From Analysis to Code Generation of Distributed Systems with a UML-Based Formal Environment Named

TURTLE’2005

Ludovic Apvrille
ludovic.apvrille@enst.fr
Eurecom, Office 223

Outline

- Context
- TURTLE, A UML profile
- TURTLE’2005: analysis, design and deployment
- TTool, the TURTLE toolkit
- Demonstration
- From analysis to design
- Generation of Java code
- Future work
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)
  - 12 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, …

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
    => RT-LOTOS / Petri Nets with real-time extensions
    => TURTLE UML profile (Timed UML and RT-LOTOS Environment)
Chronology of TURTLE

1999
- First definition of operators

2000-2001
- Definition of a methodology supporting validation
- Modeling and translation rules
- Translation from TURTLE to RT-LOTOS partially implemented

2002
- New operators (temporal operators, new diagrams)
- Methodological extensions

2003
- First release of the TURTLE toolkit (TTool)

2005
- TURTLE analysis
- TURTLE deployment
- Generation of executable code (Java)

Labs and People Involved in TURTLE

- LAAS / CNRS
  - Jean-Pierre Courtiat
  - Pierre de Saqui-Sannes

- ENSICA
  - Ferhat Khendek

- Concordia University
  - Ludovic Apvrille

- ENST
  - Ludovic Apvrille

- ENST Bretagne
  - Christophe Lohr

- Alcatel Space Industries
References

- Definition of the profile


- Use of the profile


TURTLE: Methodology

1. Analysis
   - IOD + SDs
   - Formal validation

2. Design
   - CD + ADs
   - Automatic synthesis
   - Formal validation

3. Deployment
   - components + DD
   - Formal validation
   - Generation and execution of Java Code

4. Code (Java)
   - Automatic synthesis
   - Code generation
   - Execution
TURTLE Deployment

TTool: an Open-Source Toolkit Developed at ENST - LabSoC
Demonstration

- Basic authentication mechanism of HTTP
  - Analysis, design

- Proof of security flaw
  - We demonstrate that a network attacker may obtain protected data
    - Hacking of network packets + replay of the same packet with slight modifications
  - Design

- Automatic Java code generation from design and deployment diagram
  - From design: monolithic application is generated
  - From deployment: a distributed application is generated

Automatic Generation of TURTLE Design from TURTLE Analysis

- Fundamentals
  - One instance in scenarios -> one TURTLE class
  - Behavior of instance -> behavior of the corresponding TURTLE class
    (activity diagram, and more generally, state machine)
  - Communication channels between instances -> for each trio (I1, msg, I2), a FIFO communication channel is settled between the corresponding TURTLE classes and gates
    - Two TURTLE classes / FIFO channel
    - On FIFO channel is modeled with two TURTLE classes
  - But:
    - Described scenarios are not always implementable
    - TTool does not detect scenario non-implementability
    - May be detected, sometimes, at generated design reachability graph level
      - Deadlock situation
Example of Non-Implementability

For example, trace !a!c!e is included into the generated design

Main Features of The Java Code Generator

- Code generated from TURTLE design
  - Monolithic application
  - Translation process relies on libraries
    - Temporal operators, synchronization, etc.
  - Code may be generated, compiled and executed from TTool

- Code generated from TURTLE deployment
  - Networked application
  - Asynchronous links are implemented using UDP, TCP or RMI
    - May be configured in TTool
  - Code may be generated and compiled from TTool
Generating Java Code from TURTLE Deployment

- Reuse of the generator implemented for TURTLE designs
- Issue of links between nodes
  - Links are modeled both for formal validation purpose and for code generation
  - Some parameters cannot be taken into account …
    - Delay, loss rate, etc.
  - …and some others have to be taken into account
    - Protocols, port number, network addresses, etc.

Generating Java Code from TURTLE Deployment (Cont.)

1. Generation of a “marked” TURTLE design
   - UDP / TCP / RMI + corresponding parameters
2. Reuse of the TURTLE design code generator
   - Extended generator
     - Network marked gates are computed differently
     - Several executable modules are generated
   - Libraries managing actions on gates have been extended
     - Data sending and receiving using UDP / TCP / RMI

C1_T1 = L12.g1
C2_T2 = L12.g2
C1_T1.g1 = L12.g1
C2_T2.g2

Main_C2.java

Ludovic Apvrille - October 12, 2005, Page 17
Conclusions

- TURTLE profile
  - Real-time UML profile
    - Analysis, design and deployment have been enhanced with logical and temporal operators
  - Graphic syntax compliant with UML 2.0
  - Formal semantics
  - Java code generation
- TTool
  - Diagram edition
    - Highly intuitive interface
  - Transparent use of powerful formal validation techniques
- Applications
  - Alcatel Space
  - UDCast
  - Used in several European research projects
    - DIPCAST, MAESTRO, etc.

Future work

- Refinement process at specification and design level
- TURTLE 2.0
- Formal semantics
  - Petri Nets
- Code generation
  - SystemC
- Toolkits
  - TINA
Related Work

- Real-Time UML profiles (selection)
  - ACCORD/UML
  - OMEGA
  - ARTEMIS

- OMG profile
  - STP (Scheduling, Time, Performance)
    - TURTLE libraries
  - MARTE (Modeling and Analysis of Real-Time and Embedded Systems)

- Formal verification tools
  - Synchronous languages
  - Hybrid automata (HYTECH)
  - Timed automata (UPPAAL)
  - Promela (SPIN)
  - Time Petri nets (TINA)

Any questions?

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

- TTool’s website: