# Toward integrated analysis & testing of component-based, adaptive robot systems

**Benedikt Eberhardinger, Axel Habermaier, Alwin Hoffmann, Alexander Pöppel, Wolfgang Reif**

**Angaben zur Veröffentlichung / Publication details:**

Eberhardinger, Benedikt, Axel Habermaier, Alwin Hoffmann, Alexander Pöppel, and Wolfgang Reif. 2016. "Toward integrated analysis & testing of component-based, adaptive robot systems." In *2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), 1-3 Aug. 2016, Vienna, Austria*, edited by Franz Wotawa, Yves Le Traon, and Zhenyu Chen, 301–2. Piscataway, NJ: IEEE. https://doi.org/10.1109/qrs-c.2016.45.



# Toward Integrated Analysis & Testing of Component-Based, Adaptive Robot Systems

Benedikt Eberhardinger, Axel Habermaier, Alwin Hoffmann, Alexander Poeppel, and Wolfgang Reif
Institute for Software & Systems Engineering, University of Augsburg, Augsburg, Germany
Email: {eberhardinger, habermaier, hoffmann, poeppel, reif}@isse.de

## I. Status Quo

Recent developments in robotics are heading toward component-based architectures [1] that are able to cope with novel challenges, e.g., increasing autonomy or human-robot-collaboration in Industry 4.0. These applications have in common that the robot system has to adapt to new situations or external disturbances and, thus, has to change or at least to adjust its current task. This development is, among other things, driven by collaborative working environments of robots and humans which pose new challenges for engineering integrated, autonomous robot systems. Especially sensing is an important aspect to safely detect human presence and calculate an according reaction [2]. This includes safe and time-critical reactions to avoid collisions and moreover to evaluate, adapt or even change the current task. Overcoming the separation of working environments and enabling autonomy revises established concepts and techniques that aim at ensuring functional correctness and safety in particular.

This (r)evolution has to be taken into account on many different levels, e.g., the revision of safety standards such as ISO 10218 [3] addresses the ongoing development by updating and renewing regulations. The next iteration of these standards will include further restrictions and safety considerations for human-robot-collaboration. In order to comply with these future standards, scalable, automated, and reliable analysis and testing (A&T) techniques will be needed that are able to cope with autonomous, adaptive robot systems. We therefore propose a systematic, model-based approach for analyzing this class of systems: The approach is fully integrated into a robot control system, thus making it possible to address functional quality goals as well as safety aspects within one model-based A&T approach. Robot software in the loop testing is enabled by the use of the S# modeling and analysis framework [4].

### A. Component-Based Adaptive Robot Systems

Our component-based run-time environment for robotics [5] is able to reorchestrate the composition of its current components instantaneously at run-time, adapting itself to new or modified tasks. This run-time environment is comprised of real-time components for accessing sensors and for controlling actuators. Moreover, computational tasks (e.g., algorithms or continuous behavior) can be specified in a data-flow language [5]. Each computational task, specified as a data-flow net, is executed in a dedicated real-time component, a net executor. Data-flow nets are able to communicate with sensors and actuators as well as with each other using dedicated ports. At run-time, new computational tasks can be submitted; their coordination is specified using synchronization rules which are able to stop running tasks and start newly submitted ones. Hence, this run-time environment can change the composition of active components in order to adapt the overall behavior of the robot system to newly arising situations (cf. Fig. 1).

### B. The S# Modeling and Analysis Framework

The S# modeling and analysis framework uses Deductive Cause Consequence Analysis (DCCA) [6] for fully automated safety analyses with the explicit-state model checker LTSmin [7]. DCCA computes all minimal critical fault sets (the causes) that can cause hazards (the consequences), i.e., situations that result in environmental damage, injuries, or loss of lives. S# provides a domain specific modeling language embedded into the C# programming language and the .Net run-time environment [4], supporting convenient modeling of faults such as sensor or actuator failures. The C# code constituting a model is executed using its normal semantics during model checking; consequently, models are allowed to use most C# language features as well as many standard .Net libraries and tools. Despite the high level of modeling flexibility and expressiveness, S# can still efficiently and automatically assess system safety by conducting DCCAs.

## II. Integrating Component-Based Robotics and S#

The core idea of our approach is to combine the strengths of our component-based robot run-time environment with the model checking-based verification and safety analysis techniques provided by S# in order to obtain an integrated analysis approach for highly safety-critical human-robot systems. As illustrated by Fig. 1, the foundation of this approach is the integration of the actual robot controller software into a S# model, the latter of which describes behavioral and structural aspects of the controlled plant as well as the sensors and actuators available to the controller. This software in the loop model (SithLM) is used for simulation, analysis as well as test suite derivation and execution. The S# parts of the SithLM test-drive the actual controller software using S#'s model checker, enabling systematic, explorative, model-based testing of adaptive, component-based robot systems.

