From Analysis to Code Generation of Protocols and Embedded Software with a UML-Based Formal Environment Named TURTLE’2005

Ludovic Apvrille
ludovic.apvrille@enst.fr

Outline

- Problematic
- The TURTLE UML profile
  - Methodology
  - Definition
  - Semantics
- TTool: The TURTLE Toolkit
- References
UML at a Glance

- Noting new but a federation of best practices
- A notation, not a methodology
- An international standard at OMG (Object Management Group)
  - 13 diagrams to express complementary point of views
  - Semantic variation points
  - No standardized formal semantics
  - Profile: the possibility to tailor the UML for your application domain
- Industry support
  - Tools (TAU G2, …)
  - Lingua franca for software-intensive system designers and developers
- Research work
  - Real-time?
  - Adding formality to the UML
  - A UML model for simulation, verification, code generation, test generation, performance evaluation, …

UML Profiles for Embedded Systems and Protocols

- Profile for Performance, Scheduling and Time
  - Profile defined at the OMG
  - Addresses more specifically real-time systems
- Rose RT Profile
  - Toolkit
    - Capsules
    - Ports
    - Protocols
    - Communication channels
  - Methodology
    - RUP
- TAU G2
  - Toolkit based on UML 2.0 elements issued from SDL
  - Methodology
Propositions

Idea: let us enrich UML
- UML operators are informal
- UML lacks advanced temporal operators such as time intervals
- UML has no methodology (no validation)

Proposition: Semi-formal UML-based environment
- Semantics given by mapping to a Formal Description Technique

What formal language?
- Well-defined formal semantics
- Logical and temporal operators
- Tools
  - $\Rightarrow$ RT-LOTOS / Petri Nets with real-time extensions
  - $\Rightarrow$ TURTLE UML profile (Timed UML and RT-LOTOS Environment)

TURTLE: Methodology

(1) Analysis
IOD + SDs

(2) Design
CD + ADs

(3) Deployment:
components + DD

(4) Code (Java)

Execution

Formal validation
Code generation

Automatic synthesis
Formal validation

Generation and execution of Java Code
TURTLE Analysis

Interaction Overview Diagram

Sequence Diagram

TURTLE Design
Relations between Tclasses: TURTLE’s Composition Operators

- Default relation
  - Parallel

- Communication relations
  - Synchro
  - Invocation

  *Note: Tclasses exchange information exclusively through communication gates*

- Others
  - Sequence
  - Preemption

- There can be only one composition relation between two tclasses

Activity diagrams: Logical and Temporal Operators
TURTLE Deployment diagrams

TURTLE artifacts
- Set a classes modeled in a TURTLE designs

TURTLE Deployment diagrams
- Execution nodes
  - May hosts TURTLE artifacts
- Links between nodes
  - Interconnection of Artifacts' gates
  - Formal specification
    - Parameter: delay, loss rate
    - Pseudo FIFO
      - Actions in the same time slot may be reordered
  - For Java code generation
    - Protocol: UDP, TCP, RMI
    - Ports

Example of TURTLE Deployment Diagram

Artifact PkgClient is defined here, and used there
LOTOS

**LOTOS = Language Of Temporal Ordering Specification**
- Standardized at ISO
- Process algebra

Choice, Parallel, Synchro, Sequence, Disrupt
- \( P[a,b, c, d] = P1[a, b] / P2[c, d] \)
- \( P[a,b, c, d] = P1[a, b] || P2[c, d] \)
- \( P[a, b, c] = P1[a, b] /\{b\} P2[b, c] \)
- \( P[a, b] = P1[a] >> P2[b] \)
- \( P[a, b] = P1[a] \uparrow P2[b] \)
RT-LOTOS

LOTOS with real-time extensions!

- Delay
- Latency
- Time-limited offer

- $\text{delay}(10)$ $a$;
- $\text{latency}(15)$ $b$;
- $c\{12\}$;

Example of an RT-LOTOS Specification

```plaintext
specification MEDIUM : noexit :=
(…)
behaviour
  hide iu_s, iu_d in
  let period : nat = 30000 in
  stream_sender[iu_s](0, period)
  |[iu_s]|medium[iu_s,iu_d](14000, 20000)
where
  process stream_sender [iu_s] (n : nat, period : nat) : noexit :=
    iu_s!n; delay(period) stream_sender[iu_s] (n+1, period)
endproc

  process medium [m_in,m_out] (dmin, dmax : nat) : noexit :=
    m_in?x:nat; delay(dmin,dmax)m_out!x; medium[m_in,m_out](dmin,dmax)
endproc
endspec
```
Semantics of TURTLE Operators

Translation algorithms

\[
\begin{align*}
\tau(AD_1) \parallel \tau(AD_2) \parallel \ldots \parallel \tau(AD_n) & \gg \\
(\tau(AD_1') \parallel \tau(AD_2') \parallel \ldots \parallel \tau(AD_m')) \\
\end{align*}
\]

RTL Toolkit (Developed at LAAS-CNRS)

Simulation
Generation of DTAs
Generation of reachability graphs
Visualization of Graphs (Cont.)

Logical state

Logical transition

Temporal transition

Substate

Regular termination or deadlock

Logical transition with data exchange

Minimized Graph (Obtained with CADP – INRIA)

Projection and Minimization

Ludovic Apvrille - UML - 2005. Slide #19

Ludovic Apvrille - UML - 2005. Slide #20
### From Graphs to TURTLE Actions

- Action on the graph
- Corresponding TURTLE actions

### Other Useful Features

- **Generation of executable Java code**
  - *From design: monolithic application*
  - *From deployment: network-aware code*
    - UDP, TCP, RMI
- **Libraries of TURTLE elements**
- **Design**
  - *Observer*
    - Reduce combinatory explosion
  - *TObjects*
    - Instances of TURTLE classes
  - *TData*
    - Definition of complex types
Other Useful Features

- TTool is released with a CeCILL License
  - GPL-like
  - Sources may be quite easily extended to
    - Support other diagrams
    - Provide other semantics to diagrams
    - Implement other code generators

Use of TURTLE / TTool

- Modeling of embedded software
  - Alcatel Space (proofs of dynamic reconfiguration)
- Modeling of protocols
  - TéSA laboratory (DIPCAST Project)
  - UDCast (Maestro project)
- Modeling of security constraints
  - LAAS-CNRS (Safecast project)
- Others
  - Link between Telelogic TAU G2 / TTool
  - Generator of documentation
  - Use in lab sessions
  - ...

Page 12
Any questions?

- **TURTLE**’s website:
  - [http://labsoc.comelec.enst.fr/turtle](http://labsoc.comelec.enst.fr/turtle)

- **TTool**’s website:

---

References

- **Definition of the profile**

- **Use of the profile**