Teaching Formal Methods: Experience at UPMC and UP13 with *CosyVerif*

Étienne André, Laure Petrucci
Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS UMR 7030, F-93430, Villetaneuse, France
Email: Etienne.Andre,Laure.Petrucci@lipn.univ-paris13.fr
Fabrice Kordon
Sorbonne Universités, Université Pierre et Marie Curie, LIP6, CNRS UMR 7606, F-75005, Paris, France
Email: Fabrice.Kordon@lip6.fr

***Abstract*—Nowadays, students are more and more demanding for practical coursework, which is a challenge when teaching formal approaches to software engineering. The solution is to provide environments for such hands-on sessions and homework, but this raises numerous difficulties. The environment must be: (i) multi-platform (Mac OS, Linux, Windows) so as to enable student practice at home, (ii) easy to deploy, (iii) easy to use and to take charge of, and (iv) flexible enough to enable the integration of new notations and associated services.**

***CosyVerif* is a software environment dedicated to graphical notations, that provides the mechanisms and means for an easy integration of additional existing software for teaching (or demonstration) purposes. This makes it an interesting platform to establish new courses.**

This paper presents our experience using *CosyVerif* for teaching Petri nets and parametric timed automata in two universities of the Paris region, i.e. Université Pierre et Marie Curie, and Université Paris 13. We also use *CosyVerif* to build demonstrators of Ph.D. students' work.

***Keywords*—Formal methods, Software platform, Master courses**

I. Introduction

Nowadays, students are more and more demanding for practical coursework, which is a challenge when teaching formal approaches to software engineering. The solution is to provide environments for such hands-on sessions and homework, but this raises numerous difficulties. The environment must be: (i) multi-platform (Mac OS, Linux, Windows) so as to enable student practice at home, (ii) easy to deploy, (iii) easy to use and to take charge of, and (iv) flexible enough to enable the integration of new notations and associated services.

This paper reports our experience of teaching Petri nets and parametric timed automata in Master courses, in two universities of the Paris region, using the *CosyVerif* [1] platform. *CosyVerif* is a software environment dedicated to graphical notations.[1] It is a client/server environment providing numerous ways to configure labs for students (one server and several clients on students' machines, or both the client and the server running on the same machine). A Linux-based server infrastructure embeds both the core functions and provided services within a virtual machine (virtualisation is now available even for low-cost machines). Hence, only the user interface has to be tuned for a given OS, which is easily done thanks to Eclipse. This interface thus provides an easy solution for students to take charge of and use the environment.

CosyVerif offers flexibility in its usage and deployment. First, if embedded services are time and memory consuming, the server can be deployed on a powerful machine, while the graphical user interface is run on the students client machines. Furthermore, a bundle can group altogether the graphical user interface and the elements required to operate the server in a virtualisation machine. The advantage of these bundles is to be *zeroconf*, i.e. they do not require manual configuration. Moreover, they are easy to parametrise in order to extract only a subset of dedicated services from a larger catalogue. In our teaching experiments, UPMC used *CosyVerif* for Petri nets and UP13 used it for parametric timed automata [2].

Finally, the *CosyVerif* architecture provides the mechanisms and means for an easy integration of additional existing software for teaching (or demonstration) purposes. This makes it an interesting platform to establish new courses. We also use *CosyVerif* to build demonstrators of Ph.D. students' work.

Outline: We briefly describe the universities and their respective Master courses in Section II. The *CosyVerif* platform is presented in Section III. Then, our experience of teaching formal methods using *CosyVerif* is described in Section IV. Finally, we conclude and give some perspectives in Section V.

II. UPMC and UP13 Master Courses

A. Master SAR at UPMC

Université Pierre & Marie Curie (UPMC[2]) is the heir of the Sorbonne's historical faculty. It hosts about 33,000 students dispatched in science and medicine only. It was the first university in France to open a degree in Computer Science (1967) and it is ranked 37th worldwide, 6th in Europe and 1st in France in the Academic Ranking of World Universities (2013[3]).

