Modeling and Analyzing MAPE-K Feedback Loops for Self-adaptation

Paolo Arcaini

Dept. of Distributed and Dependable Systems, Charles University in Prague, Czech Republic -- arcaini@d3s.mff.cuni.cz

Elvinia Riccobene

Dept. of Computer Science - University of Milan, Italy -- elvinia.riccobene@unimi.it

Patrizia Scandurra

Dept. of Managment, Information and Production Engineering, University of Bergamo, Italy -- patrizia.scandurra@unibg.it

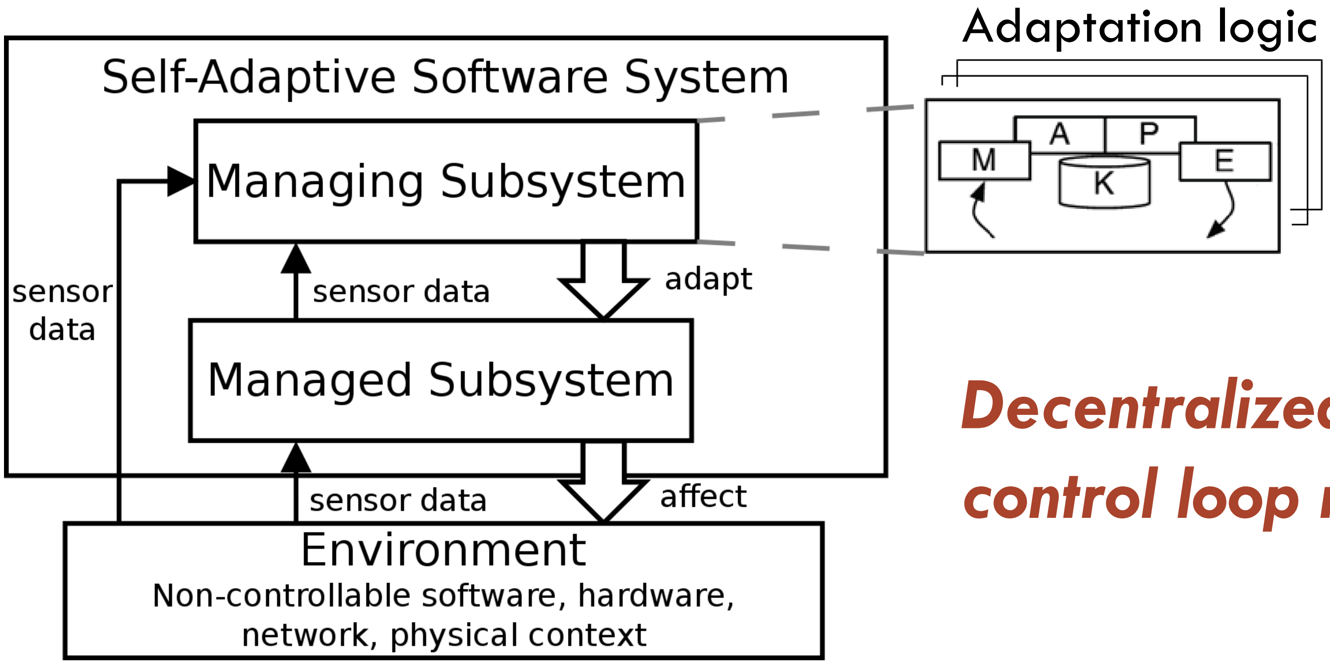
Problem statement

- **Self-Adaptation (SA)** is a promising approach to deal with the complexity, uncertainty and dynamicity of modern software systems
- The **MAPE-K** (**Monitor-Analyze-Plan-Execute** over a shared **Knowledge**) feedback loop is a well-known *control model* for autonomic and self-adaptive systems
- **Formal methods** for specifying and reasoning about self-adaptive systems' behavior **are highly demanded**
  - A study (reference [34] in the paper) shows the number of works that employ formal methods in self-adaptive systems are low
- Our **proposal**:
  - A **formal framework for** modeling, validating, and verifying **self-adaptive systems with multiple interactive MAPE-K loops**
  - based on the formal method ***Abstract State Machines*** and **model-checking** techniques

Outline

- *Decentralized MAPE-K control loops*: reference model for Self-Adaptation
- Background on *Abstract State Machines (ASM)*
- *Self-adaptive ASMs:* enhanced ASM constructs and patterns to model self-adaptive behavior
- Tool-supported formal analysis techniques
- Conclusions and future work

Reference model for Self-Adaptation



Decentralized MAPE-K control loop model

- **MAPE-K (Monitor-Analyse-Plan-Execute components over a shared Knowledge)** : well known architectural solution to realize the control loop of a self-adaptive system
  - J. O. Kephart and D. M. Chess. The vision of autonomic computing. IEEE Computer, 36(1):41-50, 2003
- **Separation of concerns**: a set of interacting MAPE loops, one per each adaptation concern
- **Decentralization**: MAPE computations may be decentralized throughout multiple MAPE loops
  - They need to be coordinated to avoid conflicts!

