Introduction and Vocabulary

Software Modelling

Modelling structure

- (simplified) Class & Object diagrams
- (simplified) Object Constraint Logic (OCL)

Principles of Design

- modularity, separation of concerns
- information hiding and data encapsulation
- abstract data types, object orientation

Design Patterns

Modelling behaviour

- Communicating Finite Automata (CFA)
- Uppaal query language

Model-driven/-based Software Engineering

Vocabulary

- System, Architecture, Design
- Modelling
- Software Modelling
- views & viewpoints
- the 4+1 view
- Class Diagrams
- concrete syntax,
  abstract syntax,
  semantics: system states.
- object diagrams at work.
- Object Diagrams
- concrete syntax,
  dangling references,
  partial vs. complete,
  object diagrams at work.

architecture — The fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution. IEEE 1471 (2000)

design — (1) The process of defining the architecture, components, interfaces, and other characteristics of a system or component. (2) The result of the process in (1). IEEE 610.12 (1990)
Architectural Description is the result of the process of defining the architecture, components, interfaces, and other characteristics of a system or component. A set of software units and their relations, if they together serve a common purpose. A model – document, product or other artifact – to communicate and record a system's architecture. An architectural description conveys a set of views each of which depicts the system by describing domain concerns. The fundamental organization of a system embodied in its components, their relations to each other and to the environment, and the principles guiding its design and evolution.

A boundary across which two independent entities meet and interact or communicate with each other. The interconnection (of component) may be a linkage editor, or executive routine.

A logically separable part of a program. A set of operations and data visible from the outside only in so far as explicitly permitted linkage editor, or executive routine.

A program unit that is discrete and identifiable with respect to compiling, combining and may be subdivided into other components. A component may be hardware or software and may be subdivided into other components. A component may be hardware or software and may be subdivided into other components.

—a set of operations and data visible from the outside only in so far as explicitly permitted linkage editor, or executive routine.

—a logically separable part of a program.

—a program unit that is discrete and identifiable with respect to compiling, combining and may be subdivided into other components. A component may be hardware or software and may be subdivided into other components. A component may be hardware or software and may be subdivided into other components.

—a model – document, product or other artifact – to communicate and record a system's architecture. An architectural description conveys a set of views each of which depicts the system by describing domain concerns. The fundamental organization of a system embodied in its components, their relations to each other and to the environment, and the principles guiding its design and evolution.

—a set of software units and their relations, if they together serve a common purpose.

—a model – document, product or other artifact – to communicate and record a system's architecture. An architectural description conveys a set of views each of which depicts the system by describing domain concerns. The fundamental organization of a system embodied in its components, their relations to each other and to the environment, and the principles guiding its design and evolution.

—a set of structural or organizational aspects of a system that are relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.

—a structural or organizational aspect of a system that is relevant to the problem domain, such as the roles played by actors, the entities that are manipulated in the system, the information flows among actors and entities, and the rules that govern the actions of actors and entities.
The structure of something is the set of relations between its parts. Something not built from (recognisable) parts is called unstructured.

Design...

(i) structures a system into manageable units (yields software architecture),
(ii) determines the approach for realising the required software,
(iii) provides hierarchical structuring into a manageable number of units at each hierarchy level.

Oversimplified process model "Design":

req. design arch. designer
modulespec. impl. programmer
impl. code

Topic Area Architecture & Design: Content

• Introduction and Vocabulary
• Software Modelling
  • model; views / viewpoints; 4+1 view
  • Modelling structure
    • (simplified) Class & Object diagrams
    • (simplified) Object Constraint Logic (OCL)
• Principles of Design
  • modularity, separation of concerns
  • information hiding and data encapsulation
  • abstract data types, object orientation
• Design Patterns
• Modelling behaviour
  • Communicating Finite Automata (CFA)
  • Uppaal query language
• CFA vs. Software
  • Unified Modelling Language (UML)
    • basic state-machines
    • an outlook on hierarchical state-machines
• Model-driven/-based Software Engineering
  • an outlook on hierarchical state-machines

Vocabulary
• System, Architecture, Design
• Modelling
• Software Modelling
  • views & viewpoints
  • the 4+1 view
  • Class Diagrams
    • concrete syntax,
    • abstract syntax,
    • semantics: system states.
  • class diagrams at work,
• Object Diagrams
  • concrete syntax,
  • dangling references,
  • partial vs. complete,
  • object diagrams at work.
Definition. (Folk) A model is an abstract, formal, mathematical representation or description of structure or behaviour of a (software) system.

