Architecting Trustworthy Self-adaptive Systems

Radu Calinescu*, Danny Weyns†, Simos Gerasimou* and Ibrahim Habli*
*Department of Computer Science & Assuring Autonomy International Programme, University of York, UK
Email: {radu.calinescu,simos.gerasimou,ibrahim.habli}@york.ac.uk
†Department of Computer Science, Katholieke Universiteit Leuven, Belgium
Email: danny.weyns@kuleuven.be

I. Summary

Architecting self-adaptive software systems is challenging. These systems must achieve their goals not only in the environment in which they are deployed initially, but also as this environment changes over time. When self-adaptive systems are used in safety-critical and business-critical applications, this challenge is compounded by the need to also provide guarantees that the system operates correctly at all times. For traditional software, such guarantees are provided through *assurance cases*. These are structured arguments which use comprehensive development-time evidence to explain why a system can be trusted when used for its planned application in a given environment. This tutorial will present the ENTRUST methodology for achieving a similar level of trust in self-adaptive systems. ENTRUST represents the first end-to-end methodology for architecting trustworthy self-adaptive systems and *dynamic assurance cases* guaranteeing the suitability of the software for its intended applications. As advocated by major research initiatives such as the UK-led Assuring Autonomy International Programme and the US Assured Autonomy program, ENTRUST dynamic assurance cases seamlessly combine evidence obtained during the development of a self-adaptive system with evidence obtained from its additional verification at runtime. As such, each dynamic reconfiguration of an ENTRUST self-adaptive system is accompanied by a new version of the assurance case that confirms the correctness of the reconfigured system architecture.

The tutorial will start with an overview of self-adaptive systems used in safety-critical and business-critical applications. This will be followed by an introduction to assurance cases, an explanation of the recent paradigm shift to dynamic assurance cases, and the description of the ENTRUST methodology and of its use of the model checkers UPPAAL and PRISM at different stages of the self-adaptive system lifecycle. We will conclude by showing how ENTRUST can be used to engineer a self-adaptive unmanned underwater vehicle system and a self-adaptive service-based system.

The tutorial will actively engage attendees and include practical demonstrations. Attending it will benefit researchers and software architects with an interest in self-adaptive and autonomous software systems, as well as those interested in the rigorous modelling, analysis and verification of the control software of such systems.

II. Topic description and relevance for the ICSA community

(i) Description of the topic

Self-adaptive software systems modify their architecture and parameters at runtime, in response to changes in the environment. As such, they can continue to achieve user-specified goals despite variations in workload, component failures, changes in available resources, etc. The demand for these systems has grown rapidly in the past decade, as such changes are increasingly common in applications that use software in healthcare, transportation, finance, e-commerce and numerous other domains. Many of these applications are safety critical or business critical. Goal violations can result in harm to patients in a self-adaptive system that monitors the vital signs of sufferers from chronic conditions like diabetes or high blood pressure [1]. Alternatively, goal violations may lead to expensive equipment damage in a self-adaptive system of unmanned vehicles pursuing a search operation [2].

Accordingly, the provision of assurances for these self-adaptive systems was recently identified as a priority by an international research team that includes many prominent members of the ICSA community [3]. Our proposed tutorial will present ENTRUST [4], a newly developed methodology for the systematic provision of such assurances.

(ii) State of the art in the topic

As assurance has become a major concern for self-adaptive software only recently, the research in the area is limited. The state of the art is typically confined to providing evidence that individual aspects of the self-adaptive software are correct. These aspects include the software platform used to execute the system controller [5], the controller functions [6], [7], or the runtime adaptation decisions [8], [9], [10]. The assurance evidence for each aspect can be obtained using methods that range from formal proof techniques [11] and model checking [6] to simulation [10] and testing [12].

However, this evidence represents only one component of the established industry process for the assurance of software-based systems used in critical applications [13]. In real-world applications, assuring a software system requires the provision of an *assurance case*, i.e., a structured argument that use comprehensive development-time evidence to explain why a system can be trusted when used for its planned application in a given environment. ENTRUST, the methodology for architecting trustworthy self-adaptive systems we will present in the tutorial, is the first approach that addresses this discrepancy between the industrial practice and the current research on assurances for self-adaptive software. To this end, ENTRUST uses a combination of (1) development-time and runtime modelling, testing and verification, and (2) an industry-adopted standard for the formalisation of assurance arguments [14].

