Software Model Refactoring Driven by Performance Antipattern Detection

Vittorio Cortellessa Daniele Di Pompeo Vincenzo Stoico
Michele Tucci

{vittorio.cortellessa, daniele.dipompeo}@univaq.it
vincenzo.stoico@graduate.univaq.it
tucci@d3s.mff.cuni.cz

Tools for Stochastic Modelling and Evaluation (TOSME)
12 November 2021

Introduction

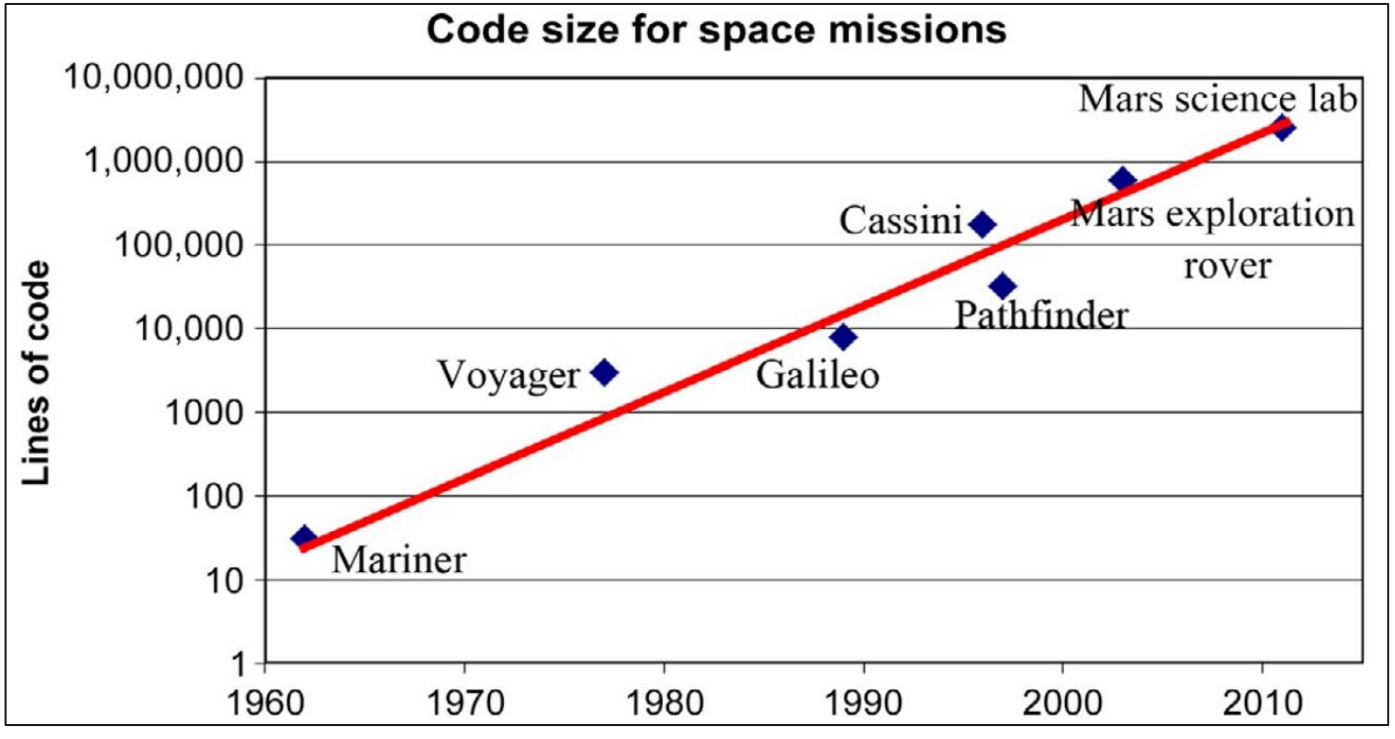
Software complexity is constantly growing

Complexity hampers performance evaluation

Refactoring to face:

➔ New Requirements

➔ Change of Context



PADRE

*P*erformance *A*ntipattern *D*etection and *RE*factoring (PADRE)