The Computer Science Teaching Department is highly ranked and hosts about 800 students in master (400 in 1st year

[1] http://www.cosyverif.org

[2] http://www.upmc.fr

[3] http://www.shanghairanking.com/ARWU2013.html

and 400 in 2nd year). It offers seven tracks among which SAR (standing for *Systèmes et Applications Répartis*, i.e. Distributed Systems and Applications[4]). This track is dedicated to students willing to learn the design and implementation of complex systems including distributed, operating-system-based, real-time and critical features.

Within this master track, we are responsible for courses dealing with the modelling and analysis of behaviours for parallel programs. Our courses are based on several formal notations: various types of automata [3] and Petri nets [4].

To understand and capture the essence of such notations, several practical sessions, as well as modelling and analysis projects, have been elaborated. Since 2004, we were using CPN-AMI [5], a Petri net based modelling and analysis environment developed in our research laboratory. CPN-AMI gathered several tools from several universities within a common graphical user interface and was a useful tool for such courses. However, the user interface was developed under MacOS, which hindered its use by students outside our dedicated computer room.

The switch to *CosyVerif* was a great progress in that direction since it is multi-platform. Numerous tools from CPN-AMI were ported to *CosyVerif*.

B. Master PLS at UP13

Université Paris 13[5] (UP13) is a multidisciplinary establishment comprising four campuses situated in the northern suburbs of Paris. It offers its 23,000 students a wide spectrum of subjects. They address five major domains: Humanities and Social sciences ; Science, Technology and Health ; Culture and Communication ; Law, Economics and Management ; Arts, Literature and Languages.

The Institut Galilée is the department of the UP13 focussing on education and research in sciences: mathematics, physics, chemistry and informatics. It delivers degrees from bachelor to doctorate, as well as engineering. Research is carried in 7 laboratories within the different areas of expertise.

Institut Galilée proposes the Master of Science in Informatics[6] with two tracks: EID^2 (standing for *Exploration Informatique des Données et Décisionnel*, i.e. Data Mining, Analytics and Knowledge Discovery), and PLS (standing for *Programmation et Logiciels Sûrs*, i.e. Programming Tools and Safety).

Within Master PLS, we are responsible for the SITH module (standing for *Systèmes Infinis, Temporisés et Hybrides*, i.e. Infinite, Timed and Hybrid Systems). In 2013–2014, the total number of students in the Master PLS was 30, and the number of students attending the module SITH was 20.

The content of the course concerns formal verification, and more specifically the verification of concurrent timed systems, and parametric timed systems. In short, the objectives of the course are to become familiar with timed automata [6], parametric timed automata [2], understand the associated (possibly parametric) model checking algorithms, as well as being able to model a system using these formalisms and verify it using available tools. Such tools include UPPAAL [7] (for timed automata) and IMITATOR [8] (for parametric timed automata). Whereas UPPAAL is a well-established tool featuring a convenient graphical interface, parameter synthesis techniques are more state-of-the-art, and hence tools such as IMITATOR are more confidential and less mature, especially in terms of usability. The recent integration of IMITATOR within the *CosyVerif* platform made it possible to use it in a graphical manner (otherwise, IMITATOR is command-line only), and hence the students were able to design their model using a graphical user interface. In contrast, for SITH's previous occurrences, the practical sessions had to be done in command-line mode only, which is by far less intuitive to students.

[4]http://www-master.ufr-info-p6.jussieu.fr/lmd/specialite/sar/
[5]http://www.univ-paris13.fr
[6]http://lipn.univ-paris13.fr/~bennani/Web_Master_Info/coord.html

III. THE *CosyVerif* PLATFORM

A. Principles

CosyVerif [1] is a distributed and open verification environment that currently handles two families of formalisms: Petri nets and timed automata. So far, 12 declared concrete formalisms from these two families are available, interrelated through a modular architecture of definitions, reusing common concepts, and enabling easy addition of new notations. They are syntactically supported by a two-layered XML-based language: the Formalism Markup Language (FML, the superstructure) and the Graph Markup Language (GrML, the infrastructure) [9].

Tools developers can declare a new formalism in the platform using FML, by reusing portions of existing formalisms (when they share common concepts). GrML is the internal representation of specifications in *CosyVerif*. FML and GrML ensure syntactic interoperability among tools that may only manipulate abstract syntax trees. These XML-based technologies enable rapid development and reuse of parsers and syntactic validation. Thanks to such facilities, the typical integration effort for tools developers is half a day.

