Modelling Languages

Raimundas Matulevičius

Institute of Computer Sciences
University of Tartu
“An abstraction externalised in a professional language

A model is assumed to be simpler, resemble, and have the same structure and way of functioning as the phenomena it represents”

Language

A set of symbols

- **Syntax** – the graphemes being the smallest units capable of causing contrast in meaning, a set of words constituting the vocabulary, rules to form sentences
- **Semantics** – agreed definitions of what the different sentences mean

Professional language is used by a set of persons working in a certain kind of area or in a scientific discipline

A set of symbols

Syntax – the graphemes being the smallest units capable of causing contrast in meaning, a set of words constituting the vocabulary, rules to form sentences

Semantics – agreed definitions of what the different sentences mean

Modelling Perspective

System model

Modelling Perspective

Structural perspective

Functional perspective

Behavioral perspective

System model

Goal and rule perspective

Actor and role perspective

Topological perspective

Modelling Languages

- UML Class diagrams
- Activity diagrams
- BPMN
- System model
- Use cases
- Notations with topology

Modelling Languages

- UML Class diagrams
- BPMN
- System model
- Activity diagrams
- Use cases
- Notations with topology

UML Class diagrams

- Class
- Association
- Association class
- Aggregation
- Generalisation / Classification

- Prof. Steve Easterbrook, Requirements engineering course, University of Toronto
Classes

• A class describes a group of objects with
  • similar properties (attributes),
  • common behaviour (operations),
  • common relationships to other objects,
  • and common meaning ("semantics").

• Examples
  • employee: has a name, employee# and department; an employee is hired, and fired; an employee works in one or more projects
Finding classes

• **Finding classes source data**
  • Look for nouns and noun phrases in stakeholders’ descriptions of the problem
    • include in the model if they explain the nature or structure of information in the application

• **Finding classes from other sources**
  • Reviewing background information
  • Users and other stakeholders
  • Analysis patterns

• **It’s better to include many candidate classes at first**
  • You can always eliminate them later if they turn out not to be useful
  • Explicitly deciding to discard classes is better than just not thinking about them
Selecting classes

• **Discard classes for concepts which:**
  • Are beyond the scope of the analysis;
  • Refer to the system as a whole;
  • Duplicate other classes;
  • Are too vague or too specific
    • e.g. have too many or too few instances

• External entities that produce or consume information essential to the system should be included as classes
Objects vs. Classes

• The instances of a class are called objects

<table>
<thead>
<tr>
<th>Fred_Bloggs:Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>name: Fred Bloggs</td>
</tr>
<tr>
<td>Employee #: 234609234</td>
</tr>
<tr>
<td>Department: Marketing</td>
</tr>
</tbody>
</table>

• Two different objects may have identical attribute values (like two people with identical name and address)

• Objects have associations with other objects
  • E.g. Fred_Bloggs:employee is associated with the KillerApp:project object
  • But we will capture these relationships at the class level (why?)
Associations

- Objects do not exist in isolation from one another
  - A relationship represents a connection among things

- Types of relationships
  - Association
  - Aggregation and Composition
  - Generalization

- Class diagrams show classes and their relationships
Multiplicity

• **Optional** (0 or 1) 0..1
• **Exactly one** 1 = 1..1
• **Zero or more** 0..* = *
• **One or more** 1..*
• **A range of values** 1..6
• **A set of ranges** 1..3, 7..10, 15, 19..*
A staff member has zero or more clients on His/her clientList. Multiplicity
A client has exactly one staffmember as a contact person.

Name of the association

:StaffMember
staffName
staff#
staffStartDate

liaises with

0..*
clientList

:Client
companyAddress
companyEmail
companyFax
companyName
companyTelephone

Role
The staffmember’s role in this association is as a contact person.

Direction
The “liaises with” association should be read in this direction.