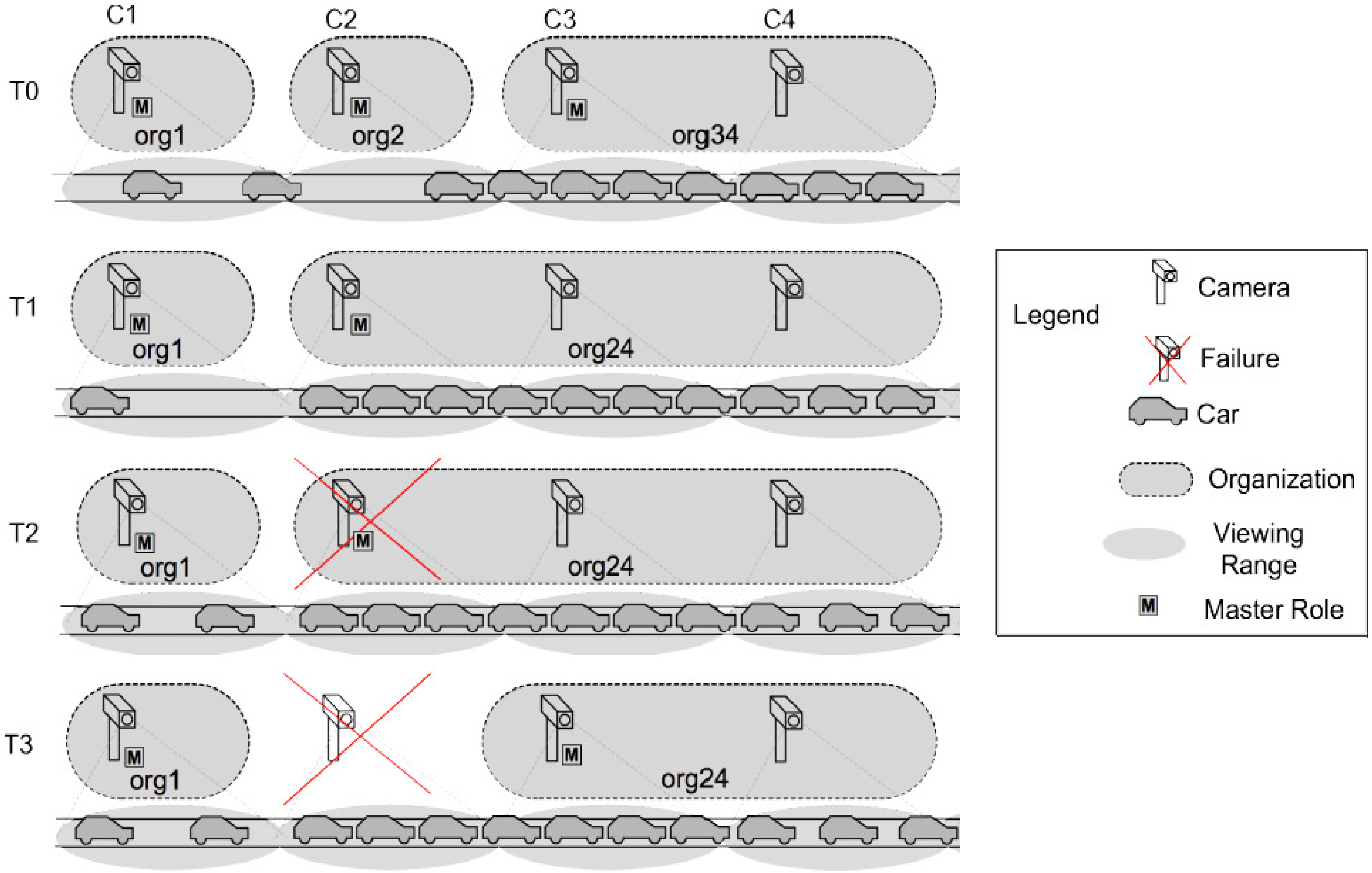
Running case study: Traffic Monitoring application

(inspired by *)

Intelligent cameras collaborate in *master/slaves organizations* to monitor and aggregate useful data whenever the traffic jam enters/leaves their viewing range