- ➔ Detecting performance antipatterns
- ➔ User-driven refactoring
- ➔ Automatic performance estimation

UNIFIED MODELING LANGUAGE™

MARTE

Software Performance AntiPatterns: Common Performance Problems and Their Solutions

Connie U. Smith
Performance Engineering Services
PO Box 2640
Santa Fe, New Mexico, 87504-2640
(505) 988-3811
http://www.perfeng.com/

Lloyd G. Williams
Software Engineering Research
264 Ridgeview Lane
Boulder, Colorado 80302
(303) 938-9847
boulderlgw@aol.com

A pattern is a common solution to a problem that occurs in many different contexts. Patterns capture expert knowledge about "best practices" in software design in a form that allows that knowledge to be reused and applied in the design of many different types of software. Antipatterns are conceptually similar to patterns in that they document recurring solutions to common design problems. They are known as antipatterns because their use (or misuse) produces negative consequences. Antipatterns document common mistakes

Tool Architecture

PADRE exploits *Eclipse Epsilon*

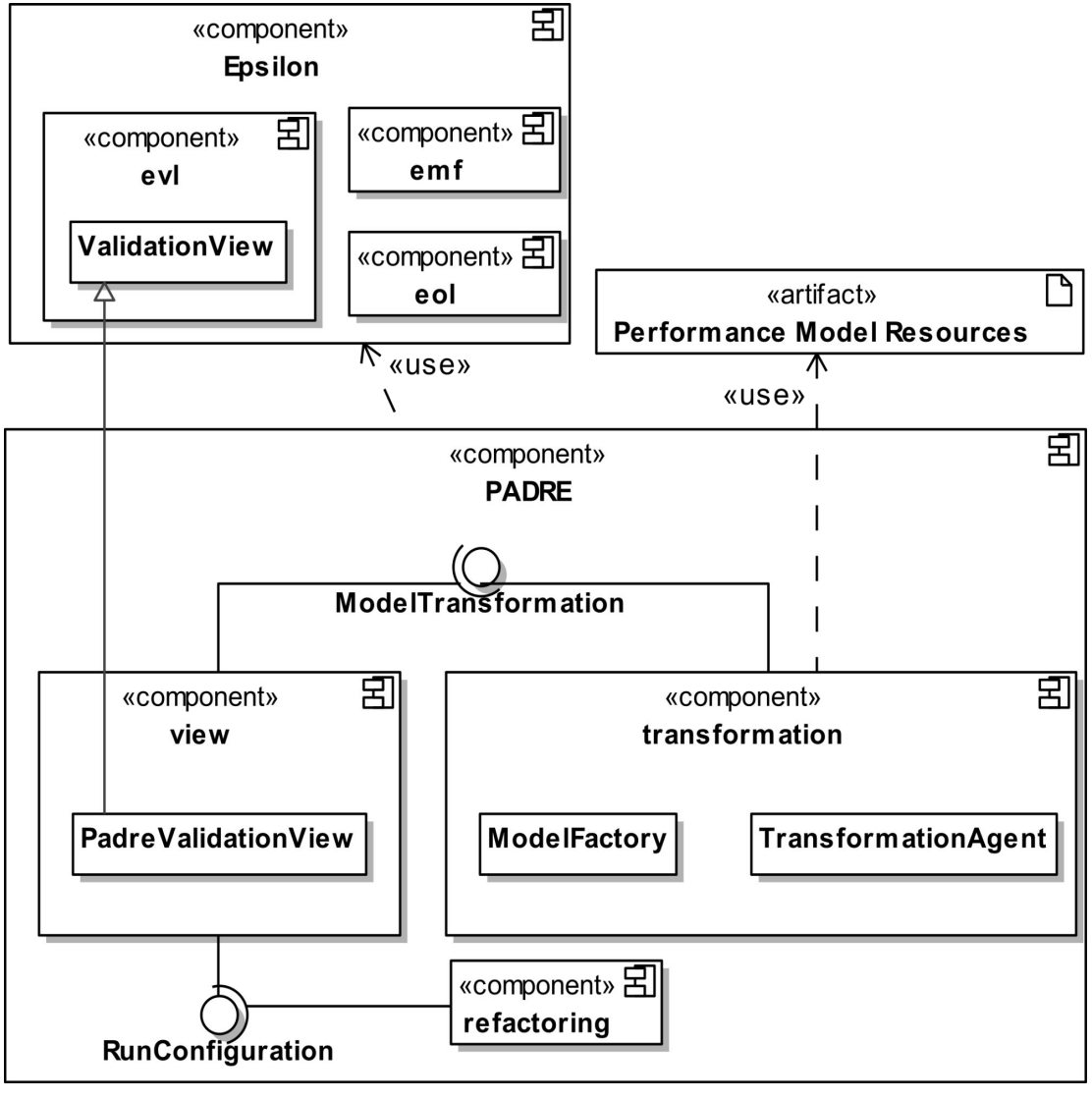
Transformation:

Creates an internal representation of the performance model

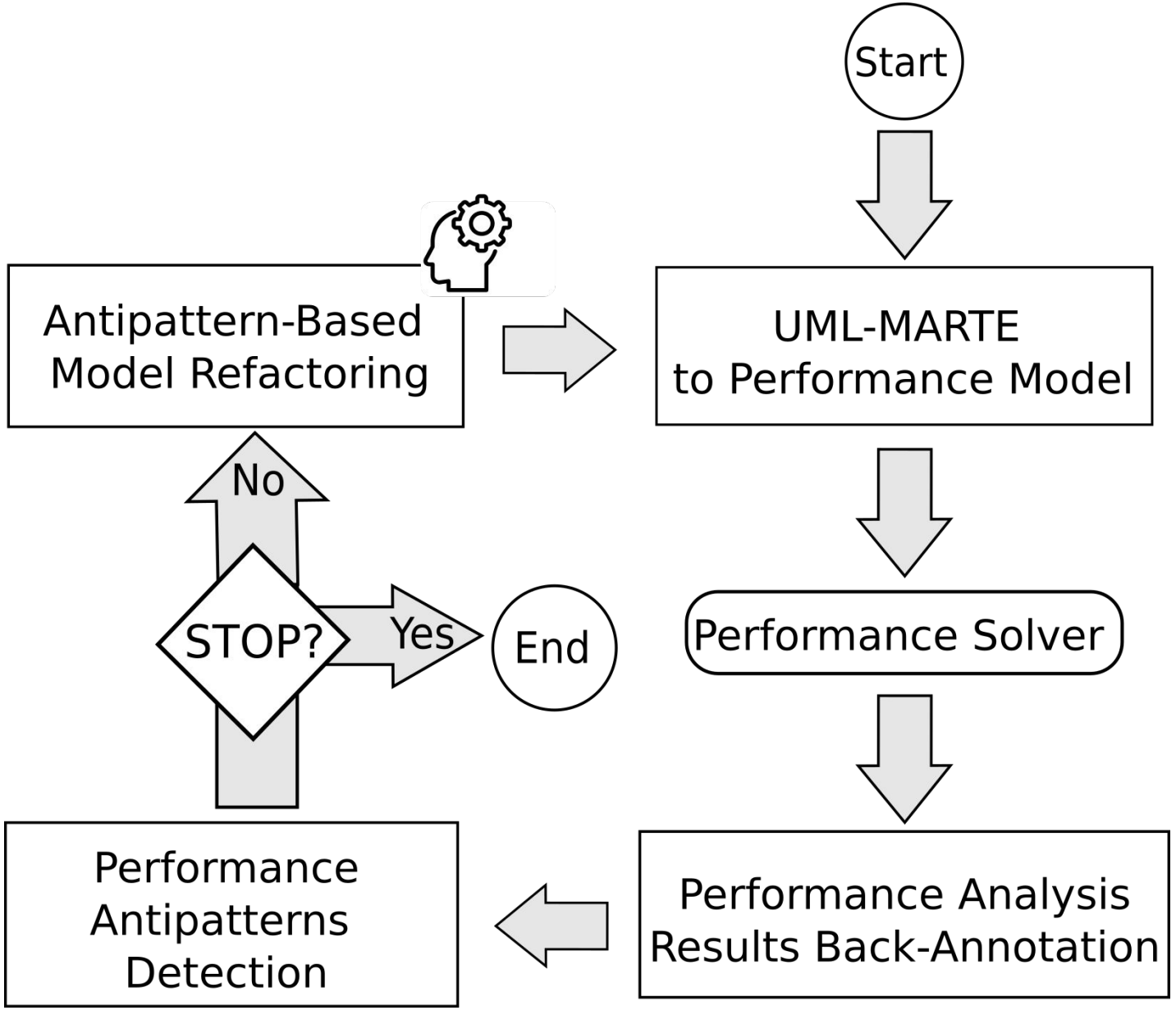
Runs a model-to-model transformation

