Real time system modeling with UML: current status and some prospects

François Terrier, Sébastien Gérard

LETI (CEA - Technologies Avancées) DEIN
CEA/Saclay
F-91191 Gif sur Yvette Cedex France
Phone: +33 1 69 08 62 59 ; Fax: +33 1 69 08 83 95

Francois.Terrier@cea.fr ; Sebastien.Gerard@cea.fr
Embedded systems soon > 50% of market
✓ With more and more importance of software
☞ How to master…

… the software wave!
Use of a « universal » modeling standard

- We must go from artisanal practices to industrial production solutions
  - high level modeling and component based development
  - Idea integration of complementary/concurrent modeling notations proposed for OO methods

OMT
Booch
OOSE
Fusion
Classe-Relation
ROOM
HOOD
etc...
Fin 1990
Use of a « universal » modeling standard

- We must go from artisanal practices to industrial production solutions
  - high level modeling and component based development
  - Idea integration of complementary/concurrent modeling notations proposed for OO methods
But what about real-time systems?

- Importance of dynamic in such system requires the specialization of the modeling language.

- Solutions have been developed to integrate current practices into the UML OO framework:
  - More or less advanced levels of integration of real time and object paradigms.

- Variability of the practices of real time domain depending on the context: small embedded system or installation control and command, production automaton, distributed systems, safety critical systems, telecom, high performance computing...
  - Low level of automatic integration of the « good practices »
Plan of the presentation

- Current status of UML 1.3
- ARTiSAN proposal
- RT-UML / Rhapsody tool
- UML/SDL tools association
- UML-RT / ROSE-RT tool
- Toward a stronger paradigm integration
- Some open points
UML 1.3: essential models

- **Use case diagram**
  - Requirement modeling
- **Class diagrams**
  - Static structure
- **Activity, sequence or collaboration diagrams**
  - Interaction
- **State machine diagrams**
  - Behavior view
- **Component, deployment diagram**
  - Structure of material implantation
Use case diagram

Speed Regulator

- regulate speed
- start regulating
- stop regulating

Actor:
- Regulator On/Off
- Motor
- SpeedSensor

System:
- Regulator
- Motor
- SpeedSensor

Relation:
- « include »
- environment
- system border

Use case:
- use case

System:
- system

Actor:
- actor
The UML concept of active object

- They are objects having an (*only ?*) independent processing resource (*thread*, *process* or other)

- Behavior not well defined: *connection between messages processing and use of the processing resource are not defined*

- *are they just an Object Oriented view of « tasks » ?*
Sequence diagram

::RegulatorScreen

::Regulator

::RegulatingLaw

Synchronous message

Asynchronous message

maintainSpeed() calculate() dtorque

update(info)

Life line

time

response

instance

Sequence diagram
UML communication mechanisms

Communication: only by message passing

- A message = an action + an event
  - Point to point communication, possibility to have a set of targets

Two types of messages

- Operation call (CallAction + CallEvent)
  - Synchronous/asynchronous, input and output parameters
- Signal sending (SendAction + SignalEvent)
  - Asynchronous communication, input parameters only

No specific communication mechanisms for the active objects…
Off

Initial state

State machine

Trigger event

Guard

Action expression

Final state

Transition

start [vitesse>30] / startRegulating()
UML state machines

- Each object can have a state machine (or even several)
  - Transitions can have four triggering types of events (or none):
    - Object operation call: “CallEvent”
    - Signal receipt: “SignalEvent”
    - Condition becoming true: “ChangeEvent”
    - Date occurrence: “TimeEvent”

- A state machine has a queue to store incoming events
  - Storing mechanisms and its extraction protocol has to be defined (implemented) by developers

- Current event processing has to be completed before handling of next incoming event
  - Run To Completion (RTC) assumption
  - No distinction between internal and external events
UML 1.3 timing specifications

- Time can be expressed through two notations:
  - Timer setting on transitions of state machines (TimeEvent)
  - Specification is performed through implementation mechanisms

- Dates definition in sequence diagrams

- No relation is defined between these specifications neither with the rest of the model nor with the system scheduling policy
UML meta-model: four levels of instanciation

Meta Meta Model (M3)
- MOF

Meta Model (M2)
- SPE
- UML
- Real Time?
- SDL?
- UML-RT

Standard profiles (M2)
Specific user profiles (M2)

Application model

Model (M1)

Application implantation

Objects (M0)

instanceOf

Entity

Class

Car

myCar

François Terrier, Sébastien Gérard  27-06-2000  SAM'2000
**UML 1.3: specialization mechanisms**

- **Tagged value**
  - Properties added to a meta-class
    - E.g., \{documentation = “…”\} on Element

- **Constraint**
  - Addition of « Well-Formedness Rules »
    - E.g., \{destroyed\}, \{new\} or \{transient\} on Instance

- **Stereotypes**
  - Indirect add of elements to the meta-model
    - E.g., « thread » or « process » on Classifier
UML 1.3: specialization mechanisms