Role
The clients’ role in this association is as a clientList.
Examples
Association classes

- Sometimes the association is itself a class
  - because we need to retain information about the association
  - and that information doesn’t naturally live in the classes at the ends of the association
  - E.g. a “title” is an object that represents information about the relationship between an owner and her car

```plaintext
:car
VIN (vehicle Id Number)
YearMade
Mileage

owns
0..*

:person
Name
Address
DriversLicenceNumber
PermittedVehicles

:title
yearbought
initialMileage
PricePaid
LicencePlate#
```

owner
1

owns
0..*

owner
1
Aggregation and Composition

- **Aggregation**
  - This is the “Has-a” or “Whole/part” relationship

- **Composition**
  - Strong form of aggregation that implies ownership:
    - if the whole is removed from the model, so is the part
    - the whole is responsible for the disposition of its parts
Generalisation

- Subclasses **inherit** attributes, associations, & operations from the superclass
- A subclass may override an inherited aspect
- Superclasses may be declared `{abstract}`, meaning they have no instances
  - Implies that the subclasses cover all possibilities
Generalisation

• **Look for generalisations in two ways**
  - **Top Down**
    • You have a class, and discover it can be subdivided
  - **Bottom Up**
    • You notice similarities between classes you have identified

• **But don’t generalise just for the sake of it**
  - Be sure that everything about the superclass applies to the subclasses
  - Don’t add subclasses or superclasses that are not relevant to your analysis
Example
Another Example
And yet another example
Modelling Languages

- UML Class diagrams
- Activity diagrams
- BPMN
- System model
- Use cases
- Notations with topology

BPMN diagrams
**• Approach**
  - What organization needs to do to achieve their business objectives?

**• Advantages**
  - Reasonably intuitive
  - Explicit declaration of business activities, processes and sub-processes

**• Disadvantages**
  - Captures only a dynamic picture
  - Not focused on the business support by technology
BPMN diagrams

  - [http://fundamentals-of-bpm.org](http://fundamentals-of-bpm.org)
Modelling Languages

UML Class diagrams

Activity diagrams

BPMN

System model

*i*/ Tropos

Use cases

Notations with topology

Moving towards specification

• What functions will the new system provide?
  • How will people interact with it?
  • Describe functions from a user’s perspective

• Use Cases
  • Used to show:
    • the functions to be provided by the system
    • which actors will use which functions
  • Each Use Case is:
    • a pattern of behavior that the new system is required to exhibit
    • a sequence of related actions performed by an actor and the system via a dialogue

• An actor:
  • anything that needs to interact with the system:
    • a person
    • a role that different people may play
    • another (external) system
Use case diagram

• Capture the relationships between actors and Use Cases
Notation for Use Cases

Staff contact

Actor

Communication association

System boundary

Use case

Change client contact
Example

Financial Module

- Add new staff member
- Add new staff grade
- Change rate for the client
- Change grade for the staff member
- Calculate staff bonuses

Accountant
<<extends>> and <<includes>>

- **<<extend>>:** one use case adds behaviour to a base case
  - used to model a part of a use case that the user may see as optional system behavior
  - also models a separate sub-case which is executed conditionally

- **<<include>>:** one use case invokes another (like a procedure call)
  - used to avoid describing the same flow of events several times
  - puts the common behavior in a use case of its own
Another example

Car System
Identifying Actors

• **Ask the following questions:**
  • Who will be a **primary user** of the system? (primary actor)
  • Who will need support from the system to do her daily tasks?
  • Who will maintain, administrate, **keep the system working**? (secondary actor)
  • Which **hardware devices** does the system need?
  • With which **other systems** does the system need to interact with?
  • Who or what **has an interest** in the results that the system produces?

• **Look for:**
  • the users who directly use the system
  • also others who need services from the system
Finding Use Cases

• For each actor, ask the following questions:
  • Which **functions** does the actor require from the system?
  • What does the **actor need to do**?
  • Does the actor need to **read, create, destroy, modify, or store** some kinds of information in the system?
  • Does the actor **have to be notified** about events in the system?
  • Does the **actor need to notify** the system about something?
  • What do those events require in terms of system functionality?
  • Could the actor’s daily work be simplified or made more efficient through new functions provided by the system?
Documenting Use Cases

• **For each use case:**
  • prepare a “flow of events”
  • document from an actor’s point of view
  • describe what the system must provide to the actor when the use case is executed

• **Typical contents**
  • How the use case starts and ends
  • Normal flow of events
  • Alternate flow of events
  • Exceptional flow of events

• **Documentation style**
  • Textual use case description
  • Sequence diagrams
### Use case template (1)

(Wiegers, 2004)