Definition. (Glinz, 2008, 425) A model is a concrete or mental image (Abbild) of something or a concrete or mental archetype (Vorbild) for something. Three properties are constituent:

(i) the image attribute (Abbildungsmerkmal), i.e. there is an entity (called original) whose image or archetype the model is,
(ii) the reduction attribute (Verkürzungsmerkmal), i.e. only those attributes of the original that are relevant in the modelling context are represented,
(iii) the pragmatic attribute, i.e. the model is built in a specific context for a specific purpose.

Example: Design-Models in Construction Engineering

1. Requirements
   • Shall fit on given piece of land.
   • Each room shall have a door.
   • Furniture shall fit into living room.
   • Bathroom shall have a window.
   • Cost shall be in budget.

2. Design model
   http://wikimedia.org (CC nc-sa 3.0, Ottoklages)

3. System
   http://wikimedia.org (CC nc-sa 3.0, Bobthebuilder82)

Observation (1): Floorplan abstracts from certain system properties, e.g.,
• kind, number, and placement of bricks,
• subsystem details (e.g., window style),
• water pipes/wiring, and
• wall decoration → architects can efficiently work on appropriate level of abstraction

Observation (2): Floorplan preserves/determines certain system properties, e.g.,
• house and room extensions (to scale),
• presence/absence of windows and doors,
• placement of subsystems (such as windows). → find design errors before building the system (e.g. bathroom windows)
• e.g., needs to know which component is running on which host,

A team leader: information needs and concerns is determined by


Process View — A representation of a whole system from the perspective of a related set of viewpoints.

Scenarios

From Process Model to Concrete Process

Examples for (Software) Models?
The (possibly partial) function \( S \) of the computation paths 

\[ S \cdot \rightarrow S \cdot \rightarrow \cdots \] 

of a (possibly infinite) set of (finite or infinite) \( \pi \). 

Computation paths 

• Structure vs. Behaviour / Constructive vs. Reflective 

How are component instances mapped onto infrastructure and hardware units? 

How is the system under development integrated into (or seen by) its environment? 

Which other systems (including users) does it interact with? 

• How and when are components instantiated and how do they work together at runtime? 


How are component instances mapped onto infrastructure and hardware units? 

Example: modern cars, electronic systems. 

Deployment / Physical View 

An Early Proposal: The 4+1 View 

Figure 4: Example model of an architecture.
Class Diagrams
views & viewpoints
Software Modelling
• System, Architecture, Design
Vocabulary

Class Diagrams at work.
• dangling references,
• concrete syntax,
• abstract syntax,
• concrete syntax,
• Class Diagrams

Note.
things are computed
how → "in executing the model or in translating it into executable code.
constructs [of description] contain information needed

"should (or should not) be computed
what → or to set constraints on behavior in preparation for verification.
[description used] to derive and present views of the model,statically or during execution,
constructive and reflective descriptions of behaviour:
reflective

1997, Harel

pretation is called
S of

·

The (possibly partial) function
S of
π

[]

·

[]

·

·

[]

·

·

·

·

·

·

·
Concrete Syntax: Example

```latex
\text{Concrete Syntax: Example}
```

Alternative notation for $C_{0,1}$ and $C^*$ typed attributes:

```latex
D_x : \text{Int}
\text{f}(\text{Int}) : \text{Bool}
```

```latex
get_x() : \text{Int}
```

Alternative lazy notation for alternative notation:

```latex
D_x : \text{Int}
\text{f}(\text{Int}) : \text{Bool}
```

```latex
get_x() : \text{Int}
```

And nothing else!