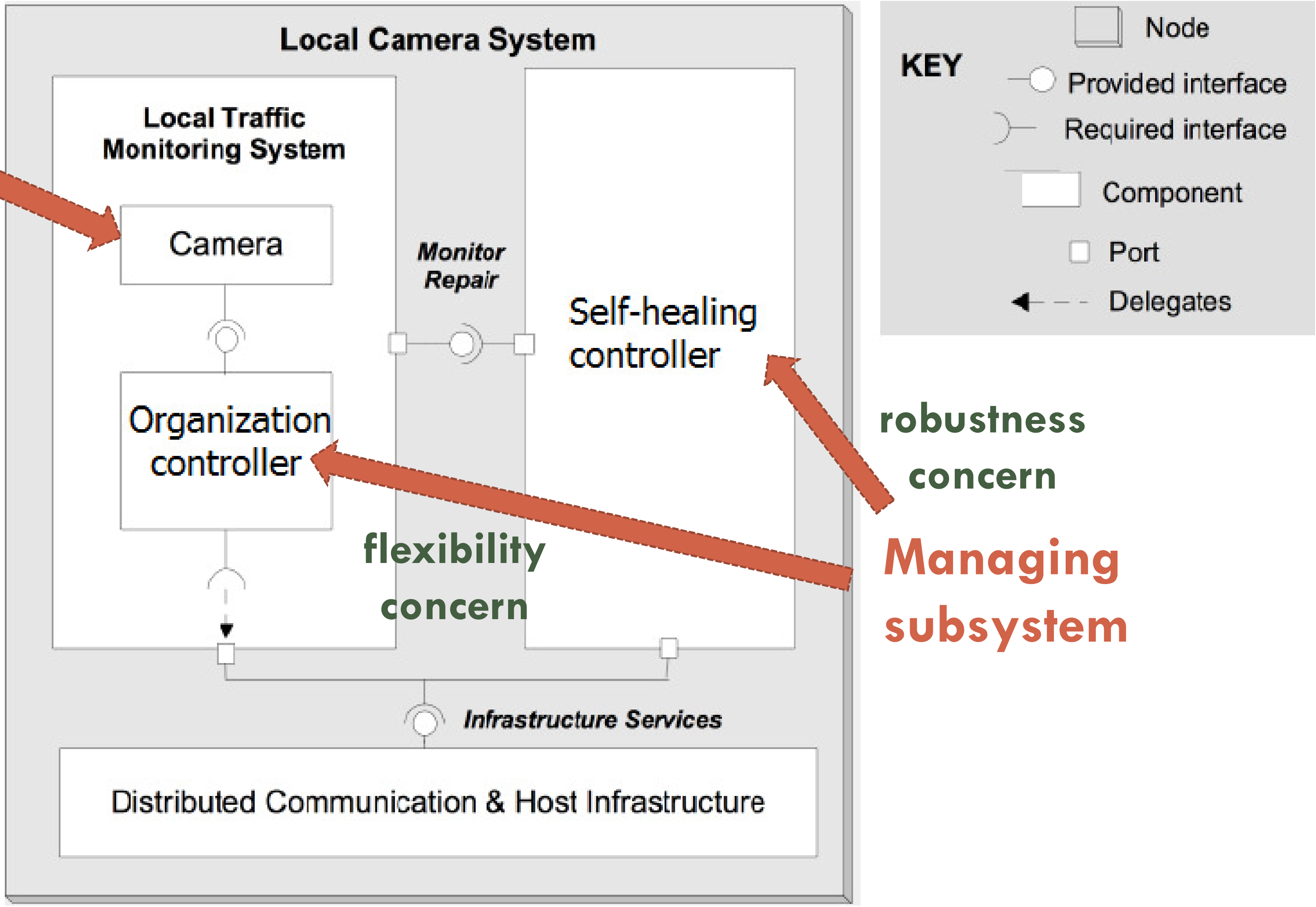
FLEXIBILITY
adaptation
concern

ROBUSTNESS
adaptation
concern



* M. U. Iftikhar and D. Weyns. A case study on formal verification of self-adaptive behaviors in a decentralized system. In FOCLASA 2012, Newcastle, U. K.

Running case study: camera system architecture



Managed subsystem

flexibility concern

robustness concern

Managing subsystem

Abstract State Machines (ASMs)

- **ASMs are an extension of FSMs**
  - states: ***multi-sorted first-order structures***, i.e. <u>domains of objects</u> with <u>functions</u> defined on them
  - transitions: named ***transition rules*** describing how <u>functions change from one state to the next</u>

