

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

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— The TURTLE toolkit, or TTool for short, offers a real-time UML front-end and a user-friendly interface to simulation techniques and formal verification techniques such as reachability analysis, observer-based analysis and traceability matrices generation. TTool's main strength is the total hiding of formal languages to UML modelers, while offering formal verification capabilities.

I. INTRODUCTION

Real-Time UML profiles extend the Unified Modeling Language [1] with formal semantics, tools and methods that answer real-time systems development needs. Real-time UML tools designed as front-ends to formal verification tools, enable early checking of UML models against design errors. The TURTLE Toolkit (TTool [2]) initially developed for the TURTLE real-time UML profile [3] implements interfaces to RTL [4], CADP [5] and UPPAAL [6] verification tools. TTool additionally facilitates temporal requirement verification [7]. This paper presents the latest release of TTool - called TTool version 0.91 - , discusses the way it hides formal languages and verification tools to end users, and at last explains why formal verification is made amenable to practitioners, in particular in education context.

The paper is organized as follows. Section 2 introduces the verification-centric method associated with the TURTLE UML profile. Section 3 overviews TTool. Section 4 discusses how the latest release of the tool offers user-friendly access to formal verification. Section 5 concludes the paper.

II. TURTLE METHOD

TTool supports the seven-step, verification-centric method that is recommended for the TURTLE modeling language.

- 1) **Requirement capture.** SysML requirement diagrams [8] capture informal requirements as well as formal temporal requirements. Timing diagrams are used to formally express temporal requirements.
- 2) **Use-case driven analysis.** Use-cases expressed in use-case diagrams are documented by scenarios (sequence diagrams) structured by interaction overview diagrams.
- 3) **Formal verification of analysis diagrams.** In particular, scenarios may be checked 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 by UML activity diagrams extended with temporal operators: deterministic delay, non-deterministic delay, and time-limited synchronization offer.
- 5) **Object-oriented design.** Class/objects and activity diagrams automatically generated in previous steps are enriched.
- 6) **Formal verification of design diagrams,** particularly against temporal requirements.
- 7) **Rapid prototyping** based on component and deployment diagrams. Objects defined at design step are first organized into components which are themselves deployed over execution nodes.

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

The open-source toolkit *TTool* supports several UML2 / SysML profiles, in particular TURTLE [3] and DIPLODOCUS [9]. The main idea behind TTool is that any model expressed in a UML2 profile may be formally verified using RTL, CADP or UPPAAL (see Figure 1). In practice, TTool translates UML diagrams into an intermediate format expressed in a formal language called TIF¹, and invokes one of its code generators to provide LOTOS, RT-LOTOS or UPPAAL code to CADP, RTL or UPPAAL tool, respectively.

IV. FORMAL VERIFICATION

Unlike real-time UML tools that exclusively generate formal code from design diagrams, TTool also enables formal verification of analysis UML diagrams. Thus, someone unfamiliar with object-oriented design may use TTool and apply formal verification to use-case driven and scenario based analysis. The rest of the section describes a verification approach that indifferently apply to design or analysis diagrams of the TURTLE profile. Those techniques also apply to other profiles such as the DIPLODOCUS UML profile.