This is the concrete syntax of class diagrams for the scope of the course.
The class diagram syntax can be used to which map (parts of) the code to class diagram elements.

Once Again: Concrete vs. Abstract Syntax
A more precise class diagram semantics...
• The class diagram syntax can be used to visualize code:
  Provide rules which map (parts of) the code to class diagram elements.

```java
package pac;
import pac.D;

public class C {
  public D n;
  public void print_nx() {
    System.out.printf("%i \n", n.get_x());
  }
  public C() {
  }
}

package pac;
import pac.C;

public class D {
  private int x;
  public int get_x() {
    return x;
  }
  public D() {
  }
}
```

Visualisation of Implementation: (Useless) Example

- Open favourite IDE,
- Open favourite project,
- Press "generate class diagram",
- Wait...
- Wait...
- Wait...
use multiple class diagrams with colour.

Player 0

\[x, y\]

\[
(x, y) \rightarrow \text{Engine}
\]

ca. 35 classes, 46 class diagrams.

A diagram is Ambler
Alternative notation: $\sigma \mapsto \emptyset, x^5 \mapsto \{p\}, \{p\} \mapsto \{p\}$

Int: $x \_ \rightarrow 0, p, n, \{p, n\}, \{p\}, \{p\}$

Object Diagrams
Is the following object diagram complete?

\[
\begin{align*}
\emptyset &= \emptyset \\
x &= \emptyset \\
1 &\in C \\
5 &\in D
\end{align*}
\]

\[
\begin{align*}
\emptyset &\rightarrow \{c, d\} \\
\emptyset &\rightarrow \{n\} \\
\emptyset &\rightarrow \{p\} \\
1 &\rightarrow \emptyset \\
5 &\rightarrow \emptyset
\end{align*}
\]

What about the other way round...?
Object Diagrams at Work

Example: Data Structure

(Schumann et al., 2008)

BaseNode

parent : BaseNode
∗
prevSibling : BaseNode
∗
nextSibling : BaseNode
∗
firstChild : BaseNode
∗
lastChild : BaseNode
∗

Node
data : T

Iterator
operator ++ () : Iterator
operator −− () : Iterator
operator * () : BaseNode

0, 1

Forest
appendTopLevel( data: T )
appendChild( parent : Iterator, data : T )
remove( it : Iterator )
depth( it : Iterator ) : int
depth() : Iterator
begin() : Iterator
size() : int
empty() : bool

Example: Illustrative Object Diagram

(Schumann et al., 2008)

A : Node
B : Node
C : Node
D : Node
E : Node
F : Node

BaseNode

parent : BaseNode
∗
prevSibling : BaseNode
∗
nextSibling : BaseNode
∗
firstChild : BaseNode
∗
lastChild : BaseNode
∗

Node
data : T

Iterator
operator ++ () : Iterator
operator −− () : Iterator
operator * () : BaseNode

0, 1

Forest
appendTopLevel( data: T )
appendChild( parent : Iterator, data : T )
remove( it : Iterator )
depth( it : Iterator ) : int
depth() : Iterator
begin() : Iterator
size() : int
empty() : bool

Examples

Vocabulary
System, Architecture, Design
Modelling
Software modelling
views & viewpoints
the 4+1 view
Class Diagrams
cancrete syntax,
abstract syntax,
semantics: system states.
Class diagrams at work,
Object Diagrams
cancrete syntax,
dangling references,
partial vs. complete,
Object diagrams at work.
Design structures a system into manageable units.

(Software) Model: a concrete or mental image or archetype with image/reduction/pragmatics property.

Towards Software Modelling:

Views and Viewpoints, e.g. 4+1, Structure vs. Behaviour

Class Diagrams can be used to describe system structures graphically, visualise code, define an object system structure $S$.

An Object System Structure $S$ (together with a structure $D$)

defines a set of system states $\Sigma_{DS};$ a system state $\sigma \in \Sigma_{DS}$ can be visualised by an object diagram.