- **Basic transition rule:** *if Condition **then** Updates*

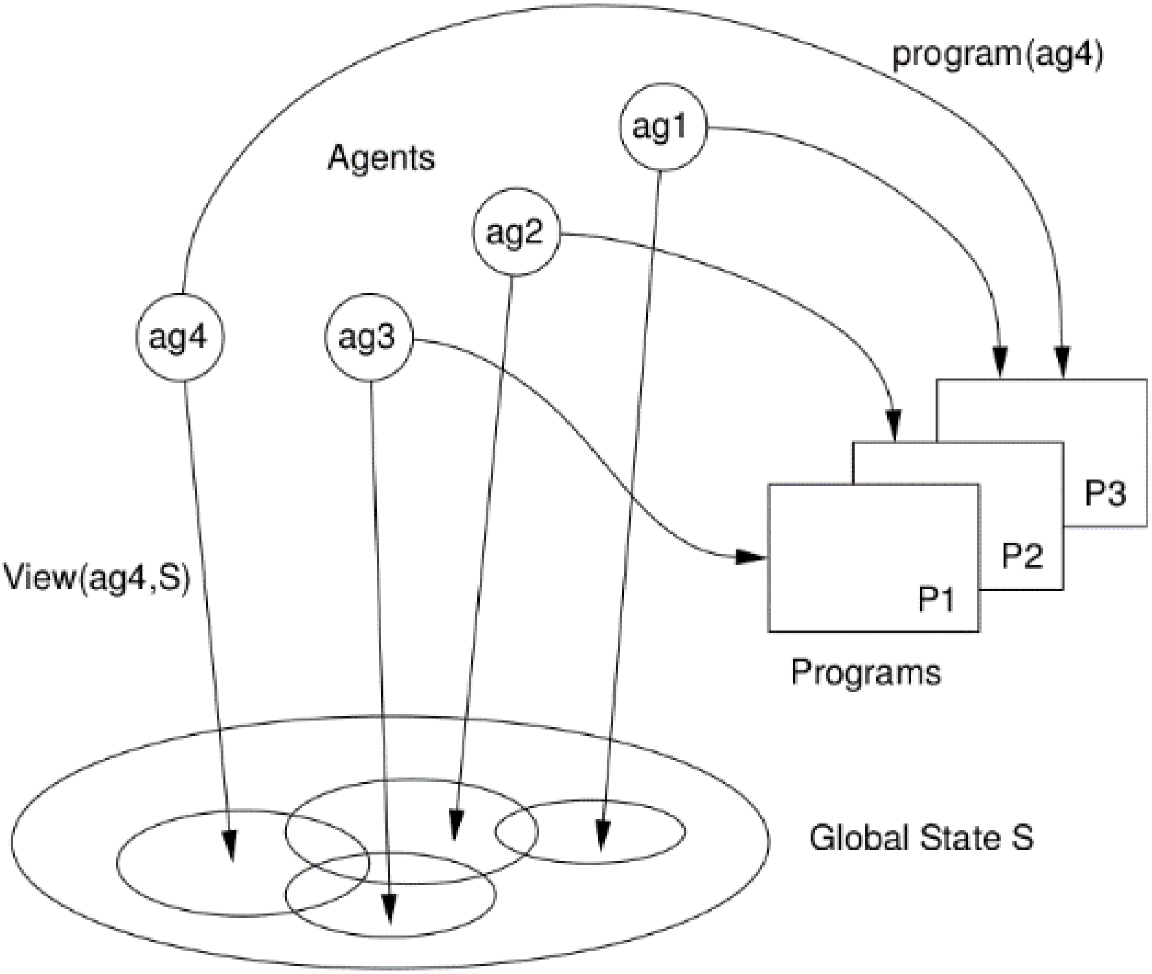
  where *Updates* is a set of **function updates f(t1, …,tn):= t simultaneously executed when *Condition*** is true

- More complex **rule constructors** exist:
  - parallel (**par**) and sequential actions (**seq**)
  - non-determinism (**choose**)
  - unrestricted synch. parallelism (**forall**)
  - etc.

Multi-agent ASM (or distributed ASM)