CosyVerif is an open distributed environment that can be enriched by any researcher willing to contribute. A registration mechanism allows for the diffusion of any service over a federation of *CosyVerif* nodes, which greatly improves the time-to-availability for new tools.

Tools are invoked through Web services transparently to end users, thanks to Coloane, an open source extensible graphical editor based on Eclipse. It offers modelling facilities and a way to apply tools services to models. Since *CosyVerif* relies on Web services, the use of Coloane is not mandatory and verification services can be accessed directly via the underlying XML-based protocol.

The *CosyVerif* project also provides a repository of models, that may be used for benchmark purposes. These models mostly come from industrial real-time case studies and the Model Checking Contest editions of 2011 [10], 2012 [11] and 2013 [12].

B. The CosyVerif Bundles

CosyVerif is a client/server software: the user interface can run on a relatively light machine, and for research experiments, tools, hosted by a Unix-based server, may have to be operated on powerful machines. So, the installation is rather complex since it requires a server machine and a client.

However, less computing strength being needed within the context of teaching and most common computers (including laptops) being good enough to support virtualisation, we decided to propose easy means to operate a version of *CosyVerif* in the form of a bundle including: (i) the Coloane user interface, (ii) a disk image containing the installation of the tool server (under Unix), (iii) several scripts to handle an easy execution within a standard and free virtualisation solution: VirtualBox[7].

These bundles are distributed for the three main architectures one can expect for students and researchers: Linux, MacOS and Windows. A single script operates both the tool server and the user interface transparently (as soon as VirtualBox is installed). For Mac, this script is embedded in a MacOS application, allowing an easy installation of the software using drag-and-drop.

So far, two bundles are proposed: *CosyVerif4PN* (used at UPMC to teach Petri Nets) and *CosyVerif4imitator* (used at UP13 to teach parametric timed automata).

C. *Demonstration of Ph.D. Students' Work*

Bundles are quite easy to make and may feature a selected subset of the services available in *CosyVerif*. Hence, another interesting use is to make bundles dedicated to a given tool. This appears to be an interesting promotion way for students who can then distribute their own work as a standalone product to the members of their jury. This will be soon experimented for Ph.D. students in LSV (École Normale Supérieure de Cachan) and LIPN (UP13).

IV. Summary of our Experience

A. *First experiments*

A first experimentation of *CosyVerif* was carried out during the Petri net tutorials of IWAISE'2012 [13]. Despite the use of a beta version, installation by students (bundles were not available yet) could be performed and a lab session organised. Another experimentation was performed during a summer school on real-time systems in Toulouse (France) in August 2013, which was prefiguring what was to be used at UPMC and UP13 for the current academic year (2013–2014).

The experimentations within the context of a master programme raises a new issue since for the tutorial and the summer school the public was mainly composed of Ph.D. students with more background, motivation and practice.

B. *Configuration and Problems at UPMC*

Security was the main issue for *CosyVerif* at UPMC. The computer rooms for the Computer Science teaching department hosts approximately 350 machines (mostly PCs under dual-boot Linux/Windows and some Macs) and is open to more than 1,500 students. So, virtualisation is a problem and our software had to be installed in a way so that no student could operate VirtualBox with a home-made disk image corresponding to a machine where they could be `root` (and thus potentially hack the other machines).

Thus, the default installation of VirtualBox, as suggested for students on their own machines, could not be operated in that context. The system administrator had to adapt our bundle and, in particular, its main script, to protect the system from potential misuse.

The solution was to have a script allowing secure access to the virtualisation environment with the only authorised configuration (the authorised VM). The students were then declared as part of a `sudo` group reserved for this unique usage. Some details on the configuration where also fixed to set-up the virtual machine on a local virtual network that could not go outside the virtualisation environment.

This solution was operated in October 2013 and was successfully used by the 25 students attending the course for both their practical work and a small project they could do at home or off-hours in our computer lab. The only observed problem was, during the first session of practical work, a misuse of permission access that could lead to a crash. This was in fact a problem with the Eclipse libraries embedded in the Coloane user interface; a corrective patch was published and a new version of the bundle made available.

