# Verifying and Monitoring UML Models with Observer Automata

Valentin BESNARD [1], Ciprian TEODOROV [2], Frédéric JOUAULT [1], Matthias BRUN [1], Philippe DHAUSSY [2]
[1] ERIS, ESEO-Tech, Angers, France [2] Lab-STICC UMR CNRS 6285, ENSTA Bretagne, Brest, France

## Context

- Increasing complexity of embedded systems
- Need for V&V at all design stages to test or prove that a system satisfies its expected properties
- **Need for runtime monitoring** to detect safety property violations at runtime

## Problems

Two problems with classical monitoring approaches:

1) **Semantic gap** between monitors model and monitors code

2) **Different languages** are usually used to express monitors and design models

**Can we address these problems with the model interpreter approach?**

## Encoding Properties as UML Observer Automata

Example of a system requirement for a cruise control:

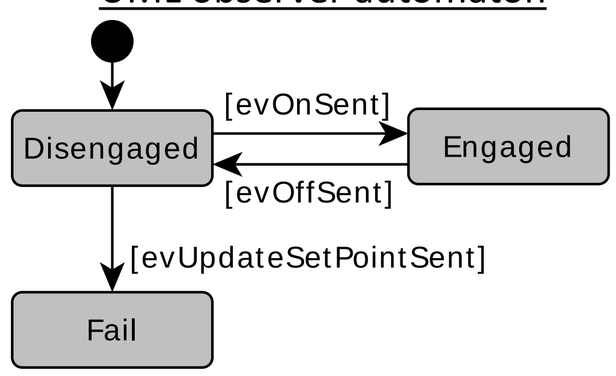
Natural Language

*"After the detection of an event that turns the control loop off and until a contrary event is sent, the system should not try to send new setpoints."*

LTL property

$$[]\ ((|evOffSent|\ and\ !\,|evOnSent|) \rightarrow (!\,|evUpdateSetPointSent|\ W\ |evOnSent|))$$

UML observer automaton



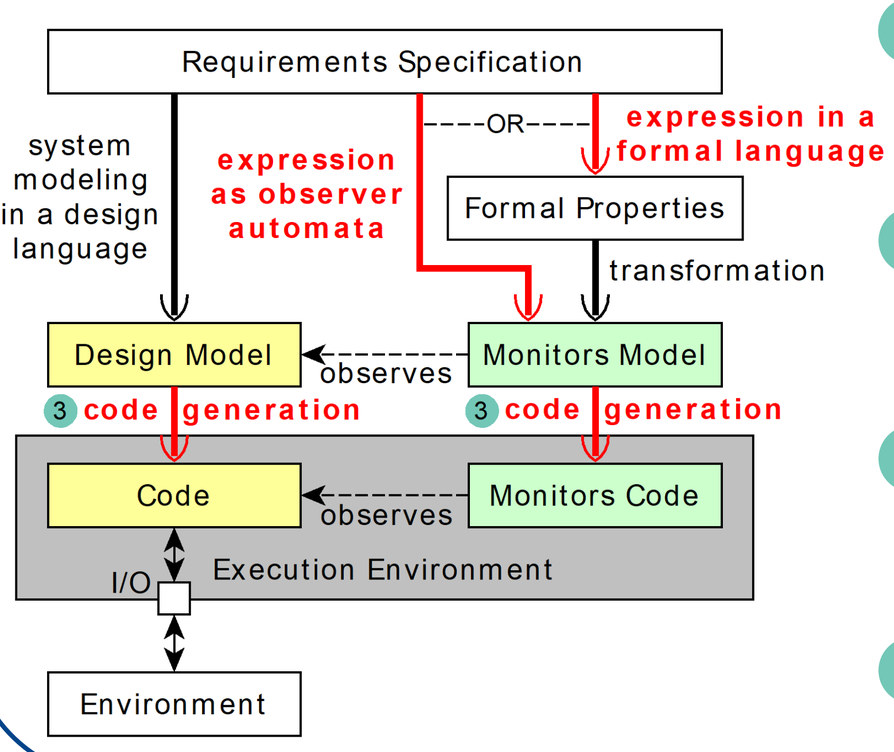

Safety properties encoded into observer automata for finite traces analysis

Expression of observer automata directly in the design language (here UML) as deterministic and complete automata

Expression language extension to read dynamic model execution attributes of the system (e.g., contents of event pools)