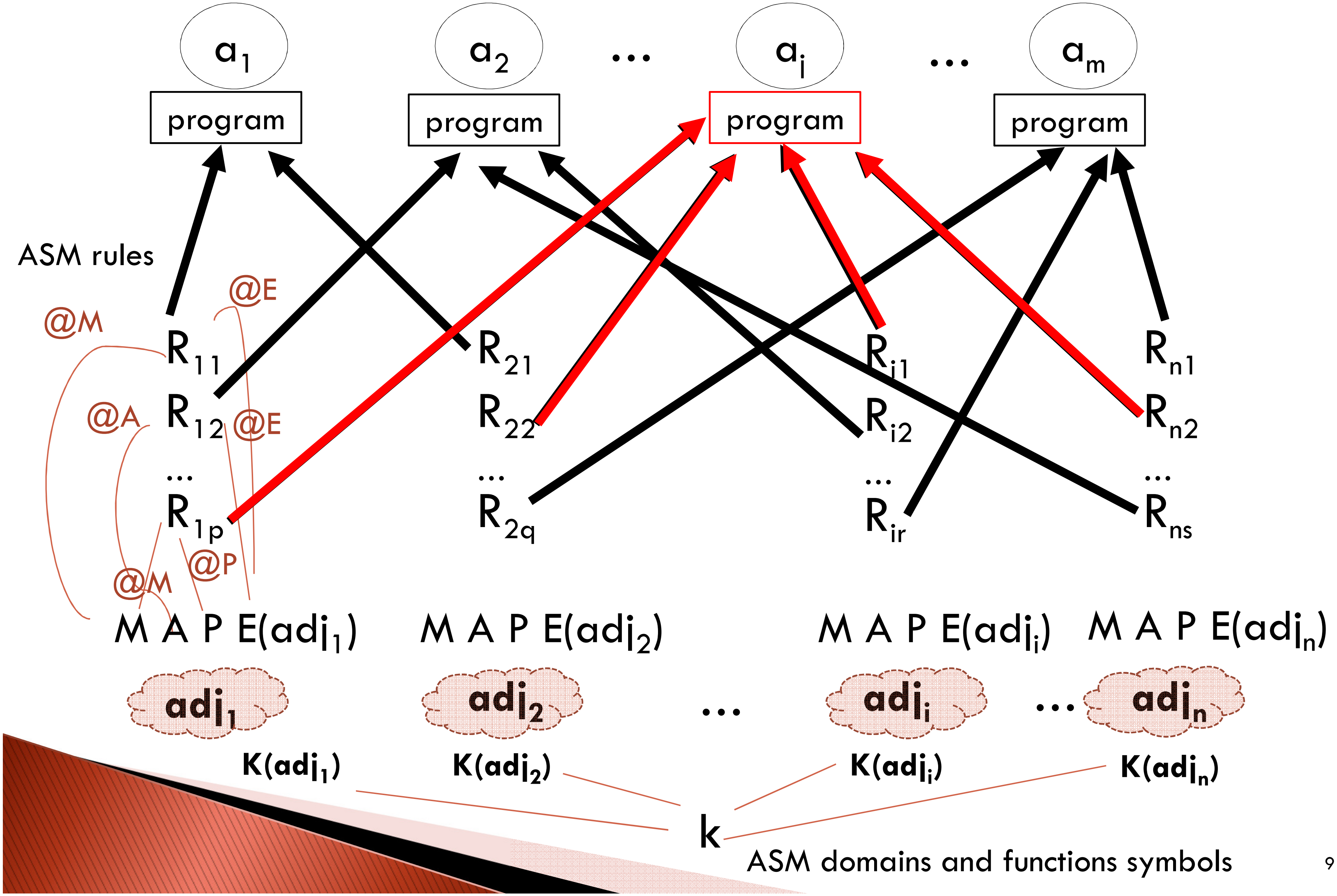
Each **agent** a ∈ **AGENT**

- **has a "local" view** ***View(a, S)*** of the global state S
- **executes its own program *prog(a)*** (i.e., an ASM rule) to determine the next global state



P. Scandurra – SEAMS 2015 May 18-19, 2015, Firenze, Italy

ASM model topology of the managing layer



a_1 a_2 … a_i … a_m

program program program program

ASM rules

@M @E @A @E @M @P

R_{11} R_{12} … R_{1p}

R_{21} R_{22} … R_{2q}

R_{i1} R_{i2} … R_{ir}

R_{n1} R_{n2} … R_{ns}

M A P E(adj_1) M A P E(adj_2) M A P E(adj_i) M A P E(adj_n)

adj_1 adj_2 … adj_i … adj_n

K(adj_1) K(adj_2) K(adj_i) K(adj_n)

k

ASM domains and functions symbols

Self-adaptive ASM

A **multi-agent ASM:**

- **managed agents** $MdA \subseteq Agent$ encapsulate the system's **functional logic**
- **managing agents** $MgA \subseteq Agent$ encapsulate the **adaptation logic of MAPE-K loops**
- A common **knowledge** $K = \bigcup_{adj} K(adj)$ is shared by all managing agents
- The notion of **environment** is represented by **ASM monitored functions**
- A **MAPE loop** for an **adaptation concern** adj_i :