Refactoring:

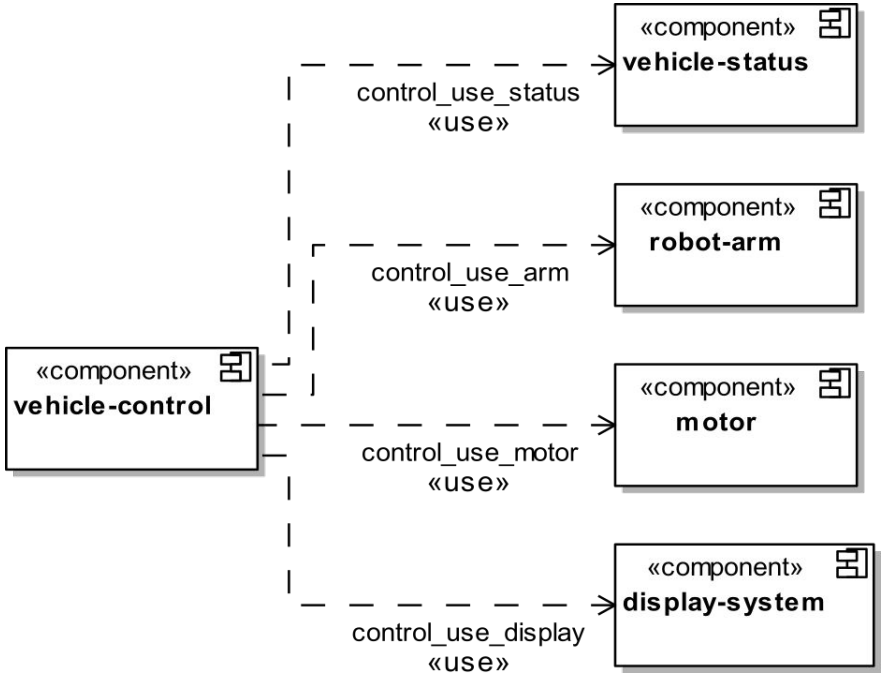
Includes the detection engine and the refactoring actions

