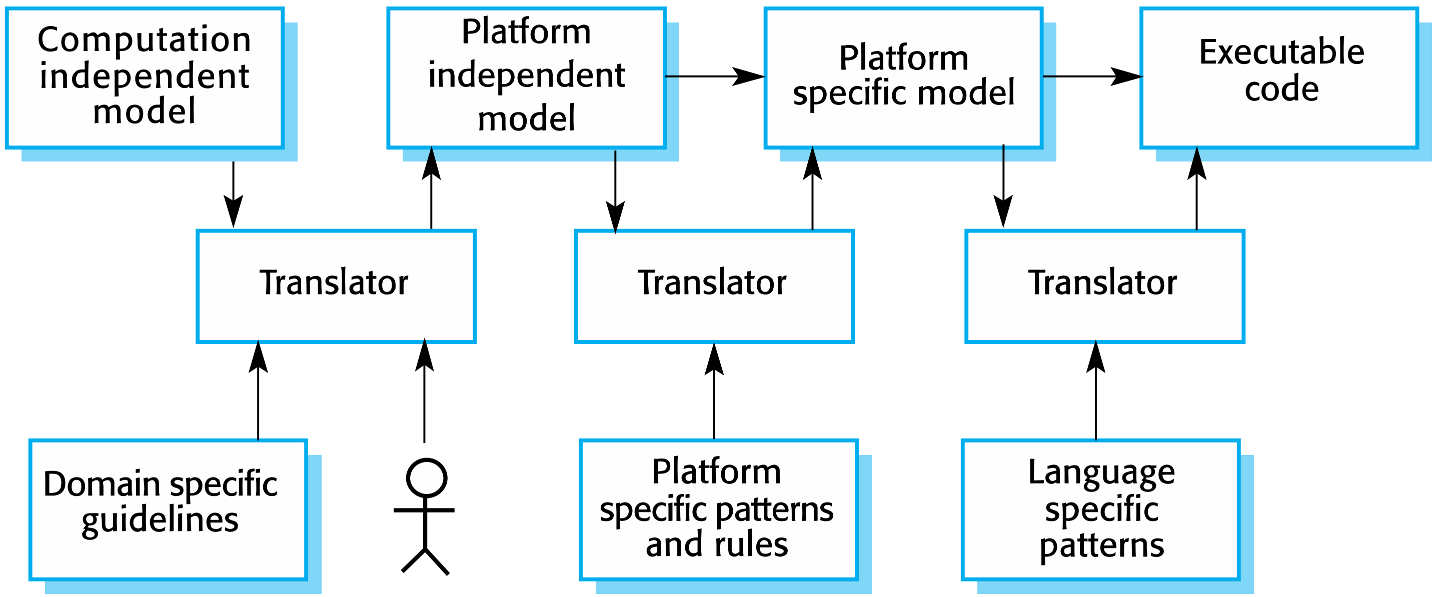
- model the operation of the system without reference to its implementation
- PIM is usually described using UML models
  - show the static system structure and how it responds to external and internal events.

MDA: Types of model

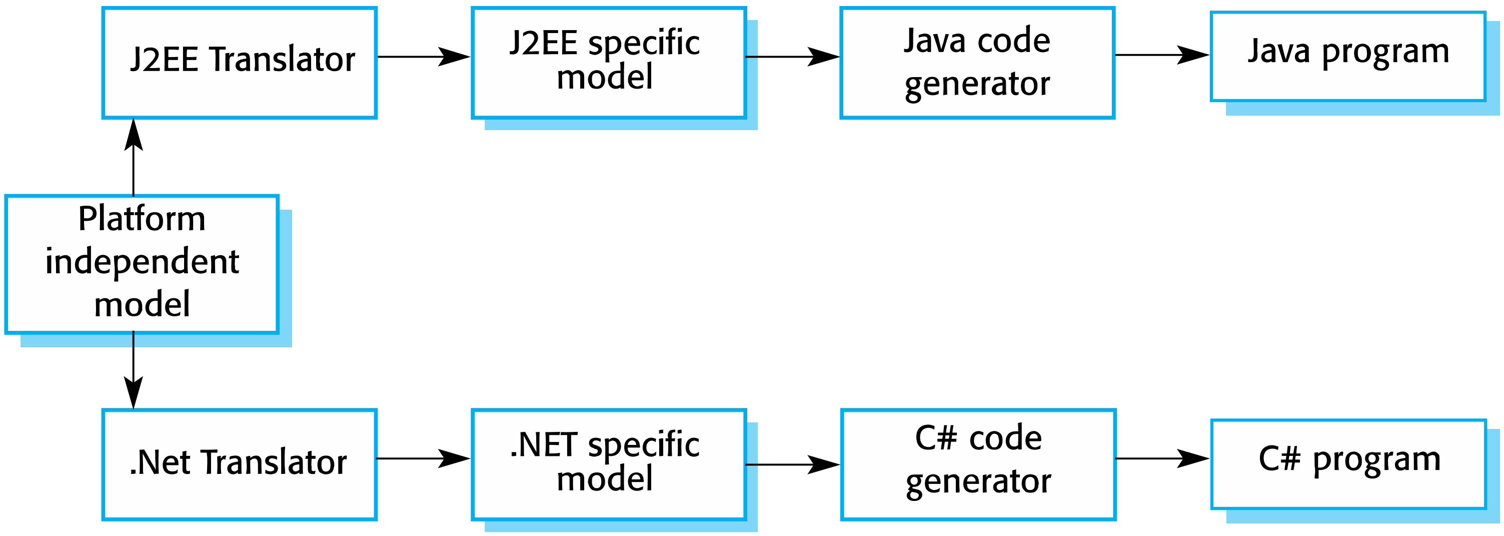
Platform specific models (PSM)

- transformations of the platform-independent model with a separate PSM for each application platform
- there may be layers of PSM
  - each layer adding some platform-specific detail

MDA Transformations



Multiple Platform-specific Models



Models – Key Points

- A **model** is an abstract view of a system that ignores system details
  - complementary system models can be developed to show the system's context, interactions, structure and behaviour
- **Context models** show how a system that is being modeled is positioned in an environment with other systems and processes.
- **Interactions models** represent interactions between users and systems, and between system components
- **Structural models** show the organization and architecture of a system

Models – Key Points

- **Behavioral models** describe the dynamic behavior of an executing system
  - from the perspective of the data processed by the system
  - from the perspective of the events that stimulate responses from a system
- **Model-driven engineering** is an approach to software development in which a system is represented as a set of models that can be automatically transformed to executable code