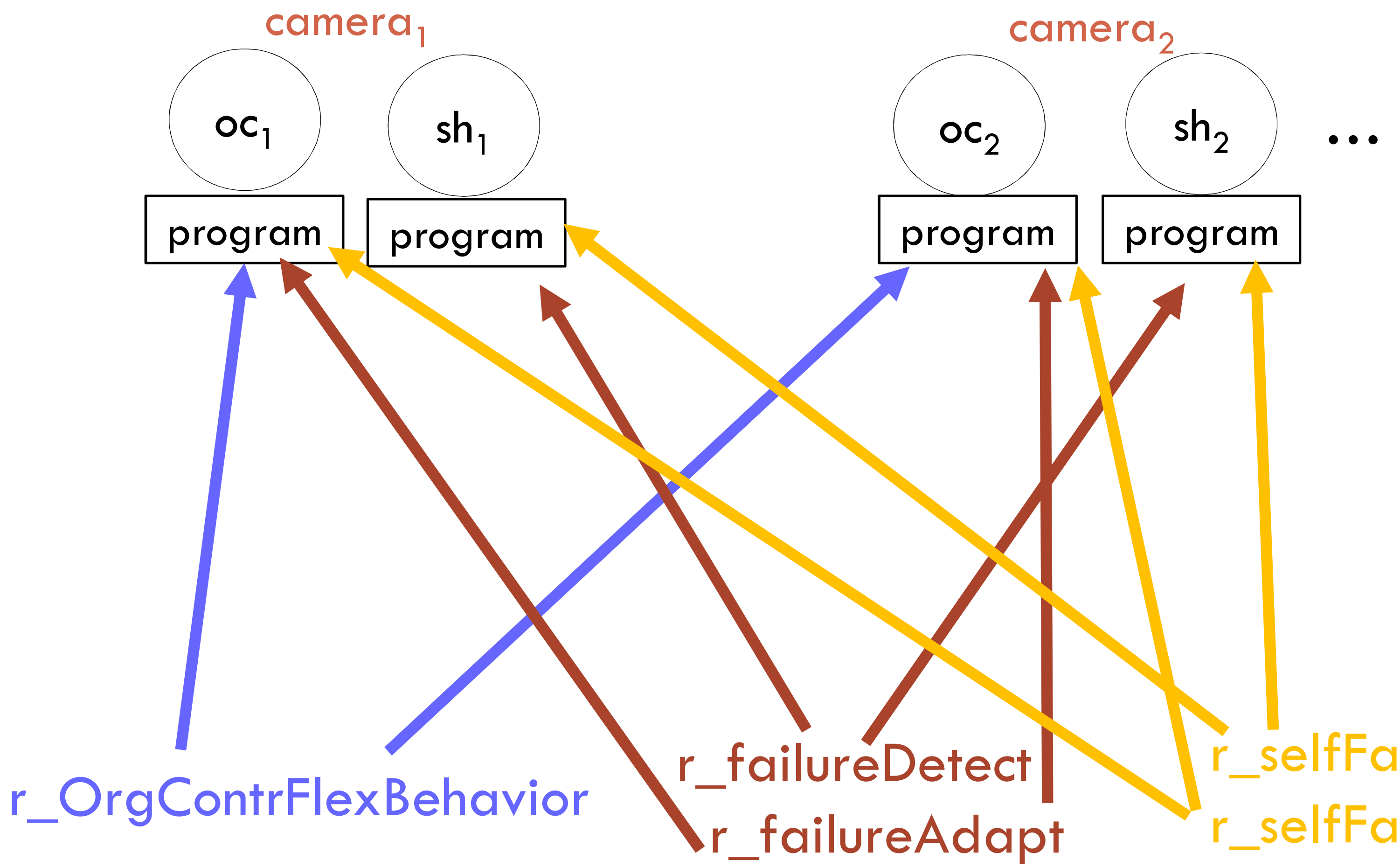
$$MAPE(adj_i) = \{R^{a_1}_{MAPE(adj_i)}, \dots, R^{a_m}_{MAPE(adj_i)}\}$$

  - $\{a_1 \dots, a_m\} \subseteq MgA$ are the managing agents involved in the loop
  - $R^{a_j}_{MAPE(adj_i)}$ is the behavioral contribution of a_j to the loop

- The **program of a managing agent** a_j is the parallel execution of all its behavioral contributions to the loops $j_1, \dots, j_k$ it is invoved to:

$$program(a_j) = \textbf{par}\, R^{a_j}_{MAPE(adj_{j_1})}, \dots, R^{a_j}_{MAPE(adj_{j_k})} \textbf{endpar}$$

ASM model topology of the Traffic monitoring case study



camera$_1$

oc_1 sh_1

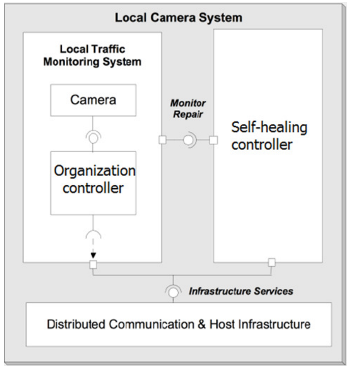
program program

camera$_2$

oc_2 sh_2

program program

…



r_OrgContrFlexBehavior

r_failureDetect

r_failureAdapt

r_selfFailureDetect

r_selfFailureAdapt

MAPE(**flexibility**)

flexibility

MAPE (***extFailure***)

extFailure

MAPE(***intFailure***)

intFailure

k

Traffic monitoring case study

Program of each organization controller

```
macro rule r_organizationController =
    par
        orgContrFlexBehavior(self) //Adaptation due to congestion
        r_failureAdapt[] //Adaptation due to external failure
        r_selfFailureAdapt[] //Adaptation due to internal failure
    endpar

agent OrganizationController : r_organizationController[]
```

Excerpt of rule with MAPE computations

```
macro rule r_selfFailureAdapt =
    par
        if stopCam(camera(self)) then //@M_s
            if state(camera(self)) != FAILED then //@A
                state(camera(self)) := FAILED //@E
            endif
        endif
        if startCam(camera(self)) then //@M_s
            if state(camera(self)) = FAILED then //@A
                par //@E
                    state(camera(self)) := MASTER
                    ...
                endpar
            endif
        endif
    endpar
```