<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Name:</td>
<td></td>
</tr>
<tr>
<td>Created By:</td>
<td>Last Updated By:</td>
</tr>
<tr>
<td>Date Created:</td>
<td>Date Last Updated:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actors:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td></td>
</tr>
<tr>
<td>Trigger:</td>
<td></td>
</tr>
<tr>
<td>Preconditions:</td>
<td>1.</td>
</tr>
<tr>
<td>Postconditions:</td>
<td>1.</td>
</tr>
<tr>
<td>Normal Flow:</td>
<td>1.</td>
</tr>
<tr>
<td>Alternative Flows:</td>
<td></td>
</tr>
<tr>
<td>Exceptions:</td>
<td></td>
</tr>
<tr>
<td>Includes:</td>
<td></td>
</tr>
<tr>
<td>Priority:</td>
<td></td>
</tr>
<tr>
<td>Frequency of Use:</td>
<td></td>
</tr>
<tr>
<td>Business Rules:</td>
<td></td>
</tr>
<tr>
<td>Special Requirements:</td>
<td></td>
</tr>
<tr>
<td>Assumptions:</td>
<td></td>
</tr>
<tr>
<td>Notes and Issues:</td>
<td></td>
</tr>
</tbody>
</table>
Use case templates (2)
(Wiegers, 2004)

<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th>Use Case Name:</th>
<th>Created By:</th>
<th>Last Updated By:</th>
<th>Date Created:</th>
<th>Date Last Updated:</th>
</tr>
</thead>
</table>

- **Use Case ID**: a unique integer sequence number identifier
- **Use Case Name**: a concise, results-oriented name for the use case
- **Created By**: the name of the person who initially documented this use case
- **Date Created**: the date on which the use case was initially documented
- **Last Updated By**: the name of the person who performed the most recent update to the use case description
- **Date Last Updated**: the date on which the use case was most recently updated
Use case templates (3)
(Wiegers, 2004)

<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th>Use Case Name:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created By:</td>
<td>Last Updated By:</td>
</tr>
<tr>
<td>Date Created:</td>
<td>Date Last Updated:</td>
</tr>
</tbody>
</table>

- **Actors**: a person or other entity external to the software system being specified who interacts with the system and performs use cases to accomplish tasks
- **Description**: the reason for and outcome of this use case, the sequence of actions and the outcome of executing the use case
- **Trigger**: the event that initiates the use case
- **Pre-condition**: list any activities that must take place, or any conditions that must be true, before the use case can be started
- **Post-condition**: the state of the system at the conclusion of the use case execution
**Use case templates**  (4)
(Wiegers, 2004)

<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Name:</td>
<td></td>
</tr>
<tr>
<td>Created By:</td>
<td>Last Updated By:</td>
</tr>
<tr>
<td>Date Created:</td>
<td>Date Last Updated:</td>
</tr>
</tbody>
</table>

- **Normal flow**: a detailed description of the user actions and system responses that will take place during execution of the use case under normal, expected conditions.

- **Alternative flows**: other, legitimate usage scenarios that can take place.

- **Exceptions**: any anticipated error conditions that could occur during execution of the use case.

- **Includes**: any other use cases that are included (“called”) by this use case.

- **Priority**: the relative priority of implementing the functionality required to allow this use case.
Use case templates (5)
(Wiegers, 2004)

<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Name:</td>
<td></td>
</tr>
<tr>
<td>Created By:</td>
<td>Last Updated By:</td>
</tr>
<tr>
<td>Date Created:</td>
<td>Date Last Updated:</td>
</tr>
</tbody>
</table>

- **Frequency of use**: the number of times this use case will be performed by the actors per some appropriate unit of time
- **Business rules**: any business rules that influence this use case
- **Special Requirements**: any additional requirements (e.g., quality) for the use case that may need to be addressed during design or implementation
- **Assumptions**: any assumptions that were made in the analysis that led to accepting this use case
- **Notes and Issues**: any additional comments
Modelling Languages

- UML Class diagrams
- BPMN
- System model
- Use cases
- Activity diagrams
- i*/ Tropos
- Notations with topology

Activity diagrams

- **Activity diagrams**
  - Represent dynamic behaviour
  - Stepwise activities and actions with support for choice, iteration and concurrency
  - Intended for computational and organisational processes
Modelling Languages

Goals

• **Approach**
  - Focus on *why* a system is required
  - Use goal refinement to arrive at specific requirements
  - Goal analysis
    - document, organize and classify goals
  - Goal hierarchies show refinements and alternatives

• **Advantages**
  - Reasonably intuitive
  - Explicit declaration of goals provides sound basis for conflict resolution

• **Disadvantages**
  - Captures a static picture - what if goals change over time?
  - Can regress forever up (or down) the goal hierarchy

• **Goals:**
  - Describe functions that must be carried out

• **Actors:**
  - Owners of goals

• **Tips:**
  - Multiple sources - better goals
  - Associate stakeholders with each goal
  - Use scenarios to explore how goals can be met
Goal Modeling

• **(Hard) Goals:**
  - Describe functions that must be carried out. E.g.
    • Satisfaction goals
    • Information goals

• **Softgoals:**
  - Cannot really be fully satisfied. E.g.
    • Accuracy
    • Performance
    • Security
    • ...

