Behavior Modeling with UML 2.0

UML standardization within OMG – for Ericsson

better tools

Requirements from ERICSSON developers world-wide

Ericsson UML standardization team

contributing in cooperation with tool vendors

issuing requirements in cooperation with allies
U2 Partners

- **Submitters**
  - Alcatel, CA, Ericsson, Fujitsu, HP, IBM, I-Logix, IONA, Kabira, Motorola, Oracle, Rational, SOFTEAM, Telelogic, Unisys

- **Supporters**

Why UML2.0?

- **for Ericsson, Motorola, Alcatel, Nokia (telecom, realtime)**
  - SDL/MSC only one vendor
  - UML/ROOM (as by RoseRT) only one vendor

  - UML2.0 combining features from these

- **for others**
  - Scalability, modeling of large, complex systems
  - Improvement of existing concepts: activities, components,
  - Completeness: action semantics, formal/precise definition

- **in general**
  - Experiences with UML1.x required an improvement
  - Model Based Development requires a good modeling language
Example - ATM

- **Domain statement**
  - An Automatic Teller Machine (ATM) is a system with mechanical as well as electronic parts. Its purpose is to provide a bank user with cash provided that the user can authenticate herself and she has adequate funds in her bank account.
  - She authenticates herself by presenting a card to the ATM cardreader, and a personal identification number (PIN) through the ATM keyboard.
  - The ATM is connected electronically and possibly through some kind of network to the bank such that the account status may be checked online.
  - The ATM is refilled with cash notes regularly or when the number of specific notes is below some limit.
  - The ATM may also provide foreign currency to the customer.

**ATM: Domain Model 1**

From
Haugen, Ø., B. Møller-Pedersen, and T. Weigert,
*Modeling Embedded Systems in UML 2.0*, in
*The Embedded Systems Handbook*,
Domain Model II

![Diagram showing the ATM and its components: CardReader, Keyboard, Screen, CashDispenser.]

Use Case Model

![Diagram showing the user interactions with the ATM, including Authentication, CashRepository, Currency, and withdrawal.]

User

ATM

CashRepository

Withdrawal «include» Authentication

Currency

Bank
Context model with UML1.x class diagrams

- with plain composition and no encapsulation
- with only provided interfaces on classes

![UML Diagram](Image)

Composite class (incomplete)

- with parts, ports and connectors

![Composite Class Diagram](Image)
Context Model in UML2.0 - I

- composite structure as part of a Collaboration

Context Model in UML2.0 - II

- Including multiplicities on parts
... as part of Packages?   No

Sequence Diagrams (Interactions)

- Sequence Diagrams are
  - simple
  - powerful
  - readable
  - used to describe interaction sequences

- History
  - Has been used for a number of years informally
  - Standardized 1992 in Z.120 (Message Sequence Charts - MSC)
  - Last major revision of MSC is from 1999 (called MSC-2000)
  - Formal semantics of MSC-96 is given in Z.120 Annex B
  - Included in UML from 1999, but in a rather simple variant
**Purpose**

- Emphasizes the interaction between objects indicating that the interplay is the most important aspect
  - Often only a small portion of the total variety of behavior is described improve the individual understanding of an interaction problem
- Sequence Diagrams are used to ...
  - document protocol situations,
  - illustrate behavior situations,
  - verify interaction properties relative to a specification,
  - describe test cases,
  - document simulation traces.

**Simple Sequence Diagram**

- Messages have one send event, and one receive event.
  - The send event must occur before the receive event.
  - The send event is the result of an Action
- Events are strictly ordered along a lifeline from top to bottom 

---

![Sequence Diagram Image](image-url)
Combined fragment example

Combined fragments of Interaction

- We want to express
  - choices: alternative, option, break
  - parallel merge
  - loops
- We may also want to add other operators
  - negation
  - critical region
  - assertion
- Other suggested operators that will not come in UML 2.0
  - interrupt
  - disrupt
References (Interaction Use / Occurrence)

Nested combined fragments
**Interaction Overview Diagram**

**EnterPIN state machine**

```
<<statemachine>>
EnterPIN
n:integer
PIN: integer

sm EnterPIN
send(msg("Give PIN")); n=0; PIN=0

enterDigit
[n<4; digit/PIN=...
\]n++;
PINC\ PIN+dig*10^{4-n} \]
ok
nok
nok

waitOK

[n=4; digit/PIN=...
send(Code(cid,PIN)) \]

definition of exit point
```
Attributes of the ATM

- Statemachine is a Classifier (that is class-like):
  - Attributes
  - Operations (local actions)

```
<<statemachine>>

ATM

authN: integer
clid: integer
sa: Amount

sendMoney(a:Amount)
```

- authN: number of tries
- cid: card id
- sa: selected amount
State machine Withdrawal

sm Withdrawal

- GetAmount
  - cancelled
  - again
  - send(CheckAccount(sa))
  - nok
    - send(msg("Amount too large"))
  - ok
    - sendMoney(sa);
    - send(Receipt(sa));

VerifyTransaction

Simple GetAmount

sm GetAmount

- Send(msg("select amount"))
- cancel
- SelectAmount
  - cancelled
  - again
  - amount(sa);
  - Send(msg("select another amount"))

- definition of entry point

- use of entry point
Extended GetAmount

Another similar service: Currency
Interactions are generalizable and redefinable

ATM revisited - generalised
Extended state machines

Extended state machines

sm WithdrawalATM

-Service (extended)

Withdrawal

Withdrawal

ok
cancelled

CardOut

sm CurrencyATM

-Service (extended)

Currency

Currency

ok
cancelled

CardOut

Decomposing a Lifeline wrt an Interaction

we want to look into this lifeline

this is the name of the diagram where we find the decomposition
Decomposition

Composite (design) class
Structured Classes are like other Classes

- Structured Classes may have
  - attributes & operations, interfaces, ...
- Internal structure is inherited, inherited parts may be redefined by extension

What about Components?

- Components have all the properties of structured classes

Note that these are just derived, that is they are also defined for classes
What is special for Components

- Realization by a number of classes

... and

- may be kind of ‘package’, i.e. it may have model elements that you would not have for classes

A component may have e.g. use cases, sequence diagrams, packages, dependencies, components
Deployment of components

- Artifacts,
- Nodes,
- Network of Nodes,
- ...

Must be profiled for actual component models

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Base Class</th>
<th>Parent</th>
<th>Tags</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJGEntityBean</td>
<td>Component</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>EJGJSessionBean</td>
<td>Component</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>EJGMessageDrivenBean</td>
<td>Component</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>EJGHome</td>
<td>Interface</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>EJGRemote</td>
<td>Interface</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Finally

- **Tools**
  - IBM Rational Software Modeler (eller Architect)
  - Telelogic real-time, telecom, but moving towards general
  - I-Logix real-time, telecom, control systems
  - Softteam general, with emphasis on profiling
  - MagicDraw
  - Enterprise Architect

- **Books**
  - Selic et al. (eds) *UML for Real* (Chapter 3)
  - Chonoles and Schardt: *UML2.0 for Dummies*
  - Fowler *UML Distilled (Third Edition)*
  - Craig Larman: *Applying UML and Patterns*