Centralized self-aware monitoring

if $Cond$ **then** $Analyze$ //@M_c[s]

ASM rule schemes or patterns capture the general semantics of MAPE computations

Formal analysis techniques

Supported by the toolset **ASMETA** (ASM mETAmodeling)

- **Model validation**
  - provide early feedback, less demanding than property verification
  - Techniques
    - **Simulation** (interactive simulation, random simulation)
    - **Scenario-based validation**

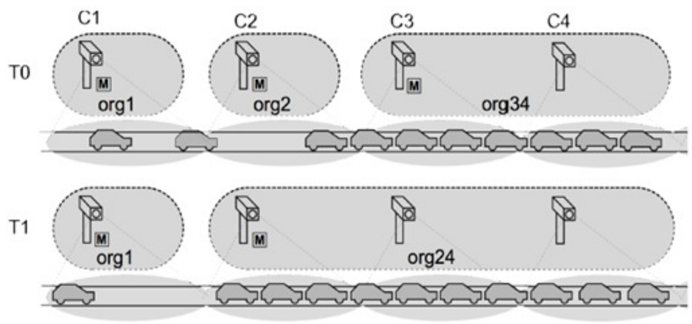
- **Model verification**
  - based on the *model checking* technique
    - **Model review:** verification of *meta-properties* (system-independent properties) defined as CTL formulae
    - **Verification** of **invariants** and **adaptation goals** expressed in CTL/LTL formulas

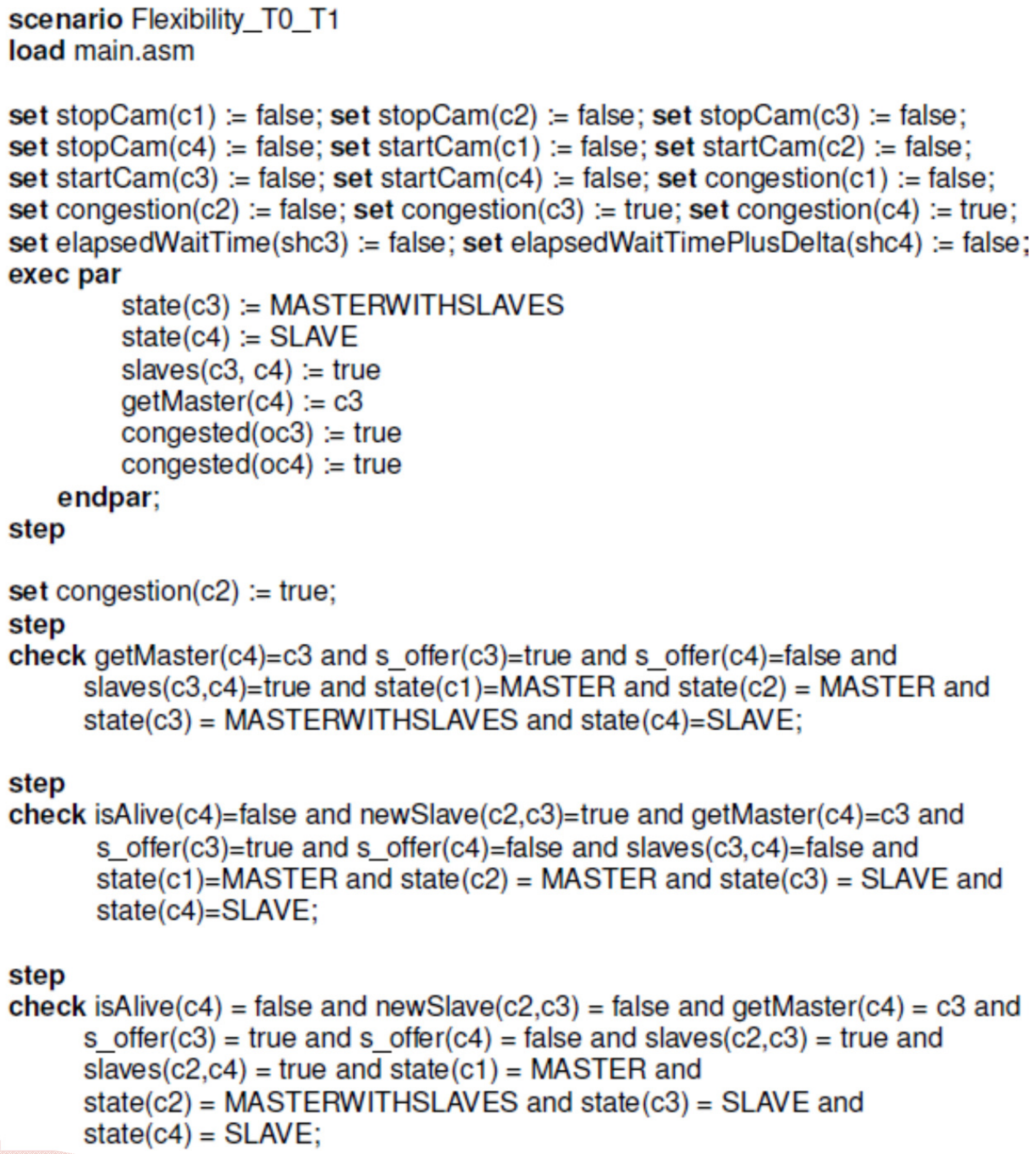
Scenario-based validation

- Definition of **key scenarios specifying the expected behavior** of the model
- Scenarios are written in the language Avalla and
- executed through the validator ASMETA/AsmetaV

