Joel Greenyer, Malte Lochau, and Thomas Vogel (Hrsg.): Explainable Software for Cyber-Physical Systems (ES4CPS), GI-Dagstuhl Seminar 19023, January 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 1

# Explainable Quality Assurance of Behavioral Requirements

Thomas Vogel[1]

## 1 Context: Safe.Spec

Imprecise, incorrect, and inconsistent specifications of requirements often cause faults in the software and result in increasing costs for developing and maintaining the software. In the *Safe.Spec*[2] project, we develop an integrated tool for the quality assurance of behavioral requirements, that particularly:

1. supports requirements engineers and domain experts in formally modeling behavioral requirements as scenarios expressed by UML sequence diagrams. For this purpose, we provide a catalog of behavior and interaction patterns (e.g., send and reply). A requirements engineer or domain expert instantiate and compose such patterns to create a scenario.
2. automatically composes the modeled scenarios to an overall specification of the requirements that allows us to analyze the interplay between individual requirements. The overall specification is expressed by a network of timed automata for the UPPAAL[3] tool. To achieve the composition, we extend an existing algorithm [UKM03] by adapting it to the technical domains of UML and UPPAAL.
3. supports requirements engineers and domain experts in formalizing properties to be verified on the overall specification of the behavioral requirements. Due to the difficulty and error-proneness of specifying properties in a temporal logic, we use the property specification patterns collected and proposed by Autili et al. [Au15] that allows us to define properties in natural language using a structured English grammar.
4. verifies the defined properties against the overall specification of the requirements using UPPAAL. For this purpose, the Safe.Spec tool automatically translates the properties defined in natural language to Timed Computational Tree Logic (TCTL) and observer automata that serve together with the overall requirements specification as an input to UPPAAL. The translation connects the property specification patterns [Au15] to UPPAAL taking UPPAAL's limited support of TCTL into account.

Users of the Safe.Spec tool iterate over these four steps to specify and verify behavioral requirements. We anticipate that most faults in the specification will be avoided by formally modeling the requirements, in contrast to describing the requirements in natural language,

[1] Humboldt-Universität zu Berlin, Software Engineering, Unter den Linden 6, 10099 Berlin, Germany (thomas.vogel at informatik.hu-berlin.de)

[2] Safe.Spec is sponsored by the German Federal Ministry of Education and Research under Grant No. 01IS16027.

[3] `http://www.uppaal.org/`

while subtle faults will be detected by the verification. Thus, Safe.Spec contributes to the quality assurance of requirements and to the formalization of high-quality requirements.

Focus of the Safe.Spec project are automotive systems that become more and more autonomous and open, and therefore require a high quality, for instance, with respect to safety, starting from the beginning of the development process with the requirements.

## 2 Explainability in Safe.Spec

In the Safe.Spec project, we encounter at least three explainability problems:

1. The scenarios/requirements modeled with UML sequence diagrams should be textually described and explained. On the one hand, a textual description of the requirements might be needed due to legal reasons. On the other hand, it supports domain experts and novice users with little experience of modeling with UML in creating and understanding the scenarios. In the Safe.Spec project, we aim for automatically generating a textual description for each scenario that is derived from the behavior and interaction patterns used for composing the scenario.
2. The verification properties expressed in TCTL and observer automata should be described and explained textually. For this purpose, the structured English grammar for creating the properties is used to generate a description of each property in natural language.
3. The results from the verification performed on a network of timed automata (NTA) should be lifted and explained at the abstraction level of the scenarios (UML sequence diagrams). This is critical since users specify and change the requirements by means of scenarios while the NTA is automatically obtained by composing the scenarios. Thus, users interact with the scenarios and not directly with the NTA. Consequently, a counter example obtained for the NTA should be traced back to the UML sequence diagrams so that users can understand and relate it to the scenarios. This would provide actionable hints on how to change the scenarios to take the verification results into account. In this context, a research question is the selection of a suitable counter example. While UPPAAL provides either a *shortest*, *fastest*, or *some* counter example, a notion of *explainable* counter examples is missing.

To address these explainability problems, at least knowledge about model transformation, traceability, formal methods, and model checking is needed. To obtain reasonable and usable explanations, including domain knowledge seems to be a promising direction.

## References

[Au15] Autili, Marco; Grunske, Lars; Lumpe, Markus; Pelliccione, Patrizio; Tang, Antony: Aligning Qualitative, Real-Time, and Probabilistic Property Specification Patterns Using a Structured English Grammar. IEEE Trans. Softw. Eng., 41(7):620–638, 2015.

[UKM03] Uchitel, Sebastian; Kramer, Jeff; Magee, Jeff: Synthesis of Behavioral Models from Scenarios. IEEE Trans. Softw. Eng., 29(2):99–115, 2003.