- Tagged value
- Constraint
- Stereotypes

Set of tagged values and of constraints specializing an element of the meta-model

Organisation needs

≡ notion of Profile in UML 1.3
Objective

Specialization of a standard meta-model (e.g., UML) into a specific meta-model dedicated to a given application domain.
A profile can contain:

- Fundamental meta-classes on which is based the profile
- Selected elements of the reference meta-model
A profile can contain:

- Selected elements of the reference meta-model
- Stereotypes, tagged values, constraints added to the profile
- Extension mechanisms
A profile can contain:

- Selected elements of the reference meta-model
- Extension mechanisms
- Descriptions of the profile semantics

Clarification of « Semantics Variation Points » or UML ambiguities
**UML profile definition**

**A profile can contain:**

- Selected elements of the reference meta-model
- Extension mechanisms
- Descriptions of the profile semantics
- Additional notations

**Example:**

Mr Dupont « driver »
A profile can contain:

- Selected elements of the reference meta-model
- Extension mechanisms
- Descriptions of the profile semantics
- Additional notations
- Rules for model translation, validation, presentation

e.g.:
UML 1.3: lacks, ongoing works at the OMG

Three points concerning real-time:

- Proposal of an « Action language semantics »
  - Must integrate proposals from all the submitters (« SDL » domain consortium, Rational and other tool vendors)
  - Standard only on semantics not on notations…
  - Several times postponed…

- Proposal of a real-time profile on
  - Time semantics, scheduling and real-time concepts at implementation level → a UML virtual machine…

- Works are also in progress on Profile formalization, and on pattern description and use…
ARTiSAN: two orthogonal models, weak integration

This tool aims to offer in the same environment:
- A classical UML modeling facilities
- A classical task model called the concurrent model
- An implementing stage (assignment of the objects to tasks)

Task model has to be manually implemented

Relation between task and object model is weak

Timing constraints must be translated on the implementation
  - Communication between object of different tasks must be implemented by the user with low level concepts
RT-UML: two « orthogonal » but « integrated » models

- Tasks are identified by declaration of active objects
  - Behavior specified by state machine of reactive objects

![Diagram of RT-UML models and behavior](image-url)

- Rhapsody tool
**RT-UML**: two « orthogonal » but « integrated » models

- Tasks are identified by declaration of active objects
  - Behavior specified by state machine of reactive objects
  - Reactive objects are assigned to active objects...
  - Communication by message between reactive objects

![Diagram of RT-UML](attachment:diagram.png)

- **ActuatorControl**
- **RegControl**
- **SensorControl**

- **ActControlTask**
- **RegControlTask**

- **Rhapsody tool**
RT-UML: non homogenous features

Different processing semantic between signals and operation calls

- Signals: parallel execution in thread of receiver object
- Operation calls: execution in thread of sender object
  - Under control of the object state machine « triggered op. call »
  - Without control of the object state machine...
- Concurrency is only managed through RTC assumption
- Return value of operation call can be defined... under responsibility of the caller

Priorities can be assigned to the active objects
  - Not on the event themselves...

Timing constraints must be implemented
UML/SDL: association of UML analysis and SDL design

- UML modeling is used at the first modeling stage
  - Use cases, sequence and class diagrams are defined
  - Active object declarations → SDL processes
  - Passive objects → Abstract Data Types of SDL

```
ActuatorControl
RegControl
SensorControl

pRC
\[\text{RegControl}\]

pAC
\[\text{ActuatorControl}\]

pAC
\[\text{RegulatingLaw}\]

C1
\[\text{cmd}\]

C2
\[\text{speed}\]

Newtype RegulatingLaw
Operators
calculate : Speed, Speed → TorqueVariation
endnewtype::;
```
**UML/SDL**: association of UML analysis and SDL design

- **UML modeling is used at the first modeling stage**
  - Use cases, sequence and class diagrams are defined
  - Active object declarations → SDL processes
  - Passive objects → Abstract Data Types of SDL

---

```
ActuatorControl
RegControl

SensorControl

speed

pAC 1

ActuatorControl
RegulatingLaw

RegControl
pRC 1

SensorControl

SpeedRegulator_Behavior

stopRegulating()
/updateScreen(OFF);

On

startRegulating()
/targetSpeed = returnValue;

Off

Stopped

Running

Stopped

Running

stopped

startRegulating()
/targetSpeed = returnValue;

[cmd] C1

ActuatorControl

[Speed] C2

SensorControl

Newtype RegulatingLaw

Operators

calculate : Speed, Speed -> TorqueVariation
endnewtype::;
```
UML/SDL: mapping is under user responsibility

- The developer has to clarify the mapping of:
  - UML state machines → SDL specifications
  - Use of shared passive objects in UML model → SDL
  - Active object communication → SDL signal exchange

- Independency between model and implementation? implementation?