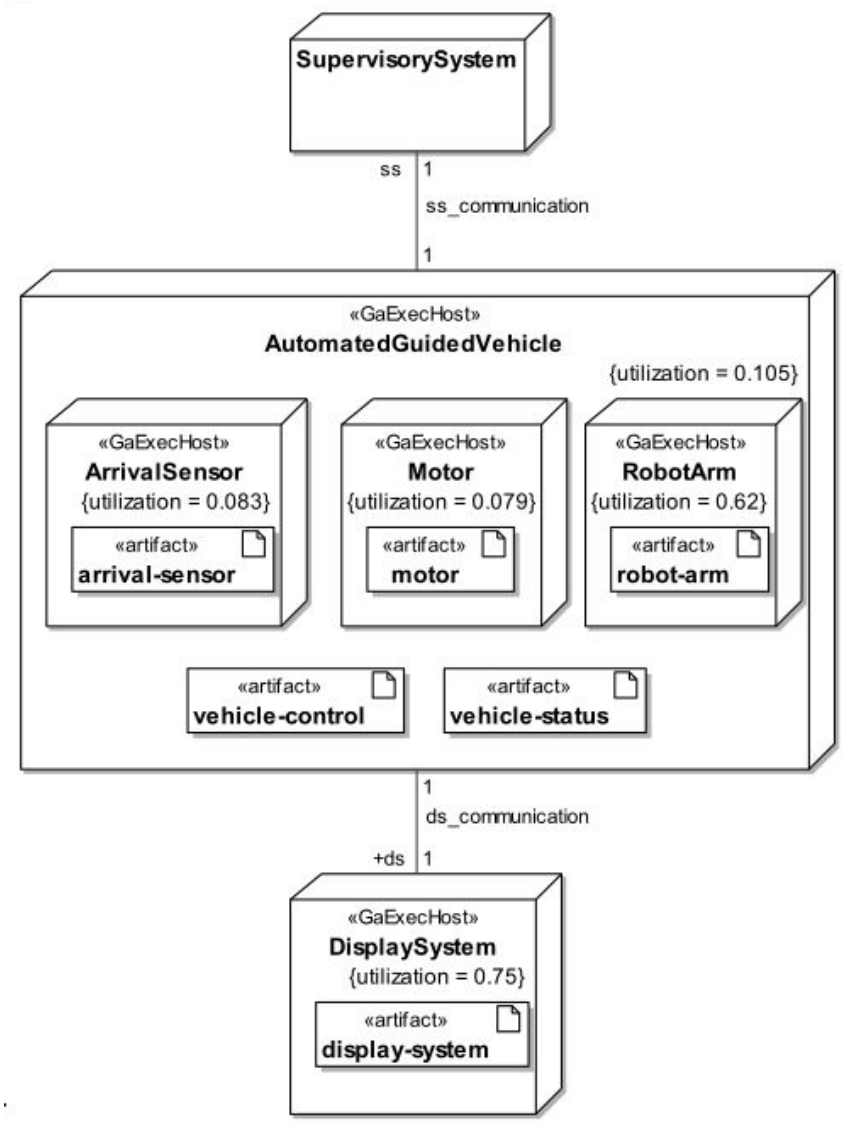


Tool workflow



UML Model Structure

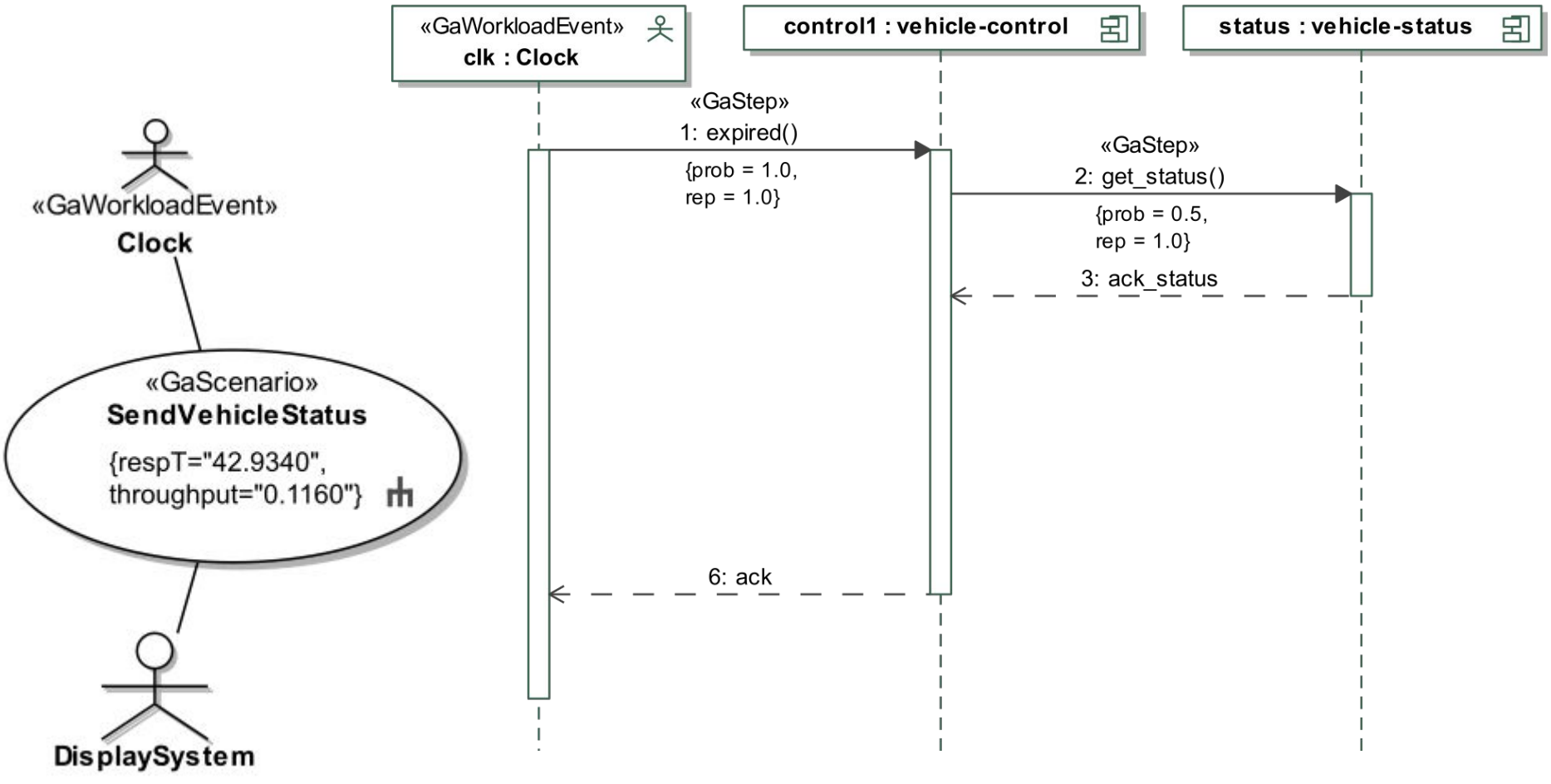




Static View

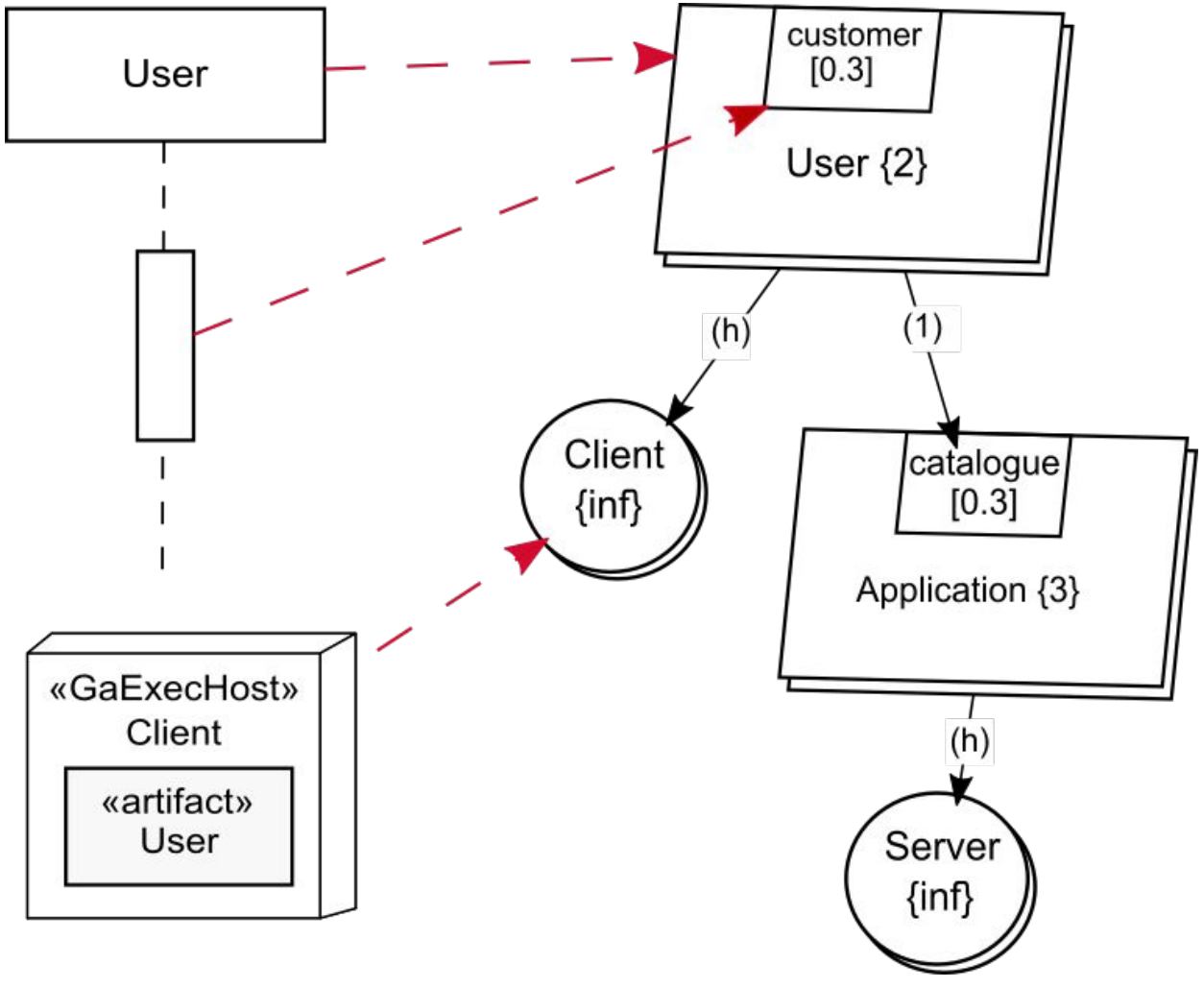
Deployment View

UML Model Structure



Dynamic View

UML2LQN



LQN

Processor

Entry

Task

[0.3] Mean Host Demand

(1) Mean Number of Request

(h) Host

UML

Lifeline

Node

Artifact

Antipatterns detection and model refactoring

UML Metaclass

Performance Antipattern Formula

Refactoring Action

```
context Operation {
    critique PaF {

        check:
            not (self.PaF_F_probExec() and self.PaF_F_resDemand() and
            (self.PaF_F_maxHwUtil() or self.PaF_F_throughput()))

        message: "PaF <Operation>" + self.name

        fix {
            title: "Move it to a new Component deployed to a new Node"
            do {