Feedback: The use of *CosyVerif* appears to have been satisfactory to our students. We observed a small download peak when practical sessions started, which is an evidence of students using *CosyVerif* on their own machines. No major problem was reported during both lab sessions and the project. Finally, all expected projects where submitted on time by the students.

C. *Configuration and Problems at UP13*

The practical sessions took place in a dedicated room, with 24 machines using a 64 bits single-boot Linux.

A good point is that VirtualBox was installed by default on all machines, thus allowing us to install the *CosyVerif* bundle very easily. However, a problem arised as the students' disk limit was rather low: around 800 MiB. Since most of the students already used at least half of the their disk quota, and the bundle's size is 951 MiB, we encountered space problems. Even downloading the bundle, that is only 476 MiB when compressed, was impossible (the download process froze as soon as the limit was reached). Fortunately, the `/tmp` directory was in read and write mode, without any specified space limit (only the disk size). Hence, we could download the bundle there, extract it and launch it from there.

The complete installation process took about an hour during the first session, that is it took about an hour until all students had the *CosyVerif* environment running. This time also includes the search for solutions following the discovery of the space limit problem. (The configuration could not be tested beforehand, since teachers have different account configurations than students.)

To avoid the loss of documents (the `/tmp` directory is entirely erased after *each* reboot), the students could set their *CosyVerif* root directory (where they save their models) in their home directory. However, in case of reboot, the bundle

[7]http://www.virtualbox.org

should be again copied and extracted. When starting the second session, it took less than ten minutes until all students were able to extract and launch again *CosyVerif*.

We also tried another solution, that was to launch the *CosyVerif* from the students' USB stick. Although this could be possible in theory, it unfortunately failed for almost all of them, due to a formatting issue (their sticks were usually in FAT 32, a file system format that does not handle permissions, and a possible explanation is that the *CosyVerif* bundle requires to set some permissions to be able to launch).

Surprisingly, the students that encountered most problems were the students that brought their own laptop. This is mainly due to the fact that there is no possibility to connect to the Internet in the computers classrooms of Institut Galilée: no WiFi is available in classrooms, and only the university's computers can connect to wired Internet (a possibility could have been to usurpate the MAC address of a university computer, but we usually do not provide this kind of solution to our students). Although these students had downloaded VirtualBox and *CosyVerif* beforehand, they sometimes encountered problems that could not be solved without an Internet connection (e.g. a deprecated version of VirtualBox, or a missing package, such as the `dot` utility required by IMITATOR to output graphical results).

Feedback: In the Master PLS, a post-course anonymous evaluation showed that 87 % of the students attending the SITH module were satisfied or very satisfied by their experience in practical sessions using *CosyVerif* and IMITATOR.[8]

V. Conclusion and Perspectives

This paper presented experiences conducted with the *CosyVerif* environment for teaching formal methods, in two universities of the Paris area, namely Université Pierre et Marie Curie (UPMC) and Université Paris 13 (UP13).

The design of the *CosyVerif* environment made it possible to have a flexible tool adapted to each specific course. Bundles thus only embed the necessary components of the platform (Petri nets tools at UPMC, parametric timed automata at UP13), for an easier distribution and installation. It also can be run on several clients architectures, thanks to the use of virtual machines.

Most of the problems encountered in these experimental sessions were due to permissions granted to students in the respective universities. Fallback solutions were quite easily found, allowing for the complete use of the *CosyVerif* environment.

Short-term perspectives include the use of *CosyVerif* at a full-day tutorial associated with the Petri nets international conference in June 2014[9].

The environment aims at supporting more tools and formalisms. The experience conducted will be disseminated so that other universities or organisations can make their own bundles, fitted to their specific needs.

We are also working on a formalisms and models repository, in order to share examples, case studies, etc. that will facilitate lecturing and students' work.

Acknowledgement: We thank the engineers of UPMC for their precious help in solving the security issues and the engineers of Paris 13 for their suggestions. We also thank all the students of the SITH and SAR Master courses for their motivation.