¹TIF stands for TURTLE Intermediate Format

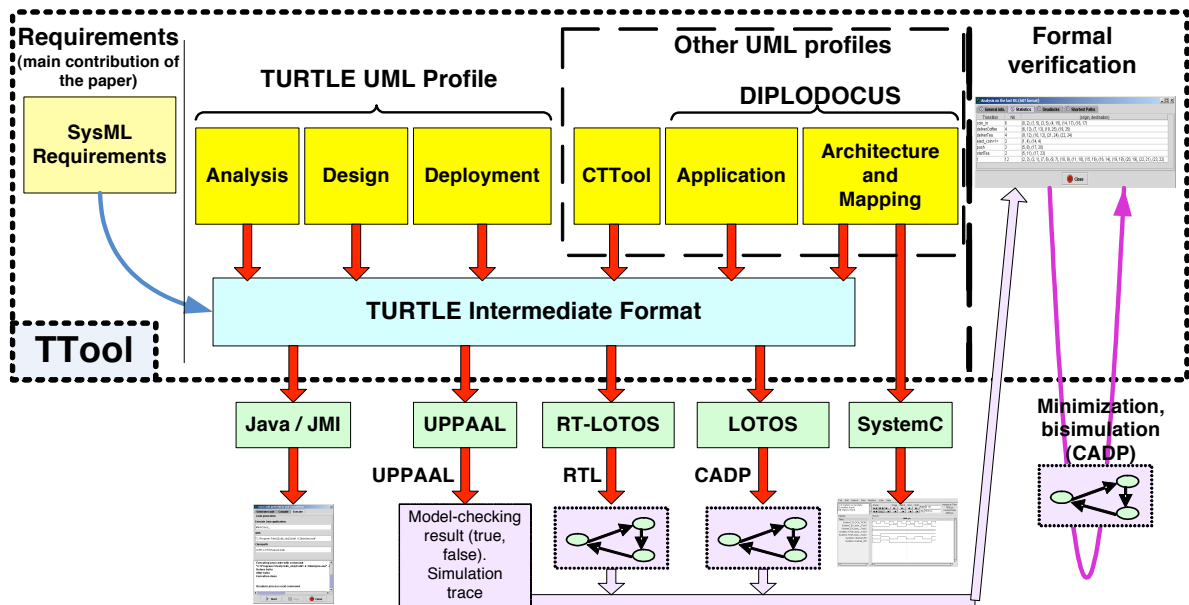


Fig. 1. TTool: profiles and verification techniques

A. Reachability analysis

TTool has been interfaced to verification tools that implement reachability analysis, a technique which computes the set of stable states the system may reach from its initial state. The rest of the paper assumes a reachability graph may be computed in reasonable time.

CADP, a tool developed for an untimed version of LOTOS, enables quick generation of reachability graphs; nevertheless, temporal information of the original model is lost. Conversely, RTL and UPPAAL take the temporal operators - including non-deterministic temporal operators - of TURTLE models into account.

TTool not only invokes a verification tool it has catered with appropriate formal code, it also offers user-friendly interfaces to exploit the reachability graphs computed by CADP or RTL. Thus, TTool computes statistics on states and transitions. It also identifies deadlocks as well as shortest and longest paths in the graph. Also, TTool uses *dotty* to display graphs. The latter contain identifiers that may not coincide with the identifiers used in the TURTLE model. Therefore, TURTLE provides a conversion table which allows one to trace identifiers from TURTLE models to formal code and reachability graphs.

In practice, it does not suffice to display a reachability graph to decide whether some property is met or not. The reachability graph of real-size systems may indeed have millions of states and transitions. Logic-based model checking and minimization are two complementary techniques offered by TTool and its companion tools.

B. Model checking

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: The UPPAAL's verifier is invoked with corresponding CTL formulae, and the result of reachability / liveness properties is displayed. Temporal logic formulae in CTL may also be entered directly in TTool.

C. Minimization of labeled reachability graphs

A reachability graph 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" transitions 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 reachability graph is generated (RTL, CADP) and minimized by considering service primitives exchanges as observable events. The minimization thus outputs a quotient automaton of the service rendered by the protocol layer.

D. Observer guided verification of temporal requirements

An observer is a TURTLE object manually or automatically [7] included in the TURTLE model of the system in order to drive formal verification. Given an observer O checking a TURTLE model against requirement R, O must behave as follows: each time R is violated, a transition label in the reachability graph must unambiguously identify R's violation.

Quick search of requirement violation labels thus enables identification of unmet requirements.

Observer-guided verification particularly applies to temporal requirements expressed by extended timing diagrams [7]. The tool not only synthesizes observers from temporal requirements: It also creates a two-column (requirement, met/unmet) traceability matrix.

V. CONCLUSIONS

TTool offers real-time UML practitioners a user-friendly interface to formal verification techniques. It has been used in various projects and for education purpose. Unlike model transformation tools such as Topcased [10], TTool falls in the category of UML tools based on profiles.

A stable and open-source version of the tool is available from [2]. Next version of the tool will include methodological assistants [11] based on patterns. Links between TTool and other model-checking toolkits are also under study.

REFERENCES

- [1] O. M. Group, "UML 2.0 Superstructure Specification," in <http://www.omg.org/docs/ptc/03-08-02.pdf>, Geneva, 2003.
- [2] LabSoc, "The TURTLE Toolkit," in <http://labsoc.comelec.enst.fr/turtle/ttool.html>.
- [3] 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.
- [4] "The RTL toolkit," <http://www.laas.fr/RT-LOTOS/index.html.en>.
- [5] "The CADP toolkit," <http://www.inrialpes.fr/vasy/cadp>.
- [6] "The UPPAAL toolkit," <http://www.uppaal.com/>.
- [7] B. Fontan, "Mthodologie de conception de systmes temps rel et distribus en contexte UML/SysML," in *Doctorat de l'Universit de Toulouse dlviv par l'Universit Paul Sabatier*, January 2008.
- [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.enseiht.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.