                self.moveToNewComponentOnNewDevice();
            }
        }

        fix {
            title: "Move it to a new Component deployed to the less" +
            "used neighbour Node"
            do {
                self.moveToNewComponentOnLessUsedNearDevice();
            }
        }
    }
}
```

EVL script

References

1. J. Abella and F.J. Cazorla, *Chapter 9 - Harsh computing in the space domain*, In: Rugged Embedded Systems, pp. 267-293, 2017
2. C.U Smith and L.G. Williams, *Software Performance AntiPatterns; Common Performance Problems and their Solutions*, In: International CMG Conference, 2001
3. Object Management Group, *A UML profile for MARTE: modeling and analysis of real-time embedded systems*, https://www.omg.org/spec/MARTE/ (Accessed 10/11/2021)
4. Eclipse Foundation, *Epsilon: Extensible Platform for Specification of Interoperable Languages for Model Management*, https://www.eclipse.org/epsilon/ (Accessed 10/11/2021)
5. V. Cortellessa, A. Di Marco, and C. Trubiani, *An approach for modeling and detecting software performance antipatterns based on first-order logics*, In: SoSyM, pp. 391–432, 2014
6. G. Franks, T. Al-Omari, M. Woodside, O. Das and S. Derisavi, *Enhanced Modeling and Solution of Layered Queueing Networks*, In: IEEE Transactions on Software Engineering, pp. 148-161, 2009

PADRE DEMO

Thanks!
Any Questions?

Software Model Refactoring Driven by Performance Antipattern Detection

Vittorio Cortellessa Daniele Di Pompeo Vincenzo Stoico
Michele Tucci

{vittorio.cortellessa, daniele.dipompeo}@univaq.it
vincenzo.stoico@graduate.univaq.it
tucci@d3s.mff.cuni.cz

Tools for Stochastic Modelling and Evaluation (TOSME)
12 November 2021