## Approach

Classical Monitoring Approaches



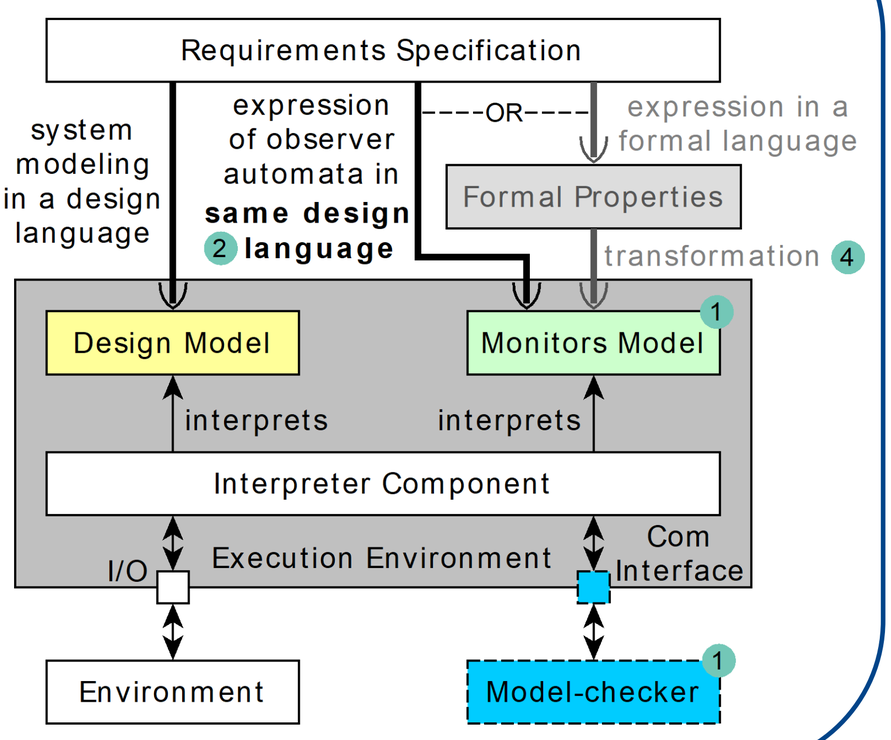

1 Unification of model-checking and monitoring by using the same observer automata in monitors model

2 Simplification of the use of verification tools for engineers without formal background

3 No longer relying on transformations that would require constructing and maintaining correctness proofs [3]

4 Still compatible with approaches that synthesize monitors from formal properties

Our Approach for Monitoring




## Results

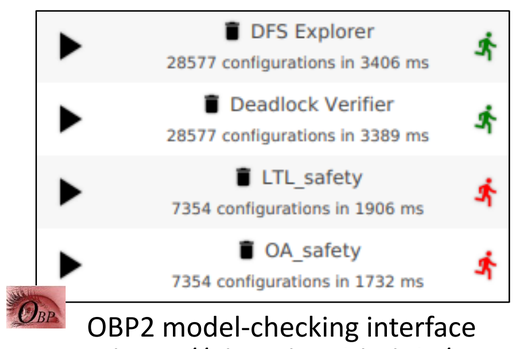
Our Embedded Model Interpreter (EMI) provides capabilities to perform:

- **Runtime monitoring based on UML observer automata** [1]
- **Model-checking based on same observer automata** [1]

This interpreter is still compatible with:

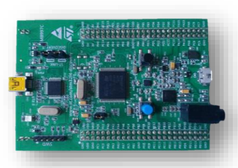
- **Model-checking based on LTL properties** [3]

Overhead of the monitoring infrastructure:
+ 6.5% memory footprint and + 1.2% execution performance




OBP2 model-checking interface
https://plug-obp.github.io/

On desktop computers



On embedded microcontrollers (e.g., STM32) for bare-metal execution

**Perspectives:** Extend expressivity of UML observer automata (e.g., state-event expressions).

## Related Work

- Iulian Ober, Susanne Graf, and Ileana Ober. Validating timed UML models by simulation and verification. International Journal on Software Tools for Technology Transfer, 8(2):128– 145, April 2006.
- Klaus Havelund and Grigore Roşu. Synthesizing Monitors for Safety Properties. In Tools and Algorithms for the Construction and Analysis of Systems, p342–356, Berlin, Heidelberg, 2002.
- Andreas Bauer, Martin Leucker, and Christian Schallhart. Runtime Verification for LTL and TLTL. ACM Transactions on Software Engineering and Methodology, 20(4):14:1–14:64, September 2011.
- Nicolas Halbwachs, Fabienne Lagnier, and Pascal Raymond. Synchronous Observers and the Verification of Reactive Systems. In Algebraic Methodology and Software Technology, p83–96, London, 1994.

**Our publications on the embedded model interpreter approach:**

[1] Verifying and Monitoring UML Models with Observer Automata. MODELS 2019, Munich, Germany, September 2019.
[2] A Model Checkable UML Soccer Player. MDETools 2019, Munich, Germany, September 2019.
[3] Unified LTL Verification and Embedded Execution of UML Models. MODELS 2018, Copenhagen, Denmark, October 2018.
[4] Embedded UML Model Execution to Bridge the Gap Between Design and Runtime. MDE@DeRun 2018, Toulouse, June 2018.
[5] Towards one Model Interpreter for Both Design and Deployment. EXE 2017, Austin, United States, September 2017.

Contact: valentin.besnard@eseo.fr

Partially funded by: DAVIDSON CONSULTING