ENTRUST: Engineering Trustworthy Self-Adaptive Software with Dynamic Assurance Cases

Radu Calinescu
Department of Computer Science
University of York, UK
radu.calinescu@york.ac.uk

Danny Weyns
Department of Computer Science
Katholieke Universiteit Leuven, Belgium
danny.weyns@kuleuven.be

Simos Gerasimou
Department of Computer Science
University of York, UK
simos.gerasimou@york.ac.uk

M. Usman Iftikhar
Department of Computer Science
Linnaeus University, Sweden
usman.iftikhar@lnu.se

Ibrahim Habli
Department of Computer Science
University of York, UK
ibrahim.habli@york.ac.uk

Tim Kelly
Department of Computer Science
University of York, UK
tim.kelly@york.ac.uk

ACM Reference Format:
Radu Calinescu, Danny Weyns, Simos Gerasimou, M. Usman Iftikhar, Ibrahim Habli, and Tim Kelly. 2018. ENTRUST: Engineering Trustworthy Self-Adaptive Software with Dynamic Assurance Cases. In *Proceedings of ICSE '18: 40th International Conference on Software Engineering , Gothenburg, Sweden, May 27-June 3, 2018 (ICSE '18),* 1 pages.
https://doi.org/10.1145/3180155.3182540

Software systems are increasingly expected to cope with variable workloads, component failures and other uncertainties through self-adaptation. As such, self-adaptive software has been the subject of intense research over the past decade [3, 4, 9, 10].

Our work, initially presented in [2], focuses on the use of self-adaptive software in applications with strict functional and non-functional requirements. These applications need compelling assurances that the software continues to meet its requirements while it reconfigures its architecture and parameters at runtime. To address this need, we propose an end-to-end methodology for the ENgineering of TRUstworthy Self-adaptive sofTware (ENTRUST) [2].

Unlike previous research, which is limited to the provision of assurance evidence that individual aspects of the self-adaptive software (e.g. the reconfiguration decisions or their execution) are correct, ENTRUST integrates design-time and runtime modelling and verification of all these aspects with industry-adopted assurance processes. Accordingly, ENTRUST supports the development of both *trustworthy self-adaptive software* and *assurance cases*[1] that confirm the suitability of the software for its intended application.

ENTRUST comprises a combination of design-time and runtime methods for engineering all software and assurance elements of a self-adaptive software system. Design-time modelling, verification and synthesis methods are used for the software and assurance case elements developed before the system is deployed. These methods support the enactment of the self-adaptive software and its controller, and the generation of a partial assurance case. The runtime reconfiguration of the software requires further modelling and verification, and the synthesis of additional assurance evidence to complete the *dynamic* (i.e. continually evolving) *assurance case* [5]. These activities are performed by the runtime ENTRUST methods.

ENTRUST is flexible about the modelling, verification and assurance-evidence synthesis methods used in its design-time and runtime activities. In particular, ENTRUST assurance cases can be based on assurance evidence produced by a combination of testing, simulation and formal verification, at both design time and runtime.

To assess the effectiveness of our methodology, we devised a tool-supported instance of ENTRUST [2] that leverages the established model checkers UPPAAL [1] and PRISM [7] for the synthesis of assurance evidence, and the Goal Structuring Notation standard [6] for assurance case development. This ENTRUST instance was successfully used to develop proof-of-concept self-adaptive software for embedded and service-based systems from the oceanic monitoring and e-finance domains, respectively, and to generate dynamic assurance cases for these systems. The code, models and assurance cases from our case studies are freely available on the project website https://www-users.cs.york.ac.uk/simos/ENTRUST/.

[1] An assurance case is "*a collection of auditable claims, arguments, and evidence created to support the contention that a defined system/service will satisfy the particular requirements.*" [8]

ICSE '18, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5638-1/18/05.
https://doi.org/10.1145/3180155.3182540

REFERENCES

[1] G. Behrmann, A. David, K.G. Larsen, J. Hakansson, P. Petterson, W. Yi, and M. Hendriks. 2006. UPPAAL 4.0. In *QEST'06*. 125–126.
[2] R. Calinescu, D. Weyns, S. Gerasimou, M. U. Iftikhar, I. Habli, and T. Kelly. 2017. Engineering Trustworthy Self-Adaptive Software with Dynamic Assurance Cases. *IEEE Transactions on Software Engineering* PP, 99 (2017), 1–1.
[3] R. de Lemos et al. 2013. Software Engineering for Self-Adaptive Systems: A Second Research Roadmap. In *Software Engineering for Self-Adaptive Systems II*. Springer, 1–32. https://doi.org/10.1007/978-3-642-35813-5_1
[4] R. de Lemos et al. 2017. Software Engineering for Self-adaptive Systems: Research Challenges in the Provision of Assurances. In *Software Engineering for Self-Adaptive Systems III (LNCS)*, Vol. 9640. Springer, 3–30.
[5] E. Denney, G. Pai, and I. Habli. 2015. Dynamic Safety Cases for Through-Life Safety Assurance. In *ICSE'15*. 587–590. https://doi.org/10.1109/ICSE.2015.199
[6] GSN Working Group Online. 2011. Goal Structuring Notation Standard, Version 1. (November 2011).
[7] M. Kwiatkowska, G. Norman, and D. Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *CAV'11*. Springer, 585–591.
[8] Object Management Group. 2017. Structured Assurance Case Metamodel (SACM). (2017). http://www.omg.org/spec/SACM
[9] H. Psaier and S. Dustdar. 2011. A survey on self-healing systems: approaches and systems. *Computing* 91, 1 (2011), 43–73.
[10] M. Salehie and L. Tahvildari. 2009. Self-adaptive Software: Landscape and Research Challenges. *ACM Trans. Auton. Adapt. Syst.* 4, 2 (2009), 14:1–14:42.