[8] The evaluation was performed after the final exam, via an anonymous form to fill in. 2 students missed the exam, 3 did not fill in the form, hence we received 15 evaluations. None found the practical sessions "not interesting at all", 2 found it "not very interesting", 10 "interesting" and 3 "very interesting".

[9] http://petrinets2014.cnam.fr/

References

[1] É. André, L.-M. Hillah, F. Hulin-Hubard, F. Kordon, Y. Lembachar, A. Linard, and L. Petrucci, "CosyVerif: An open source extensible verification environment," in *ICECCS*. IEEE Computer Society, 2013, pp. 33–36. 1, 2

[2] R. Alur, T. A. Henzinger, and M. Y. Vardi, "Parametric real-time reasoning," in *STOC*. ACM, 1993, pp. 592–601. 1, 2

[3] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen, *Systems and Software Verification. Model-Checking Techniques and Tools*. Springer, 2001. 2

[4] C. Girault and R. Valk, *Petri Nets for Systems Engineering : A Guide to Modeling, Verification, and Applications*. Springer-Verlag, 2003. 2

[5] A. Hamez, L. M. Hillah, F. Kordon, A. Linard, E. Paviot-Adet, X. Renault, and Y. Thierry-Mieg, "New Features in CPN-AMI 3 : Focusing on the Analysis of Complex Distributed Systems," in *ACSD*. IEEE Computer Society, 2006, pp. 273–275. 2

[6] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994. 2

[7] K. G. Larsen, P. Pettersson, and W. Yi, "UPPAAL in a nutshell," *International Journal on Software Tools for Technology Transfer*, vol. 1, no. 1-2, pp. 134–152, 1997. 2

[8] É. André, L. Fribourg, U. Kühne, and R. Soulat, "IMITATOR 2.5: A tool for analyzing robustness in scheduling problems," in *FM*, ser. Lecture Notes in Computer Science, vol. 7436. Springer, 2012, pp. 33–36. 2

[9] É. André, B. Barbot, C. Démoulins, L. M. Hillah, F. Hulin-Hubard, F. Kordon, A. Linard, and L. Petrucci, "A modular approach for reusing formalisms in verification tools of concurrent systems," in *ICFEM*, ser. Lecture Notes in Computer Science, vol. 8144. Springer, 2013, pp. 199–214. 2

[10] F. Kordon, A. Linard, D. Buchs, M. Colange, S. Evangelista, K. Lampka, N. Lohmann, E. Paviot-Adet, Y. Thierry-Mieg, and H. Wimmel, "Report on the model checking contest at Petri nets 2011," *Transactions on Petri Nets and Other Models of Concurrency*, vol. VI, pp. 169–196, 2012. 2

[11] F. Kordon, A. Linard, D. Buchs, M. Colange, S. Evangelista, L. Fronc, L.-M. Hillah, N. Lohmann, E. Paviot-Adet, F. Pommereau, C. Rohr, Y. Thierry-Mieg, H. Wimmel, and K. Wolf, "Raw report on the model checking contest at Petri nets 2012," Tech. Rep., 2012, coRR. 2

[12] F. Kordon, A. Linard, M. Beccuti, D. Buchs, L. Fronc, F. Hulin-Hubard, F. Legond-Aubry, N. Lohmann, A. Marechal, E. Paviot-Adet, F. Pommereau, C. Rodrigues, C. Rohr, Y. Thierry-Mieg, H. Wimmel, and K. Wolf, "Model Checking Contest @ Petri Nets, Report on the 2013 edition," CoRR, Tech. Rep., September 2013. 2

[13] S. Baarir and F. Kordon, "Modeling and Verifying Distributed Systems with Petri Nets," in *2nd IEEE International Workshop on Advanced Information Systems for Enterprises (IWAISE)*. Constantine, Algeria: IEEE Press, November 2012, p. 92. 3

[14] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[15] K. Jensen and L. M. Kristensen, *Coloured Petri Nets – Modelling and Validation of Concurrent Systems*. Springer, 2009.

[16] P. M. Merlin, "A study of the recoverability of computing systems." Ph.D. dissertation, University of California, Irvine, CA, USA, 1974.