ENTRUST uses development-time modelling, verification and synthesis of assurance evidence for the control aspects of a self-adaptive system that are engineered before the system is deployed. These activities enable the generation of a partial assurance case for the self-adaptive system. The dynamic selection of a system configuration (i.e., architecture and parameters) during the initial deployment and after internal and environmental changes involves further modelling and verification, and the synthesis of the additional assurance evidence required to complete the assurance case. These activities are fully automated and carried out at runtime.

(iii) Intended audience

The tutorial targets attendees from academia and industry with an interest in:

- Understanding the motivation and need for trustworthy self-adaptive systems;
- Learning about dynamic assurance cases and there application to safety- and business-critical software;
- Getting familiar with techniques and methods for the architecting of trustworthy self-adaptive systems;
- Learning about the architecting of concrete cases of trustworthy self-adaptive systems;
- Gaining insights into the open challenges from the field of trustworthy self-adaptive systems.

(iv) Relevance for ICSA

The tutorial is closely aligned with multiple topics of interest from the ICSA-2019 call for papers,[1] including:

- *Model driven engineering for continuous architecting*. The use of model-driven engineering is one of the two underpinning principles of the ENTRUST methodology;
- *Architecting Systems of Systems, IoT systems, CPSs, software ecosystems, self-adaptive systems, or autonomous systems*. The tutorial addresses on the architecting of self-adaptive systems.
- *Component based software engineering and architecture design*. ENTRUST is a component-based software engineering methodology.

III. Implementation

(i) Duration of the proposed tutorial

We propose a full-day tutorial. As for the ICSA-2018 tutorials, this will include four 1.5-hour tutorial sessions (six hours in total), with coffee breaks and lunch between sessions.

[1]https://swk-www.informatik.uni-hamburg.de/~icsa2019/call-for-papers/technical-papers/index.html

(ii) Preliminary schedule of events

The planned schedule of the tutorial is detailed below.

Session 1 (1.5 hours)

0:00–0:30	*Introduction to self-adaptive software systems used in safety- and business-critical applications*
0:30–1:30	*A primer on assurance case development*

Coffee break

Session 2 (1.5 hours)

0:00–0:45	*The ENTRUST methodology for architecting trustworthy self-adaptive systems*
0:45–1:30	*Development-time stages of ENTRUST*

Lunch

Session 3 (1.5 hours)

0:00–0:45	*Runtime stages of ENTRUST*
0:45–1:30	*Case study 1: ENTRUST development of a self-adaptive unmanned underwater vehicle system*

Coffee break

Session 4 (1.5 hours)

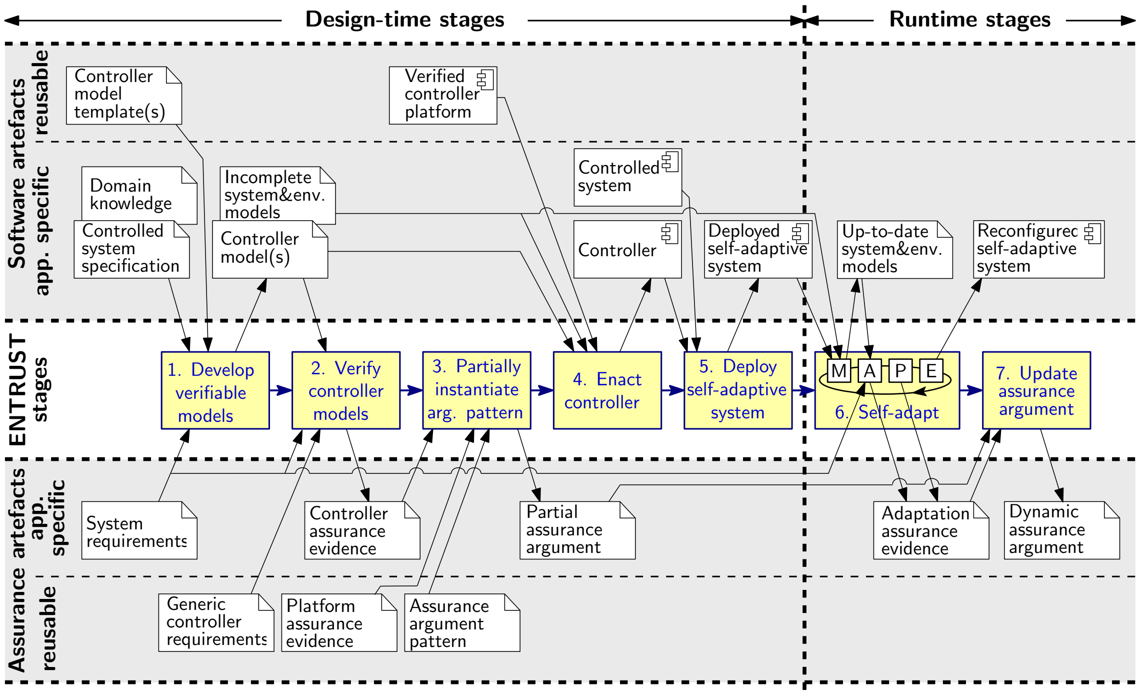
0:00–0:45	*Case study 2: ENTRUST development of a self-adaptive foreign exchange service-based system*
0:45–1:30	*ENTRUST extensions and discussion*

