Formal Specification and Verification of Software

UML Class Diagrams by Example

Bernhard Beckert

UNIVERSITÄT KOBLENZ-LANDAU
Classes and Attributes

Semantics

\(I(\text{Person})\) is a (possibly empty) set

\(I(\text{name})\) is a (partial) function from \(I(\text{Person})\) to \(I(\text{String})\)
Associations

Semantics

\(I(\text{review})\) is a relation between \(I(\text{Person})\) and \(I(\text{Paper})\)

**Multiplicity 3 requires that, for all** \(pap \in I(\text{Paper})\),
\[\text{card}\left(\{ \text{pers} \in I(\text{Person}) \mid \text{review}(\text{pers}, \text{pap})\}\right) \in I(3) = \{3\}\]
## Multiplicities

### Semantics

<table>
<thead>
<tr>
<th>$M$</th>
<th>$I(M)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0..1</td>
<td>${0, 1}$</td>
</tr>
<tr>
<td>0..*</td>
<td>$\mathbb{N}$</td>
</tr>
<tr>
<td>*</td>
<td>$\mathbb{N}$</td>
</tr>
<tr>
<td>1..3</td>
<td>${1, 2, 3}$</td>
</tr>
<tr>
<td>7</td>
<td>${7}$</td>
</tr>
<tr>
<td>15..19</td>
<td>${15, 16, 17, 18, 19}$</td>
</tr>
<tr>
<td>1..3, 7, 15..19</td>
<td>${1, 2, 3, 7, 15, 16, 17, 18, 19}$</td>
</tr>
</tbody>
</table>

(i.e., the seperator "," acts as set theoretic union)
Role Names

Semantics

\[ I(\text{referee}) : I(\text{Paper}) \rightarrow \text{Set}(I(\text{Person})) \]

\[ I(\text{paper}) : I(\text{Person}) \rightarrow \text{Seq}(I(\text{Paper})) \]  
(default role name)
Operations

Semantics

Transition from snapshot to snapshot
(relation between sets of snapshots)

Semantics of queries (operations without side-effects)

Partial functions
Subclasses

Semantics

**subclass relation**: \( I(\text{ShortPaper}) \subset I(\text{Paper}) \)

**constraint**: \( I(\text{ShortPaper}) \cap I(\text{LongPaper}) = \emptyset \)
Abstract Classes

Semantics

\[ I(Paper) = I(ShortPaper) \cup I(LongPaper) \]

No “direct” elements in \( I(Paper) \)
Class Attributes

Paper

- authors: [Person]
- number: Int
- totalnumber: Int
- sumpages: Int

Semantics

$I(\text{totalnumber})$ is an element of $I(\text{Int})$

(i.e., Paper.totalnumber is a constant)
Semantics

$I(\text{Report})$ is a subset of $I(\text{Person}) \times I(\text{Paper})$
# Data Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>(i2:Integer):Boolean</code></td>
<td></td>
</tr>
<tr>
<td><code>(i2:Integer):Integer</code></td>
<td></td>
</tr>
<tr>
<td><code>(i2:Real):Real</code></td>
<td></td>
</tr>
<tr>
<td><code>(i2:Integer):Integer</code></td>
<td></td>
</tr>
<tr>
<td><code>(i2:Real):Real</code></td>
<td></td>
</tr>
<tr>
<td><code>/ (i2:Integer):Real</code></td>
<td></td>
</tr>
<tr>
<td><code>/ (i2:Real):Real</code></td>
<td></td>
</tr>
<tr>
<td><code>abs:Integer</code></td>
<td></td>
</tr>
<tr>
<td><code>div(i2:Integer):Integer</code></td>
<td></td>
</tr>
<tr>
<td><code>mod(i2:Integer):Integer</code></td>
<td></td>
</tr>
<tr>
<td><code>max(i2:Integer):Integer</code></td>
<td></td>
</tr>
<tr>
<td><code>min(i2:Integer):Integer</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Type</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>(i2:String):Boolean</code></td>
<td></td>
</tr>
<tr>
<td><code>size:Integer</code></td>
<td></td>
</tr>
<tr>
<td><code>concat(string2: String): String</code></td>
<td></td>
</tr>
<tr>
<td><code>toUpper(string2: String): String</code></td>
<td></td>
</tr>
<tr>
<td><code>toLower(string2: String): String</code></td>
<td></td>
</tr>
<tr>
<td><code>substring(lower: Integer, upper: Integer): String</code></td>
<td></td>
</tr>
</tbody>
</table>
Data Types

Semantics

$I$(Integer) is the same in all snapshots

All operations are queries (no side effects)

No attributes
Enumerations

Semantics

Special kind of data type

\[ I(\text{Marks}) = \{\text{accept}, \text{reject}, \text{weekly accept}, \text{weekly reject}\} \]
Aggregations

Semantics

Same (formal) semantics as an associations

Compositions

Semantics

Same (formal) semantics as an associations
Example: The Composite Pattern

Component
operation()

Leaf
operation()

Composite
operation()
add(c:Component)
remove(c:Component)
getChild(n:int)

children *

B. Beckert: Formal Specification of Software – p.16