UML
The View from the Front

James Rumbaugh
9 March 1999
Rational Software Corporation
Evolution of the UML

RTF report 4/99

OMG Revision 9/97, adoption 11/97

OMG submission 1/97

6/96 & 9/96

OOPSLA 10/95

Start 1/95

Booch Method  
OMT  
Other methods

Unified Method 0.8

UML 0.9 & 0.91  
UML Partners’ Expertise

UML 1.0

UML 1.1

OMG feedback

UML 1.3
UML Revision Goals

- Fix typographical errors and omissions
- Fix technical bugs and inconsistencies
- Clarify ambiguities and special cases
- Adjust discrepancies in naming and organization
Revision Schedule

- Formed committee of 12 persons
- Cris Kobryn was chairman
- Began work December 1997 after OMG adoption
- Completed technical changes December 1998
- Final UML version 1.3 report due April 1999
- Supplement for XMI and IDL compatibility 1 or 2 months later
Revision Process

- Users submit comments through OMG
- Organize comments in a database
- Perform internal consistency checks on UML documents
- Discuss changes in telephone and OMG meetings
- Assign workgroups to update documents
Workgroup Areas

- Static structure
- Model management
- Use cases and collaborations
- State machines
- Activity graphs
- OCL language
- IDL format
- UML architecture and standards issues
Nature of Changes

- Over 500 comments submitted
- A few changes important to most users
- A moderate number of changes important for special areas
- Many clarifications of wording and interpretation
- Many typos fixed
- Lots of small internal issues of interest only to toolmakers or UML groupies
- Many comments were declined as incorrect or out of scope
Static Structure Revisions

- Allowed associations from Classes to Interfaces
- Removed unnecessary Generalization stereotypes:
  - extends, inherits, private, subclass, subtype, uses -> implementation
- Allowed Classes to declare Signals
- Reorganized and renamed Dependencies
## Classes Declaring Signals

### PrintSpooler

- `changeSettings(settings)`
- «signal» `print(job:PrintJob)`
- «signal» `printerFree()`

### PrintSpooler

- `changeSettings(settings)`
- `Signals`
  - `print(job:PrintJob)`
  - `printerFree()`
Relationships

- Association
- Generalization
- Flow
  - become, copy
- Dependency
- Metarelationship
  - instance, powertype
- Use Case
  - extend, include

\[
\begin{align*}
\text{occupation} & \quad \ast \quad 1 \\
\langle \text{become} \rangle & \\
\langle \text{call} \rangle & \\
\langle \text{instanceOf} \rangle & \\
\langle \text{extend} \rangle & \\
\end{align*}
\]
Dependencies

- **Abstraction**
  - derivation, realization, refinement, trace

- **Binding**

- **Permission**
  - access, friend, import

- **Usage**
  - call, create, instantiate, send
Model Management Revisions

- Distinguished «access» Permission from «import»
  - access: permit access
  - import: load namespace

- Defined Subsystem with multiple viewpoints
  - specification
  - implementation
  - others possible
Permissions

P

+K

-L

Q

+M

-N

R

+F

-G

«access»

«import»
Use Case Revisions

- Replaced «uses» relationship with Generalization and «include» dependency
- Reclassified «extend» relationship as a Dependency
- Defined format of «extend» relationship and extension points
Use Case Relationships

Place Order
- extension points
- more requests

Request Catalog

Supply Customer Data

Order Product

Make Payment
- Make Cash Payment
- Make Credit Payment

«extend»
(more requests)

«include»
«include»
«include»
Sequence Diagram Revisions

- Allow states on lifelines to show change of state during execution
- Replaced timing marks by functions on message names
- Show either objects or roles in an interaction
State on Lifeline

:Scheduler

release()

:Ticket

unreleased

available

locked

buy()

sold

lock()
Timing Functions

{b.receiveTime - a.receiveTime < 1 sec.}

{c.receiveTime - b.receiveTime < 10 sec.}

{d.receiveTime - d.sendTime < 5 sec.}

a: lift receiver

b: dial tone

c: dial digit

... e: ringing tone

d: route
Roles vs. Objects: Roles

Party

host:Person

invite

accept

attend

guest:Person
Objects

Emma’s Party

Emma/host:Person

invite

accept

attend

Sean’s Party

Sean/guest,host:Person

invite

accept

attend

Marcia/guest:Person
Collaboration Diagram Revisions

- Automatically generate base association if not explicit
- Allow sequence numbers on «become» and «copy»
- Apply actions concurrently to set of target objects
Collaborations

![Collaborations Diagram]

- `file:TextFile` with 1 `assignedPrinter` to `printer:Printer`
Become and Copy

:Controller

1: expand()

:Directory[closed]

1.1: «become»

2: sort()

:Directory[open]
Multiple Targets of Action

targets . op (arguments)
State Modeling Revisions

- Added currentEvent to handled chained transitions
- Dropped special syntax (^) for sending signals
- Added continuous activity to states
- Allowed call events as alternative implementation of operation calls
Chained Transitions

A \[ E(x) / f(x) \] B

\[ \text{entry / setupB (currentEvent)} \]

\[ / g(x) \]

C

\[ \text{entry / setupC (currentEvent)} \]

\[ E(x) / f(x); \text{setupB (E(x))}; g(x); \text{setupC (E(x))} \]
Continuous Activity

- Monitoring: detect intrusion / call police
- Sounding: do/ sound alarm

Reset
Call Events

calling procedure text

```
deposit (10);
...
amount := withdraw()
```

Get all or nothing depending on whether the account is locked.

Account

Locked

```
deposit (n) / balance := balance + n
withdraw / return 0
```

Unlocked

```
deposit (n) / balance := balance + n
withdraw / return balance; balance := 0
```
Activity Graph Revisions

- Allowed events to be deferred in states
- Provided synch states for synchronizing concurrent activities
- Added special icons for sending and receiving signals, conditional threads
- Added dynamic control of spawning parallel subactivities
Deferred Events and Special Icons

set up print job

signal send

turnOn

ordinary state

receive file printerReady / defer

signal receipt

printerReady

PrintEngine

print page
Synch States

Request purchase

Ticketing
- Pick seats
- Print tickets

Charging
- Validate account
- Post charges

Mail tickets
Dynamic Parallelism

1. Collect query information
2. Post to search engine
3. Sort results by priority
4. Send to each web searcher

Dynamic concurrency
Existing RFPs for Extensions

- Stream-based model interchange using XML
- Semantics of executable models ("action language")
- Profile for business enterprise modeling
- Real-time systems (architecture, performance, reliability)
Other Interim Issues

- Standard way to define profiles
- Diagram interchange formats
Possible Topics of UML 2.0

- First-class extensibility mechanism
- Precise specification of refinement semantics
- Versioning of models
- More permissive concurrency in activity and state graphs
- Associations at several levels
UML Assessment

- Clearly the dominant modeling language
- Not as clean as if one person had made it, but much better accepted
- Some mechanisms should be more general in the future, but a conservative approach was probably best
- Standard elements are uneven, probably need cleanup
- Pressures to apply UML widely, possibly beyond its intent
- Danger of incoherence if too many independent OMG RFPs push UML in different directions
- Vendors and users will provide pragmatic pushback to untried theory