(iii) Justification of the tutorial for the expected audience

With the rapid introduction of the Internet-of-Things and Cyber-Physical Systems in domains ranging from smart environment monitoring to autonomous transportation, the adaptability and trustworthiness of these systems is widely considered as foundational for the future of our society. Hence, architecting trustworthy self-adaptive systems is becoming a central point of interest and importance for the ICSA community. The particular perspective of dynamic assurance cases as driver for trustworthiness and the ability of systems to dynamically adapt their architecture in response to uncertainties and changes at runtime will provide the audience of this tutorial new insights and knowledge for their future research and practice.

(iv) Detailed description of what the tutorial will cover

Session 1 will comprise two parts. Part one will introduce the challenges faced by self-adaptive systems used in safety- and business-critical applications. We will start by presenting the general architecture of a self-adaptive system. Next, we will describe the types of uncertainty encountered by self-adaptive systems, and explain the need for the provision of perpetual assurances for these systems [15]. In part two, we will cover the use of assurance cases for these types of applications. We will introduce three key concepts associated with assurance [13]: assurance cases, assurance arguments, and assurance argument patterns, and we will present an overview of the Goal Structuring Notation standard [14] for the systematic development of assurance cases. We will end the session with a description of the recently introduced concept of a *dynamic assurance case* [16], and explain the need for using dynamic assurance cases for self-adaptive systems.



Design-time stages
Runtime stages
Software artefacts
reusable
app. specific
ENTRUST stages
Assurance artefacts
app. specific
reusable
Controller model template(s)
Verified controller platform
Domain knowledge
Incomplete system&env. models
Controlled system
Controlled system specification
Controller model(s)
Controller
Deployed self-adaptive system
Up-to-date system&env. models
Reconfigured self-adaptive system
1. Develop verifiable models
2. Verify controller models
3. Partially instantiate arg. pattern
4. Enact controller
5. Deploy self-adaptive system
M A P E
6. Self-adapt
7. Update assurance argument
System requirements
Controller assurance evidence
Partial assurance argument
Adaptation assurance evidence
Dynamic assurance argument
Generic controller requirements
Platform assurance evidence
Assurance argument pattern

Fig. 1. Stages and key artefacts of the ENTRUST methodology (diagram taken from [4]

In the first half of Session 2, we will present the principles underpinning the ENTRUST methodology, and we will introduce the design-time and runtime stage of ENTRUST (Fig. 1). We will describe the key software and assurance artefacts produced in each stage, and explain their role in architecting and assuring a self-adaptive software system. We will then use the second half of the session to describe in detail the development-time ENTRUST stages, and their partial automation using formal methods and the UPPAAL verification tool suite [17]. This will include a presentation of the generic assurance argument pattern that ENTRUST uses for self-adaptive systems, and of the method employed for the partial instantiation of this pattern using the incomplete assurance evidence available for such systems at development time. We will illustrate this stage of the methodology using a running example from the maritime environment monitoring domain, i.e., a self-adaptive system comprising an unmanned underwater vehicle (UUV) that measures a characteristic of the ocean environment such as salinity or temperature.

We will start Session 3 with a detailed description of the runtime stages of ENTRUST, and of the use of runtime probabilistic model checking [18] to fully automate these stages. As part of this description, we will present:

1) the use of probabilistic model synthesis [19], [20] to drive the architectural reconfiguration of the self-adaptive system;
2) the use of assurance evidence generated at runtime (using probabilistic model checking) to devise a complete, dynamic assurance case for the self-adaptive system.

We will end the session with a description of a case study based on the UUV self-adaptive system from the running example. This description will focus on the reconfiguration of the system architecture after disruptive changes in the environment (and after recovery from these adverse events), and on the synthesis of dynamic assurance cases for the reconfigured system.