- Timing specifications:
  → only with low level implementation mechanisms
    • Timers, SDL priorities…
**UML-RT**: attempt to integrate task and object paradigms

- « capsule » stereotype identifies active objects
  - They will be assigned to task at implementation stage
  - They have state automaton

- Communication is performed through « ports »
  - Sending of signal via port of communication

✓ Defines the set and protocol of the exchanged messages
Communication based on specific concepts
- Output parameters: return values managed by sender

Priorities can be assigned on messages
- One message queue by priority (five levels):
  At implementation level mapping with task priorities must be managed by hand

Timing specification through low level mechanisms
Synthesis on current offers

- Weak integration of object and real time...
  - Two very different models (e.g., ARTiSAN, UML/SDL)
  - Behavior lies on operation and signal processing but with poor links to the usual object interface
  - Focus is made more on signal than on operations that leads to behavior specification mixing up control action at object level and processing actions at operation level
  - Output parameters often hard to manage

- Poor facilities to express timing constraints
  - Specification of timers or of priorities
  - Implementation of real time constraints kept to the users
  - Sometimes difficulties to map model constraints on RT-OS → model / task priorities with OS priority management policies
Larger market → New users → New needs

Customers want to specify good RT models
- They want also to be able to prototype/develop the systems

- Notations are not sufficient: method of use is required
  - Continuity and “tracability” of the model is mandatory
  - Availability of Model and application validation is critical
**ACCORD/UML: introduction of real-time objects**

- High level specification of real time behavior
  - Declarative expression of the real-time constraints
  - Hide RT-OS implementation concepts

- « Formal » models
  - Validation on model
  - Automatic prototyping code generation
  - Tuning of implementation through directives, patterns...

- Keep the usual object programming practices
**ACCORD/UML: « task » model**

- Task are attached to implementation of RT objects
  - support parallel processing of the messages they receive

- User point of view:
  - an object encapsulating data & processing

**Diagram:**
- ARealTimeObject
  - Extern interface
  - operation_1
  - operation_2
  - ... operations code
- Attributes
**ACCORD/UML: « task » model**

- Task are attached to implementation of RT objects
  → support parallel processing of the messages they receive

- User point of view:
  an object with its own processing resources

```plaintext
aRealTimeObject

Extern interface
operation_1
operation_2
...

operations code

Attributes

Memory space
```
ACCORD/UML: « task » model

- Task are attached to implementation of RT objects
- support parallel processing of the messages they receive

- User point of view:
  an object performing itself the control of its processing
**ACCORD/UML: « task » model**

- Task are attached to implementation of RT objects
  - support parallel processing of the messages they receive

- User point of view: an autonomous computing entity with a standard UML object interface
By messages that convey real-time constraints

- Deadlines, priorities…

- Operation calls: Asynchronous message passing
  - Output parameters create synchronization on their use

- Signals: Asynchronous anonymous broadcast (atomic)
  - Only input parameters → signal attributes

Shared passive object can be defined

- Concurrency control is added
Instead of having one single state machine…
Object behavior is structured at two levels

- Global object control: protocol state machine
  - Transition only triggers processing of an object operation
  - Clear link with object interface is maintained

Detailed specification of operations

- Formal description of main actions performed by an operation
Assignment of task to message processing by RTO

- Default: reaction on signal and internal processing (always defined by the execution of an object operation)
- User: implementation directives on message processing
- Constraints specification only (deadline, period, priority)
  - Associated, when possible, to operation properties like execution times

Scheduling mechanisms are provided

- Event queue management based on message RT constraints
- Concurrency management based on operation constraint declaration declaration
ACCORD/UML: synthesis

- Keeps to modeling concepts the common OO view
  - It can be used by non real-time specialists
- Provides models without implementation details
  - They can be changed without changing the structure of the models (nor in class, or sequence or state diagrams)
- Facilitate automatic generation of prototype code
- Allows model validation
  - through simulation or formal analysis (execution model can be deduced from the specification)
Some open points

- Definition of development methods…

- Improvement of formal validation of models…
  - Lot of work to do for integration of existing solutions

- Integration of new paradigms:
  - Links between discrete and continuous models…
  - Links with « data flow » based modeling (high performance embedded parallel computing)
  - Links with hardware: « co-design »

- What about the next versions of UML standard?
Some web sites

- **AIT-WOODDES**: “Workshop for Object Oriented Design and Development of Embedded Systems”, IST project of the 5th PCRD
  http://wooddes.intranet.gr/project.htm

  http://www-dta.cea.fr/leti/UK/Pages/Tech_info/Sivooes.htm

  Workshop on Formal Design Techniques for Real-Time UML
  http://wooddes.intranet.gr/workshop.htm

  http://people.ce.mediaone.net/weigert/actionsemantics/home.html

- **ARTiSAN**: http://www.artisansw.com

- **RT-UML (Rhapsody)**: http://wwwilogix.com


- **UML-RT (ROSE-RT)**: http://www.rational.com