• **Also classified temporally:**
  - Achieve/ Cease goals
    • Reach some desired state eventually
  - Maintain/ Avoid goals
    • Keep some property invariant
  - Optimize
    • A criterion for selecting behaviours

• **Agents:**
  - Owners of goals
  - Choice of when to ascribe goals to agents:
    • Identify agents first, and then their goals
    • Identify goals first, and then allocate them to agents during operationalization

• **Modelling Tips:**
  - Multiple sources yield better goals
  - Associate stakeholders with each goal
    • reveals viewpoints and conflict
  - Use scenarios to explore how goals can be met
  - Explicit consideration of obstacles helps to elicit exceptions
Goal analysis

**Relationships between goals:**
- One goal **helps** achieve another (+)
- One goal **hurts** achievement of another (-)
- One goal **makes** another (++)
  - Achievement of goal A guarantees achievement of goal B
- One goal **breaks** another (--)  
  - Achievement of goal A prevents achievement of goal B

**Goal Elaboration:**
- “**Why**” questions explore higher goals (context)
- “**How**” questions explore lower goals (operations)
- “**How else**” questions explore alternatives
http://istar.rwth-aachen.de/

Tropos
Secure Tropos
...
• **Strategic dependency model**
  - used to express the network of intentional, strategic relationships among actors

• **Strategic rationale model**
  - used to express the rationales behind dependencies
Strategic dependency model (1)

- **Actor**
  - carries out actions to achieve goals

- **Role**
  - characterization of the behavior of a social actor within some context
  - a set of *roles* typically played by one *agent*

- **Agent**
  - actor with concrete, physical manifestations, such as a human individual
  - an *agent* occupies a *position*

- **Position**
  - used between a role and an agent
  - a *position* is said to cover a *role*
Strategic dependency model

- **Dependees**: Actor who is depended upon in a dependency relationship.
- **Depender**: The depending actor on a dependency relationship.
- **Dependum**: Element around which a dependency relationship centers.

![Dependency Model Diagram](image)
Strategic dependency model (3)

- **Goal dependency**
  - the depender depends on the dependee to bring about a certain state of affairs in the world

- **Task dependency**
  - the depender depends on the dependee to carry out an activity

- **Resource dependency**
  - the depender depends on the dependee for the availability of an entity

- **Softgoal dependency**
  - a depender depends on the dependee to perform some task that meets a softgoal
Strategic dependency model (4)
Strategic rationale model (1)

• **Actor boundaries**
  • all of the elements within a boundary for an actor are explicitly desired by that actor
  • to achieve these elements, an actor must depend on the intentions of other actors

• **Goal (hardgoal)**
  • intentional desire of an actor

• **Softgoal**
  • criteria for the goal's satisfaction are not clear-cut
  • judged to be sufficiently satisfied from the point of view of the actor

• **Task**
  • actor wants to accomplish some specific task, performed in a particular way

• **Resource**
  • actor desires the provision of some entity, physical or informational
Strategic rationale model (2)

- **Means-ends**
  - a relationship between an end, and a means for attaining it
  - "means" is expressed in the form of a task
  - "end" is expressed as a goal

- **Decomposition**
  - task can be decomposed into four types of elements: a subgoal, a subtask, a resource, and/or a softgoal
Strategic rationale model (3)

• **Contribution**
  - **Make**: strong enough to satisfice a softgoal
  - **Some+**: positive with unknown strength
  - **Help**: not sufficient by itself to satisfice the softgoal
  - **Unknown**: polarity is unknown
  - **Break**: sufficient enough to deny a softgoal
  - **Some-**: negative with unknown strength
  - **Hurt**: not sufficient by itself to deny the softgoal
  - **Or**: satisficed if any of the offspring are satisficed
  - **And**: satisficed if all of the offspring are satisficed
Strategic rationale model (4)
Asset Modelling
Asset-Related Concepts