Example

Flexibility scenario from T0 to T1in Avalla





```
scenario Flexibility_T0_T1
load main.asm

set stopCam(c1) := false; set stopCam(c2) := false; set stopCam(c3) := false;
set stopCam(c4) := false; set startCam(c1) := false; set startCam(c2) := false;
set startCam(c3) := false; set startCam(c4) := false; set congestion(c1) := false;
set congestion(c2) := false; set congestion(c3) := true; set congestion(c4) := true;
set elapsedWaitTime(shc3) := false; set elapsedWaitTimePlusDelta(shc4) := false;
exec par
        state(c3) := MASTERWITHSLAVES
        state(c4) := SLAVE
        slaves(c3, c4) := true
        getMaster(c4) := c3
        congested(oc3) := true
        congested(oc4) := true
    endpar;
step

set congestion(c2) := true;
step
check getMaster(c4)=c3 and s_offer(c3)=true and s_offer(c4)=false and
      slaves(c3,c4)=true and state(c1)=MASTER and state(c2) = MASTER and
      state(c3) = MASTERWITHSLAVES and state(c4)=SLAVE;

step
check isAlive(c4)=false and newSlave(c2,c3)=true and getMaster(c4)=c3 and
       s_offer(c3)=true and s_offer(c4)=false and slaves(c3,c4)=false and
       state(c1)=MASTER and state(c2) = MASTER and state(c3) = SLAVE and
       state(c4)=SLAVE;

step
check isAlive(c4) = false and newSlave(c2,c3) = false and getMaster(c4) = c3 and
      s_offer(c3) = true and s_offer(c4) = false and slaves(c2,c3) = true and
      slaves(c2,c4) = true and state(c1) = MASTER and
      state(c2) = MASTERWITHSLAVES and state(c3) = SLAVE and
      state(c4) = SLAVE;
```

Model verification

through the ASMETA/AsmetaSMV that translates ASM into models of the model checker NuSMV

- Invariant verification:

I1: ag(not(**forall** \$c **in** Camera **with** state(\$c) = SLAVE))
I2: ag(not(**forall** \$c **in** Camera **with** state(\$c) = MASTERWITHSLAVES))

- Adaptation goals:

Flexibility **F1**: ag((state(c_i) = MASTER and congested(oc_i) and
state(c_{i+1}) = MASTER and congested(oc_{i+1})) implies
af(state(c_i) = MASTERWITHSLAVES and slaves(c_i, c_{i+1})))

Robustness **R1**: ag((stateC(c_i) = FAILED and slaves(c_i, c_{i+1})) implies
ef(not(slaves(c_i, c_{i+1}))))

Model review

- through the AsmetaMA tool (based on AsmetaSMV)
- a meta-property violation may indicate the presence of a real fault or only of a stylistic defect
- Meta-properties categories for SA:
    - ***MPnc: MAPE loops are not in conflict*.** discover unwanted interferences between MAPE-K loops in terms of inconsistent ASM function updates
    - ***MPe : all rules involved in MAPE loops are executed*,** i.e., there is no over specification inside a MAPE loop
    - ***MPm: the knowledge is minimal*,** i.e., it does not contain locations that are unnecessary

Faced challenges

- **Formal modeling self-adaptive behavior** through a clear **separation of concerns** in a **decentralized view**
  - By distinguishing ASM managing agents from managed ones
  - By identifying different adaptation concerns
  - By distributing the MAPE computations of a loop among agents
  - By treating, inside the behavior of a managing agent, different adaptation concerns
  - By distinguishing between decentralized and centralized loop's control through specific ASM rule patterns
- **Formal functional analysis**
  - Validate adaptation requirements by simulation
  - Determine conflicting MAPE loops
  - Assert the system correctness by model checking a set of properties expressing invariants and adaptation goals
  - Check for model completeness without overspecification

Conclusion and future work

- **Self-adaptive ASMs** allowed us **to model and analyze the behavior of self-adaptive systems formally**
  - in terms of MAPE-K control loops executed by ASM agents
- **Validation and verification** techniques allowed us **to ensure the functional correctness of the adaptation logic** by discovering interfering adaptation concerns and goals
- In the future, we want to exploit *runtime monitoring techniques* for runtime verification
- We also want to exploit extensions of ASMs with time models for specifying time-triggered adaptation