Making Formal Verification Amenable to Real-Time UML Practitioners

Pierre de Saqui-Sannes
CNRS ; LAAS ; 7 avenue du colonel Roche,
Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS
F-31077 Toulouse, France
Email: pdss@isae.fr

Ludovic Apvrille
System-on-Chip laboratory (LabSoC),
Institut Telecom / Telecom ParisTech / CNRS LTCI
2229, routes des Crêtes, B.P. 193
F-06904 Sophia-antipolis Cedex, France
Email: ludovic.apvrille@telecom-paristech.fr

***Abstract*— TTool, a real-time UML toolkit, offers user-friendly interfaces to formal verification techniques such as reachability analysis, observer-based analysis and automatic generation of traceability matrices. Those techniques are surveyed in the paper.**

I. Introduction

The Unified Modeling Language or UML for short [1], is a wide-spectrum modeling language and a de facto standard supported by various tools from industry and academia. The increasing development of real-time systems has stimulated research work on "real-time UML profiles" [2] that customize UML with real-time capabilities. Those profiles also commonly provide a formal semantics, as well as tools and methodologies.

Real-time UML profiles reuse formerly developed formal verification tools. For instance, the TURTLE Toolkit (TTool [3]) developed for the TURTLE real-time UML profile [4], implements interfaces to RTL [5], CADP [6] and UPPAAL [7] formal verification tools. TTool particularly enables formal verification of TURTLE diagrams against temporal requirements. The purpose of this paper is to demonstrate that TTool hides the use of formal languages as much as possible to its users and thus makes formal verification amenable to illiterate practitioners, in particular in education context.

The paper is organized as follows. Section 2 introduces the verification-centric method associated with TURTLE. Section 3 overviews TTool. Section 4 discusses user-friendly access to formal verification. Section 5 concludes the paper.

II. TURTLE METHOD

We recommend using the TURTLE modeling language with a seven-step, verification-centric method (supported by TTool).

1) Requirement capture using an extended SysML requirement diagram [8] with formal description of temporal requirements.
2) Use-case driven analysis. Use-cases expressed in use-case diagrams are documented by scenarios expressed in the form of sequence diagrams, themselves being structured with interaction overview diagrams.
3) Formal verification of analysis diagrams against temporal requirements.
4) Formal synthesis of design diagrams from analysis ones. The system's architecture is depicted by a class/object diagram. Objects' behaviors are described with activity diagrams.
5) Object-oriented design. Class/objects and activity diagrams automatically generated in previous steps are enriched.
6) Formal verification of design diagrams against temporal requirements.
7) Rapid prototyping based on component and deployment diagrams.

Note that step 4 is optional. Also, step 7 is not addressed in this paper. At last, we recommend implementing incremental modeling, which assumes diagram enrichment loops from step 3 to step 2, and from step 6 to step 5, respectively.

III. TTool: The TURTLE Toolkit

TTool is an open-source toolkit that supports several UML2 / SysML profiles, including TURTLE [4] and DIPLODOCUS [9]. The main idea behind TTool, is that any model created using a UML 2 profile may be formally verified using RTL, CADP or UPPAAL (see Figure 1). In practice, UML diagrams are first automatically translated into an intermediate format expressed in formal language TIF, which serves as starting point for deriving (RT-)LOTOS or UPPAAL code amenable to verification tools.

IV. Formal verification

A. Reachability analysis

A reachability graph (RG) characterizes the set of stable states the system may reach from its initial state. The reachability analysis procedure implemented by RTL takes TURTLE temporal and non-deterministic operators into account, and generates a RG for bounded systems of reasonable size.

TTool syntactically checks a set of UML diagrams (e.g. for a TURTLE design: class/objects and activity diagrams), generates formal code (e.g. in RT-LOTOS) and invokes the related formal verification toolkit (e.g. RTL). From RGs, TTool computes statistics on states and transition and identifies deadlocks as well as shortest and longest paths in the graph. Also, TTool may invoke *dotty* to display RGs. Correspondence between actions on RGs and UML operators are also computed.

