OMEGA2
Profile & tools for system modelling and verification with UML 2.x & SysML

Iulian OBER, Iulia DRAGOMIR
IRIT / University of Toulouse

Tools developed in partnership with

Work supported by
Embedded systems development
computation under physical constraints

Physical world

Control engineering
Physical constraints
Requirements

Hardware engineering
Platform constraints

Requirements
Model driven engineering
mastering the complexity

Exploratory models
- Abstract
- Understandable
- Acurate
- Predictive
- Inexpensive

Productive models
- Abstract
- Understandable
- Acurate
- Predictive
- Inexpensive

iterative development process

the model = the system
Model driven engineering

- System model $\Omega$
- requirements:
  - functional
  - timing, schedulability, QoS

- logical (or functional) architecture
  - logical (de)composition
    - assembly
    - communication
  - structure
    - components
    - classes
    - ...
  - component behavior
    - state machines
    - operations
    - timing

- physical (non-functional) architecture
  - tasks
  - resources
  - scheduling policy
  - transaction protection
  - ...

Semantic models \(\rightarrow\) Simulation \(\rightarrow\) Validation \(\rightarrow\) Implementation
Outline

- Overview of OMEGA v1 - profile and tools
- OMEGA v2 language extensions
  - composite structures
  - concurrency model
- Implementation in IFx2
- Conclusions
OMEGA language

A large subset of UML (1)
+ (More) model coherence constraints
+ A formal operational semantics (2)
+ RT & Verification extensions (3)

(1) Structure (classes, relations, composite structures) and behaviour (SM, actions)
(2) Based on the Rhapsody tool semantics and defined in [Damm, Josko, Pnueli, Votintseva 2002 & Hooman, Zwaag 2003]
(3) Timing constraints, timed behaviour (semantic projection to timed automata), property observers
OMEGA v1 language

UML class diagrams
• active / passive classes
• associations
• composition
• generalization
OMEGA v1 language

Behaviour
- state machines
- "primitive" operations
- imperative action language
  - assignments
  - control structure
  - communication
  - object creation
- communication
  - asynchronous signals
  - asynchronous calls
  - synchronous blocking calls
OMEGA v1 language

Composition & communication semantics

- Passive objects
- Active object
- Activity groups (run-to-completion)
- Synchronous calls
- Asynchronous calls & messages
Observers: objects monitoring the system state & events and giving verdicts

```omg
<<Observer>>
  liftoffPerformed_right2

g : Ground
cmc : MissionConstants
tc : TimeConstants

wait_start

match send ::EAD::Signals::Start(void) by g /
begin mc := g.Acylic.MissionConstants; tc := g.Acylic.TimeConstants end

wait_ignition_p1

[ g.Acylic.EAP.Pyro1 @ ignition_done ]

p1_ignited

[ now >= (tc.MN_5*2 + mc.Tpstar_prep) ]

[ now >= (tc.MN_5*2 + mc.Tpstar_prep) ]

[ g.Acylic.EAP.Pyro2 @ ignition_done ]

ok

choice

[ now < (tc.MN_5*2 + mc.Tpstar_prep) ]

<<error>>
  ko
```
IFx toolset

Goals: early model validation & debugging
Principle: translation to a formal timed automata model

Functionality

- simulation
  - interactive, random, replay/
  - analyze diagnostics...
- verification
  - observers, $\mu$-calculus,
  - state graph minimisation
    (bisimulation),…
- static analysis
  - dead variable/code elimination,
  - slicing,…

Architecture

Rhapsody, Rational Rose, Eclipse UML

XMI 1.0/2.0

OMEGA UML model

UML-IF frontend

UML2IF translator + compliance checker

UML validation driver

UML tools

IF tools

IF model

IF behavioral tools

- simulator
- verifier
- scheduling analysis
- state explorer

Graph level tools (CADP)

minimization, comparison, composition…
Tool workflow
Uses of OMEGA

Case studies:

- EADS Astrium Space Transportation: Verification of functional & scheduling properties of the Ariane-5 flight software (UML) [FMOODS06]