Finally, in the first half of Session 4 we will present a second application of the ENTRUST methodology, in a case study focusing on a business-critical self-adaptive system. This will be a service-based system used in a foreign exchange trading application, and requiring the dynamic selection of the services that implement its operations. In the last part of the session, we will cover a summary of additional verification methods (e.g., testing, simulation and theorem proving) that can be used to obtain assurance evidence for ENTRUST dynamic assurance cases; and we will engage the tutorial participants in a general discussion about the provision of assurances for self-adaptive software systems.

(v) Explanation of how the tutorial will be conducted

The tutorial will be delivered as a series of four highly interactive tutorial sessions comprising slide-based presentations accompanied by frequent (i) hands-on exercises (to ensure participant engagement and understanding) and (ii) demonstrations of practical uses of the ENTRUST methodology.

IV. Presenters' background

Dr Radu Calinescu is a Senior Lecturer in the Department of Computer Science at the University of York, UK, where he leads a research team developing formal techniques for the modelling, analysis and verification of self-adaptive software systems for safety-critical applications. His research has been funded by grants totalling over £1.75M, and is published in

100+ peer-reviewed research papers. He is also the Safety of AI Theme Lead for the Assuring Autonomy International Programme, a £12M research and training initiative he helped set up in 2018. He has been a Programme Chair of major international conferences (SEAMS 2020, SEFM 2015, ICECCS 2010), and has served on the programme committees of over 50 conferences (including TACAS, ASE and MASCOTS). His University of Oxford PhD thesis won a British Computer Society Distinguished Dissertation Award. He has given keynote and invited presentations at multiple international conferences and workshops, and an ICSA-2017 tutorial with 12 registered participants.

Prof Danny Weyns is affiliated with the Department of Computer Science at KU Leuven, Belgium, where he leads a research team on engineering self-adaptive software systems. He is also affiliated with Linnaeus University Sweden. His current research interest is in runtime assurances for self-adaptive systems using both architecture-based and control-based approaches. Dr. Weyns coordinates a research project on trustworthy decentralized self-adaptive systems and another project on dependable adaptive software for the Internet of Things. He is member of the Editorial Board of ACM Transactions on Autonomous and Adaptive Systems, the Journal on Agents and Multi-Agent Systems, the Journal of Systems and Software, and IEEE Software. He was recently PC chair for VAMOS'19 and SEAMS'18 and has provided tutorials at several international events, including SASO'18 and ASE'15.

Dr Simos Gerasimou is a Research Associate in the High Integrity Systems Engineering group within the Department of Computer Science at the University of York. His research interests are in engineering trustworthy autonomous systems through the development of rigorous tool-supported approaches using model-based analysis, simulation and formal verification. Dr. Gerasimou has published over 20 research papers in peer-reviewed international conferences and journals. His current research, funded by Jaguar Land Rover, focuses on the application of assurance cases in the automotive domain and the development of an engineering framework for assuring the safety of autonomous vehicles. He is regularly reviewing for well-known software engineering and safety journals such as Computing, Software Quality, and Reliability Engineering and System Safety.

Dr Ibrahim Habli is a Senior Lecturer in the Department of Computer Science at the University of York. His research interests are in the design and assurance of safety-critical systems. In 2015, he was awarded a Royal Academy of Engineering Industrial Fellowship through which he collaborated with the English National Health Service on evidence-based means for assuring the safety of digital health systems. Ibrahim's work is highly interdisciplinary, with active collaborative links with clinicians, health scientists, economists and ethicists. His research is empirical and industry-informed, with collaborative projects with organisations such as Rolls-Royce, NASA, Jaguar Land Rover and NHS Digital. Ibrahim has led/co-led research programmes funded by the UK Research Councils, the Royal Academy of Engineering, the EU and industry. He is also the Dynamic Risk Theme Lead for the Assuring Autonomy International Programme, a £12M research and training initiative he helped set up in 2018. He has been a member of several international and national safety standardisation committees (e.g. DO178C, MISRA and BSI).

References

[1] D. Weyns and R. Calinescu, "Tele Assistance: A self-adaptive service-based system exemplar," in *SEAMS'15*, 2015, pp. 88–92.

[2] S. Gerasimou, R. Calinescu, S. Shevtsov, and D. Weyns, "UNDERSEA: An exemplar for engineering self-adaptive unmanned underwater vehicles," in *SEAMS'17*, 2017, pp. 83–89.