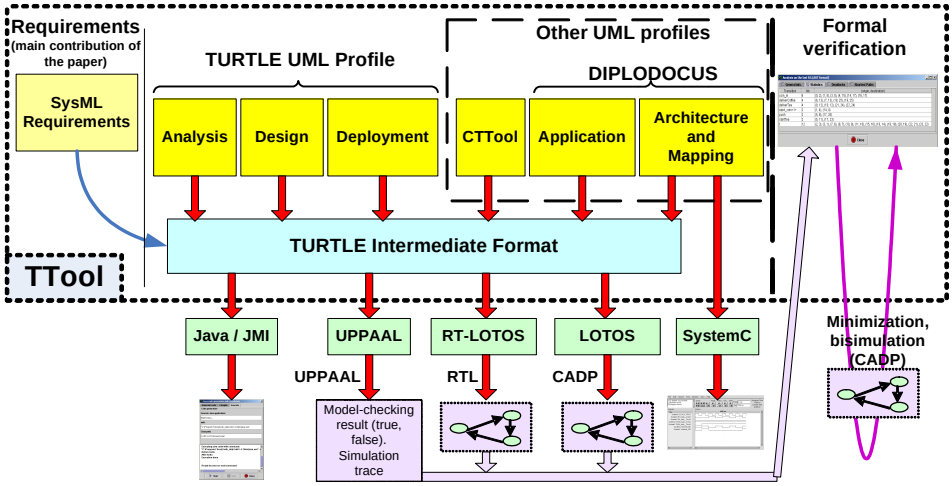

Fig. 1. TTool: profiles and verification techniques

Where other real-time UML tools focus on design diagrams, TTool also enables formal verification of analysis UML diagrams. Thus, someone unfamiliar with object-oriented design may nevertheless use TTool and apply formal verification to use-case driven and scenario based analysis.
At last, TTool offers a user-friendly interface to check for logic formulae (e.g. with UPPAAL). For example, to decide whether some UML action is reachable or not, or to study the liveness of that action, it suffices to right click on the corresponding action's symbol. Temporal logic formulae may also be entered directly in TTool.

B. Minimization of labeled reachability graphs

The RG of real-size systems may have millions of states and transitions. Such a graph is hard to visualize. Also, the statistics table is hard to interpret. Minimization is an alternative.
A RG may be transformed into a Labeled Transition System, a structure for which CADP implements minimization techniques based on trace or observational equivalences just to mention a few. Graph's transitions associated with synchronization actions are labeled by action's name. Other transitions are labeled by "nil". The minimization process discards as much "nil" as allowed by the equivalence relation and outputs a quotient automaton which gives an abstract view of the system's behavior.
Minimization particularly applies to communication architecture validation. Given a protocol layer modeled in TURTLE, a labeled RG is generated (RTL, CADP) and minimized considering service primitives exchanges as observable events. The minimization outputs a quotient automaton of the service rendered by the protocol layer.

C. Observer guided verification

Adding observers to TURTLE design diagrams enables characterization of temporal requirement violation in the form of RG transitions that may be searched for by TTool.

D. Temporal requirement traceability

TURTLE includes SysML Requirement Diagrams and extend them with formal specification of time-related properties. TTool uses that formal specification to automatically include observers into the TURTLE model. Then, it generates a traceability matrix indicating which temporal requirements is satisfied or not.

V. Conclusions

TTool offers to real-time UML practitioners a user-friendly interface to well established formal verification techniques. It has been used in various projects and for education purposes. Unlike model transformation tools such as Topcased [10], TTool lies in the category of UML tools based on profiles. Ongoing work address methodological assistants [11] based on patterns.

References

[1] O. M. Group, "UML 2.0 Superstructure Specification," in *http://www.omg.org/docs/ptc/03-08-02.pdf*, Geneva, 2003.
[2] A. Gherbi and F. Khendek, "UML Profiles for Real-Time Systems and their Applications," *Journal of Object Technology*, vol. 5:3, pp. 149–169, 2000.
[3] LabSoc, "The TURTLE Toolkit," in *http://labsoc.comelec.enst.fr/turtle/ttool.html*.
[4] L. Apvrille, C. Lohr, J.-P. Courtiat, and P. de Saqui-Sannes, "TURTLE: A Real-Time UML Profile Supported by a Formal Validation Toolkit," in *IEEE transactions on Software Engineering*, vol. 30, no. 7, July 2004, pp. 473–487.
[5] "The RTL toolkit," http://www.laas.fr/RT-LOTOS/index.html.en.
[6] "The CADP toolkit," http://www.inrialpes.fr/vasy/cadp.
[7] "The UPPAAL toolkit," http://www.uppaal.com/.
[8] O. M. Group, "UML Profile for Systems Engineering, SysML, Version 1.0," in *http://www.omg.org/cgi-bin/apps/doc?formal/07-09-01.pdf*, Geneva, Sept. 2007.
[9] L. Apvrille, "TTool for DIPLODOCUS: An Environment for Design Space Exploration," in *Proceedings of the 8th Annual International Conference on New Technologies of Distributed Systems (NOTERE'2008)*, Lyon, France, June 2008.
[10] "Topcased project," http://topcased.gforge.enseeiht.fr/.
[11] L. Apvrille and P. de Saqui-Sannes, "Adding a Methodological Assistant to a Protocol Modeling Environment," in *Proceedings of the 8th Annual International Conference on New Technologies of Distributed Systems (NOTERE'2008)*, Lyon, France, June 2008.