UML 2.0 crash course

Birger Møller-Pedersen,
Øystein Haugen

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
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

Very Short History

- December 1999 RFI (Request For Information)
  - 29 Responses to the RFI
  - Ericsson made a joint response with Motorola and Alcatel (and Nokia)
- September 2000 RFP (Request For Proposal)
- October 2001 Initial submission
- August 2002 Revised submission
- June 2003 Recommended for adoption
- Now Finalization Task Force
- April 2004 Final spec
Snapshot from one of the meetings
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
Domain Model I

User \(\rightarrow\) ATM \(\rightarrow\) Bank

Card \(\rightarrow\) Account

myAccounts

Domain Model II

ATM

CardReader \(\rightarrow\) Keyboard \(\rightarrow\) Screen \(\rightarrow\) CashDispenser
Use Case Model

Context model with UML1.x class diagrams

- with plain composition and no encapsulation
- with only provided interfaces on classes
Composite class (incomplete)

- with parts, ports and connectors

```
ATM
  CardReader
  Screen
  Keyboard
  CashDispenser

User-Reader
User-Screen
User-Keyboard
User-Cash
```

Context Model in UML2.0 - 1

- composite structure as part of a Collaboration

```
BankContext
  User
  ATM
  Bank
```

```
User-Reader
User-Screen
User-Keyboard
User-Cash
```
Context Model in UML2.0 - II

- Including multiplicities on parts

![Diagram of BankContext with multiplicities on parts]

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
### 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

send(msg("Give PIN"); n=1; PIN=0

[<4]digit/
 n++; PIN= PIN+digit*10^(3-n)

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

waitOK

definition of exit point
```
Statemachine for the ATM

Attributes of the ATM

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

<table>
<thead>
<tr>
<th>&lt;&lt;statemachine&gt;&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATM</td>
</tr>
<tr>
<td>authN:integer</td>
</tr>
<tr>
<td>cid: integer</td>
</tr>
<tr>
<td>sa: Amount</td>
</tr>
<tr>
<td>aa: Amount</td>
</tr>
<tr>
<td>sendMoney(a:Amount)</td>
</tr>
</tbody>
</table>

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

sm Withdrawal

:GetAmount

cancelled

again

send(CheckAccount(sa))

nok/
send(msg("Amount too large"))

VerifyTransaction

ok/
sendMoney(sa);
send(Receipt(sa));

ok

cancelled

use of entry point

Simple GetAmount

sm GetAmount

Send(msg("select amount"))

:SelectAmount

cancel

cancelled

Send(msg("select another amount"))

amount(sa);

again

definition of entry point
Extended GetAmount

Another similar service: Currency
Interactions are generalizable and redefinable

ATM revisited - generalised
Extended state machines

Extended state machines

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?

- Have all the properties of structured classes
- In addition:
  - A kind of ‘package’, i.e. it may have model elements that you would not have for classes
  - Realization by a number of classes
  - Deployment
    - Artifacts, nodes, …
- Still to be stereotyped for e.g. .NET and EJB components
Finally

- Tools
  - IBM (Rational) general
  - Telelogic real-time, telecom, but moving towards general
  - I-Logix real-time, telecom, control systems
  - Softteam general, with emphasis on profiling

- Books
  - Selic et al. (eds) *UML for Real* (Chapter 3)
  - Chonoles and Schardt: *UML2.0 for Dummies*
  - Fowler *UML Distilled* (*Third Edition*)
  - Coming soon: Rumbaugh: *UML Reference Manual*