• Specify important assets to protect, define criteria to guarantee asset security
  • Asset – anything that has value to the organisation and is necessary for achieving its objectives
  • Business asset – information, process, skill inherent to the business of the organisation that has value to the organisation in terms of its business model and is necessary for achieving its objectives
  • System asset – a component or part of the system that has value to the organisation and is necessary for achieving its objectives and supporting business assets

• Security criterion - property or constraint on business assets
• Security objective is defined using security criteria on business assets
Processing of Information

Everything that IT does, reduces to six functions

• Capturing information
  • Keyboard, bar code reader, digital camera

• Transmitting information
  • Wired-, wireless-phone

• Storing information
  • Hard disk, memory card, internet

• Retrieving information
  • From any storage device

• Manipulating information
  • Calculations, combinations of data

• Displaying information
  • Monitor, printer
Functional Decomposition

• **User interaction**
  • Interfacing and/or interacting with users

• **Data/storage management**
  • Storing and management of applications or information

• **Resource management**
  • Resource allocation, global scheduling, process migration,
  • Dynamic configuration of active software components

• **Distribution control**
  – Component collaboration
  – Coordination of local/remote execution
  – Synchronization/concurrency control

• **Communication**
  – Network communication

**Addressing**
  – Address, identifier and/or name allocation, distribution and discovery/lookup

© Springer International Publishing AG 2017
Intelligent Transportation Systems

**PERCEPTION LAYER**
- Sensing
- Vision
- Positioning
- Actuating

**NETWORK LAYER**
- In-Vehicle
- Vehicle to Vehicle
- Vehicle to Infrastructure

**APPLICATION LAYER**
- Computing/Server
- Data Storage
- Human
Security Criterion

• Property or constraint on business assets that characterises their security needs
• Act as indicators to assess the significance of a risk

- **Confidentiality** – a property of being made not available or disclosed to unauthorized individuals, entities or processes

- **Integrity** – a property of safeguarding the accuracy and completeness of assets
  - **Accuracy** could be threatened by (unauthorised or undesirable) update or tampering
  - **Completeness** could be threatened using altering or deletion

- **Availability** – a property of being accessible and usable upon demand by an authorised entity
Security objective

• Defined using security criteria on business assets
  • Confidentiality of the technical plans
  • Integrity of the structure calculation process
  • Availability of ticket booking service
### Literature on Autonomous Vehicles:


Business assets | Description
---|---
Video data | Video from surrounding environment
Picture data | Pictures from surrounding environment
Vehicle location data | Current location of the vehicle
Vehicle travel data | Routes with current
Working vehicle data | Vehicle speed, direction (and similar)
Ultrasonic sensor data | Ultrasonic data about surroundings
Radar data | Data from radar
Surrounding environment data | Data about the surrounding environment and objects
Inertial measurements | Vehicle speed, angle, and location
**Business assets**

**Communication data**

**Description**

Data and messages exchanged by different components
<table>
<thead>
<tr>
<th>Business assets</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map data</td>
<td>Map used for autonomous driving</td>
</tr>
<tr>
<td>Fused data</td>
<td>Combined data from perception layer</td>
</tr>
<tr>
<td>Computing data</td>
<td>Results from analysing the fused data</td>
</tr>
<tr>
<td>Actuation commands data</td>
<td>Commands generated to be sent to actuation module</td>
</tr>
<tr>
<td>Decision maker</td>
<td>Software for making driving decisions</td>
</tr>
<tr>
<td>Driving planner</td>
<td>Software for planning out the route used</td>
</tr>
<tr>
<td>System software</td>
<td>All software used for autonomous driving</td>
</tr>
</tbody>
</table>

**Application Layer**

- **Internal storage**
  - StoresData
  - GetsData
  - TransmitsData

- **Computing unit**
  - ConnectedTo Router
  - Commands

- **Map storage**
  - Traffic data storage
  - Traffic sign storage

- **Display**
  - 1..* Physical input ports

- **Actuation module**
  - Uses Electric control unit

**Perception Layer**

- **Map data**
  - ConnectedTo Map storage

- **Fused data**
  - ConnectedTo Traffic data storage, Traffic sign storage

- **Computing data**
  - ConnectedTo Display

- **Actuation commands data**
  - ConnectedTo Actuation module

- **Decision maker**
  - ConnectedTo Computing unit

- **Driving planner**
  - ConnectedTo Decision maker

- **System software**
  - ConnectedTo Driving planner, Map data, Fused data, Computing data, Actuation commands data, Decision maker, Driving planner
Thank you!