*a) Technical Feasibility:* As S# models are regular .Net programs, they can execute almost any code during simulations and model checking. Therefore, sensors and actuators within

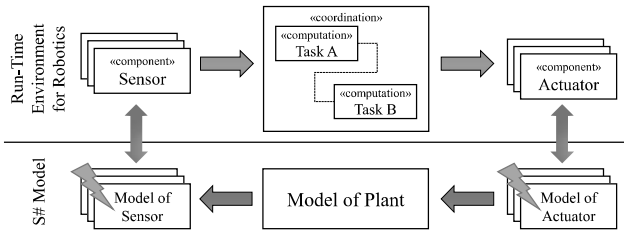


Fig. 1. The system is separated into controller and plant as common in control theory and safety analysis [8]. This forms the feedback cycle which enables the system's overall adaptivity. The upper part shows the controller, i.e., the actual component-based robot control software. The lower part depicts the plant, sensors, and actuators modeled in S#. Abstractions are generally necessary in order to establish the link between the actual control software and the discrete-state S# model. Faults (indicated by flashes) must be integrated into the S# model in order to subsequently allow S# to assess system safety.

the S# models can use standard technologies such as remote method invocation or services to interface with the device drivers of the robot controller, bridging the gap between the different programming languages (C# and C++ in this case), run-time environments, and, if necessary, levels of abstraction. S# subsequently allows the SithLM to be simulated and model checked, executing the modeled sensor, plant, and actuator behaviors as well as the connected robot controller. Abstractions are required in order to reduce large SithLM state spaces to manageable levels and to facilitate model checking. In particular, robot controllers typically have low cycle times in the range of a few milliseconds, requiring significant effort just to analyze a few seconds of robot behavior. It is therefore unlikely that fully exhaustive model checking can be used to analyze these systems; instead, bounded model checking that verifies only short periods of time might have to suffice. In addition to restricting the analyzed amount of time, it might also be necessary to limit the analyzed behavioral variance of the system if no suitable abstractions can be found that would otherwise allow exhaustive model checking. With these restrictions, S# model checking becomes a non-exhaustive form of testing, which, however, still facilitates analyses of both functional correctness as well as safety concerns.

*b) Testing Functional Correctness & Safety:* The integrated approach not only enables testing functional aspects of the systems, but also allows S#'s fully automated DCCA to assess the most safety-critical situations, uncovering potential safety issues in a testing fashion. As it is not always possible to conduct exhaustive DCCAs, appropriate test cases must be selected using appropriate test strategies; we aim at integrating mainly two different strategies: (1) random testing and (2) integration of usage profiles of the humans interacting with system. In particular, the human workflows in industrial working environments are highly standardized and consequently repetitive, leading to usage profiles with low variance. Test cases for the most likely usage scenarios are therefore highly significant while noticeably reducing the number of required test runs. Additionally, the test oracle problem has to be addressed: test runs need to be assessed, i.e., they have to be classified as either correct resp. safe or incorrect resp. unsafe. We see two different solutions: (1) we use constraints to characterize functional properties and safety hazards that can be evaluated automatically to classify test runs at run-time [9], and (2) we introduce mathematical models describing the intended behavior of the robot system, e.g., differential equations describing the robots' dynamics.

## III. Road Map

We outlined how component-based robotics and the component-based analysis techniques of S# can be brought together in order to cope with the new engineering challenges posed by recent developments in industrial and service robotics. This combination leads to a broad research road map toward systematic A&T of adaptive robot systems in times of Industry 4.0. In particular:

1) *Adequate Abstraction.* A key challenge that needs to be addressed is selecting the right level of abstraction for the S# models, especially for adequately describing the behavior of humans that interact with the robots. As outlined above, one approach is to describe their behavior statistically with usage profiles.
2) *Test Case Selection.* A common trait of the systems we investigate is that exhaustive testing is currently not feasible. As a consequence, appropriate testing strategies need to be identified.
3) *Oracle Problem.* The oracle problem is a ubiquitous challenge in testing. Our initial proposal of using constraints and mathematical models must be evaluated and extended into a concept that addresses the special challenges of the robotics application domain.
4) *Hazard Identification & Specification.* Key modeling aspects for safety analysis are the identification and specification of hazards. The norms and standards that are under development partially address these issues, yet each application requires unique care and consideration.
5) *Coping with Adaptivity.* Adaptivity poses new challenges for A&T that also need special attention [9].

Addressing these open research questions is vital to establish safe and adaptive robot systems for industrial applications.

## References

[1] D. Brugali and P. Scandurra, "Component-based robotic engineering (Part I)," *IEEE Robot. & Autom. Mag.*, vol. 16, no. 4, pp. 84–96, 2009.
[2] A. Hoffmann, A. Schierl, A. Angerer, M. Stüben, M. Vistein, and W. Reif, "Robot collision avoidance using an environment model for capacitive sensors," in *Plan., Cont., & Sen. f. Safe HRI, ICRA*, 2015.
[3] International Organization for Standardization, "ISO 10218-1/2:: Robots and robotic devices – Safety requirements for industrial robots," 2011.
[4] A. Habermaier, J. Leupolz, and W. Reif, "Executable Specifications of Safety-Critical Systems with S#," in *DCDS*. IFAC, 2015, pp. 60–65.
[5] M. Vistein, A. Angerer, A. Hoffmann, A. Schierl, and W. Reif, "Flexible and continuous execution of real-time critical robotic tasks," *Intl. J. Mechatronics & Autom.*, vol. 4, no. 1, 2014.
[6] F. Ortmeier, W. Reif, and G. Schellhorn, "Deductive Cause-Consequence Analysis (DCCA)," in *16th IFAC World Congress*. Elsevier, 2006.
[7] G. Kant, A. Laarman, J. Meijer, J. van de Pol, S. Blom, and T. van Dijk, "LTSmin: High-Performance Language-Independent Model Checking," in *TACAS*. Springer, 2015, pp. 692–707.
[8] N. Leveson, *Engineering a Safer World*. MIT Press, 2011.
[9] B. Eberhardinger, H. Seebach, A. Knapp, and W. Reif, "Towards Testing Self-organizing, Adaptive Systems," in *ICTSS*. Springer, 2014.