- Nationaal Lucht- en Ruimtevaartlaboratorium (NLR): Timing verification of airborne data acquisition module (UML) [UML&FM08]

- ESA / EADS Astrium: Simulation and verification of ATV Solar Wing Management (SysML)

Tool development partially financed by ESA
Outline

- Overview of OMEGA v1 - profile and tools
- OMEGA v2 language extensions
  - composite structures
  - concurrency model
- Implementation in IFx2
- Conclusions
Motivation - missing features

- Language
  - Structure: *hierarchical architecture modelling*
    - UML 2.x composite structures
    - SysML internal block diagrams
  - Concurrency model: *better synchronisation constructs*
  - Behaviour: parallel regions, other minor updates

- Tool
  - Compatibility with recent UML/SysML editors (Rhapsody 7.x) -- support for *XMI 2.x*
UML / SysML composite structures

Elements:
(a) port
(b) part
(c) delegation connector (port-to-instance)
(d) assembly connector (port-to-instance)
(e) assembly connector (instance-to-instance)
(f) delegation connector (port-to-port)
(g) provided interface
(h) required interface
Ambiguous structures
Ambiguous structures
Ambiguous structures

should imply
OMEGA objective: clear & coherent semantics

Rules for well-formed structures

Static type safety

Operational semantics
Bidirectional vs. unidirectional ports

Bidirectional ports lead to typing problems:

- example of action in A:
  \[ \text{port}_0.\text{op2()} \quad // \text{port}_0 \text{ complies to J} \]

- behaviour specification of port_0:
  \[
  \begin{align*}
    \text{input op2() :} & \quad // \text{port}_0 \text{ complies to J and I} \\
    \ldots & \\
    \text{input op1(x) :} & \quad // \text{port}_0 \text{ complies to J and I} \\
    \ldots & \\
    \text{input sig1(x) :} & \quad // \text{port}_0 \text{ complies to J and I} \\
    \ldots & \\
  \end{align*}
  \]

OMEGA: no bidirectional ports
Replace with:
Connector directionality

respect port directionality

respect association directionality
Connector typing

- \( \{ I, J, L \} \cap \{ I \} = \{ I \} \)
- \( \{ K \} = \text{typeOf(itsK)} \)
- \( \{ I, J, L \} \cap \{ J, L \} = \{ J, L \} \)
- \( \{ K \} \cap \{ K \} = \{ K \} \)
- \( \{ K \} = \text{typeOf(deleg_backup)} \)
Port behaviour

Default port behaviour:

Explicit port behaviour:

sK/
begin
deleg_K ! sK;
deleg_backup ! sK
end
Concurrency model

- Lack of sharing & synchronization mechanisms
  ⇒ Ada-like **protected objects** (with *functions* and *guarded entries*)
  ⇒ Coherent with the activity group semantics
  ⇒ Rules to make them *coherent with composite structures*

```
A

1 b:B
1 c:C
1 protected d:D
```
Outline

- Overview of OMEGA v1 - profile and tools
- OMEGA v2 language extensions
  - composite structures
  - concurrency model
- Implementation in IFx2 and evaluation
- Conclusions
IFx2

- Same overall architecture
  - translation of models (XMI 2.x) to IF using Eclipse/UML

- Principles and evaluation
  - ports and connectors handled as *first class* elements
    ⇒ *dynamic routing* for requests
    ⇒ allows for *dynamically reconfiguring* composite structures
  - offline partial-order reduction to reduce impact of routing actions on the size of the state space
    ⇒ state space explosion is not aggravated by new features
Conclusions and future work

- **Simple but not simplistic** profile for real-time software & systems modelling
  - fully defined operational semantics
  - simulation & verification toolset
  ⇒ complementary to broader approaches such as MARTE

- Tool & profile currently evaluated by ESA and Astrium Space Transportation on realistic models

- **Current and future work**
  - formalize composite structures type system & prove type safety
  - improve profile & tool: SysML adaptations, improved integration and user experience, advanced diagnostics features, etc.