[3] R. de Lemos, D. Garlan, C. Ghezzi, H. Giese, J. Andersson, M. Litoiu, B. Schmerl, D. Weyns, L. Baresi, N. Bencomo, Y. Brun, J. Camara, R. Calinescu, M. B. Cohen, A. Gorla, V. Grassi, L. Grunske, P. Inverardi, J.-M. Jezequel, S. Malek, R. Mirandola, M. Mori, H. A. Müller, R. Rouvoy, C. M. F. Rubira, E. Rutten, M. Shaw, G. Tamburrelli, G. Tamura, N. M. Villegas, T. Vogel, and F. Zambonelli, "Software engineering for self-adaptive systems: Research challenges in the provision of assurances," in *Software Engineering for Self-Adaptive Systems III. Assurances*, R. de Lemos, D. Garlan, C. Ghezzi, and H. Giese, Eds. Springer, 2017, pp. 3–30.

[4] R. Calinescu, D. Weyns, S. Gerasimou, M. U. Iftikhar, I. Habli, and T. Kelly, "Engineering trustworthy self-adaptive software with dynamic assurance cases," *IEEE Transactions on Software Engineering*, vol. 44, no. 11, pp. 1039–1069, 2018.

[5] M. U. Iftikhar and D. Weyns, "ActivFORMS: Active formal models for self-adaptation," in *SEAMS'14*, 2014, pp. 125–134.

[6] V. Braberman, N. D'Ippolito, N. Piterman, D. Sykes, and S. Uchitel, "Controller synthesis: From modelling to enactment," in *ICSE'13*, 2013.

[7] J. Camara, R. de Lemos, N. Laranjeiro, R. Ventura, and M. Vieira, "Robustness-driven resilience evaluation of self-adaptive software systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 1, pp. 50–64, Jan 2017.

[8] R. Calinescu, S. Gerasimou, K. Johnson, and C. Paterson, "Using runtime quantitative verification to provide assurance evidence for self-adaptive software," in *Software Engineering for Self-Adaptive Systems III. Assurances*, R. de Lemos, D. Garlan, C. Ghezzi, and H. Giese, Eds. Springer, 2017, pp. 223–248.

[9] R. Calinescu and M. Kwiatkowska, "CADS*: Computer-aided development of self-* systems," in *FASE'09*, 2009, pp. 421–424.

[10] D. Weyns and M. U. Iftikhar, "Model-based simulation at runtime for self-adaptive systems," in *ICAC'16*, 2016, pp. 364–373.

[11] N. R. D'Ippolito, V. Braberman, N. Piterman, and S. Uchitel, "Synthesis of live behaviour models," in *FSE'10*, 2010, pp. 77–86.

[12] E. M. Fredericks, B. DeVries, and B. H. C. Cheng, "Towards run-time adaptation of test cases for self-adaptive systems in the face of uncertainty," in *SEAMS'14*, 2014, pp. 17–26.

[13] R. Bloomfield and P. Bishop, "Safety and assurance cases: Past, present and possible future," in *Making Systems Safer*. Springer, 2010, pp. 51–67.

[14] J. Spriggs, *GSN – The Goal Structuring Notation. A Structured Approach to Presenting Arguments*. Springer, 2012.

[15] D. Weyns, N. Bencomo, R. Calinescu, J. Camara, C. Ghezzi, V. Grassi, L. Grunske, P. Inverardi, J.-M. Jezequel, S. Malek, R. Mirandola, M. Mori, and G. Tamburrelli, "Perpetual assurances for self-adaptive systems," in *Software Engineering for Self-Adaptive Systems III. Assurances*, R. de Lemos, D. Garlan, C. Ghezzi, and H. Giese, Eds. Springer, 2017, pp. 31–63.

[16] E. Denney, G. Pai, and I. Habli, "Dynamic safety cases for through-life safety assurance," in *ICSE'15*, 2015, pp. 587–590.

[17] G. Behrmann, A. David, K. G. Larsen, J. Hakansson, P. Petterson, W. Yi, and M. Hendriks, "UPPAAL 4.0," in *QEST'06*, 2006, pp. 125–126.

[18] R. Calinescu, C. Ghezzi, M. Kwiatkowska, and R. Mirandola, "Self-adaptive software needs quantitative verification at runtime," *Communications of the ACM*, vol. 55, no. 9, pp. 69–77, September 2012.

[19] R. Calinescu, M. Ceska, S. Gerasimou, M. Kwiatkowska, and N. Paoletti, "Efficient synthesis of robust models for stochastic systems," *Journal of Systems and Software*, vol. 143, pp. 140–158, 2018.

[20] S. Gerasimou, R. Calinescu, and G. Tamburrelli, "Synthesis of probabilistic models for quality-of-service software engineering," *Automated Software Engineering*, vol. 25, no. 4, pp. 785–831, 2018.