



Sam Procter
Senior Architecture Researcher



Keaton Hanna
Associate Software Engineer



Lutz Wrage
Senior Member of
the Technical Staff

Collaborators



Eunsuk Kang, CMU



Ian Dardik, CMU



Yining She, CMU

Formalizing and Automating STPA with Robustness

Automating Hazard Analysis with Formal Methods

Introduction

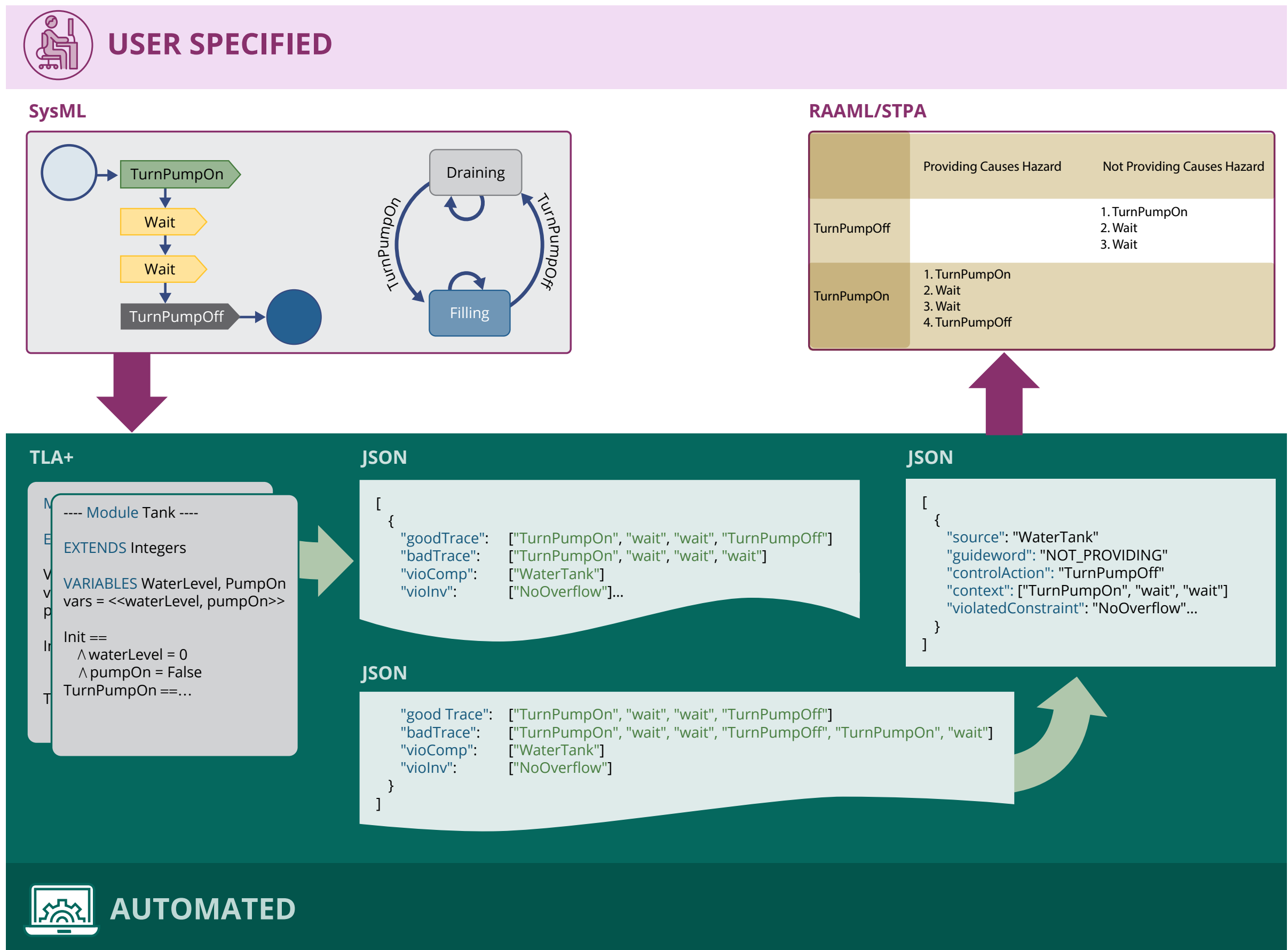
System-Theoretic Process Analysis (STPA) is a socio-technical hazard analysis that the DoD uses to examine systems for safety. We are using a new property of software-intensive systems, called robustness, to automate the identification of unsafe control actions (UCAs), which is a step of STPA.

Provided as a plugin to Eclipse, our tool uses exports from Cameo Enterprise Architecture (CEA) and allows users to model critical aspects of their system in SysML (e.g., state machine and activity diagrams). The tool produces UCAs in RAAML, a standardized format. RAAML includes support for STPA, allowing analysis output to be rendered in a familiar format.

Methods

1. The user models: a **critical portion** of the system they're building, the **expected behavior** of its operator/environment, and one or more **safety properties** that must be upheld.
2. Our tool then **translates** those models to a precise specification in a language called TLA+ and **checks** the behavior of the machine when the operator or environment doesn't behave as expected. If unsafe behaviors are found, the tool **generates** a pair of behavior traces (one that violates the safety property and one that doesn't) and classifies the unsafe behavior using STPA's guidewords.
3. The resulting classification is shown to the user in CEA using RAAML.

We are **automating** the System-Theoretic Process Analysis (STPA). Users model **critical elements** of their system in SysML and our tool identifies unsafe control actions.



Unsafe System Trace Generation

Unsafe traces are generated using formal methods (which are precise techniques for evaluating software behavior) that evaluate *robustness*. Robustness calculations involve relaxing the environment specification—essentially, evaluating what happens if things outside the system don't go as planned—and determining if the system is able to consistently ensure the specified safety property.

Generating Unsafe Control Actions from Unsafe System Traces

Our software compares safe and unsafe traces using an algorithm similar to those used in spell checkers. We classify the changes needed to convert the safe trace to the unsafe one according to the STPA guideword with which that change corresponds. For example, adding a letter to a word is like *providing* an inappropriate action and deleting a letter is like *not* providing a necessary action. That classification, combined with other data from the model, lets us build the UCA table from STPA's third step.

Results

- We'll have a small user study to evaluate the approach later this year.
- Do you use STPA or Cameo Enterprise Architecture? You can try the tool now. For more information and to access the source code, follow the QR code on the right or go to https://cmu-soda.github.io/projects/project_fasr.html

Copyright 2025 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License. Requests for permission for non-licensed uses should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM25-1328

