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 Management, 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

- **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

- **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!

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

Running case study: camera system architecture

Managed subsystem

Managed subsystem

Robustness concern

Flexibility concern

Managing subsystem

P. Scandurra – SEAMS 2015 May 18-19, 2015, Firenze, Italy
Abstract State Machines (ASMs)

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

**Basic transition rule**:  
*if* Condition *then* Updates  
where Updates is a set of function updates $f(t_1, \ldots, t_n):= t$ 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.
Each agent \( a \in \text{AGENT} \)
- has a “local” view \( \text{View}(a, S) \) of the global state \( S \)
- executes its own program \( \text{prog}(a) \) (i.e., an ASM rule) to determine the next global state.
ASM model topology of the managing layer

ASM rules

\( R_{11} \) \( @M \) \( @E \)
\( R_{12} \) \( @A \) \( @E \)
\( R_{1p} \) \( @M \) \( @P \)
\( R_{21} \) \( @A \) \( @E \)
\( R_{22} \) \( @M \) \( @P \)
\( R_{2q} \) \( @M \) \( @P \)
\( R_{i1} \) \( @A \) \( @E \)
\( R_{i2} \) \( @M \) \( @P \)
\( R_{ir} \) \( @M \) \( @P \)
\( R_{n1} \) \( @A \) \( @E \)
\( R_{n2} \) \( @M \) \( @P \)
\( R_{ns} \) \( @M \) \( @P \)

MAP E(\( \text{adj}_1 \))
\( M \) \( A \) \( P \) \( E(\text{adj}_1) \)
\( M \) \( A \) \( P \) \( E(\text{adj}_2) \)
\( M \) \( A \) \( P \) \( E(\text{adj}_i) \)
\( M \) \( A \) \( P \) \( E(\text{adj}_n) \)

ASM domains and functions symbols

\( \text{adj}_1 \)
\( \text{adj}_2 \)
\( \text{adj}_i \)
\( \text{adj}_n \)

K(\( \text{adj}_1 \))
K(\( \text{adj}_2 \))
K(\( \text{adj}_i \))
K(\( \text{adj}_n \))
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 $adj_i$:
  \[
  MAPE(adj_i) = \{ R_{MAPE(adj_i)}^{a_1}, \ldots, R_{MAPE(adj_i)}^{a_m} \}
  \]
  - $\{a_1, \ldots, a_m\} \subseteq MgA$ are the managing agents involved in the loop
  - $R_{MAPE(adj_i)}^{a_j}$ 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 $j_1, \ldots, j_k$ it is involved to:
  \[
  program(a_j) = \text{par} R_{MAPE(adj_{j_1})}^{a_j} \cdots R_{MAPE(adj_{j_k})}^{a_j} \text{endpar}
  \]
ASM model topology of the Traffic monitoring case study
Traffic monitoring case study

Program of each organization controller

```plaintext
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

```plaintext
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
      endif
      endif
    endif
  endpar
```

Centralized self-aware monitoring

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

P. Scandurra – SEAMS 2015 May 18-19, 2015, Firenze, Italy
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 T1 in Avalla
Model verification

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

- Invariant verification:

  I1: \( \text{ag}(\neg \forall c \in \text{Camera} \ \text{with} \ \text{state}(c) = \text{SLAVE}) \)
  I2: \( \text{ag}(\neg \forall c \in \text{Camera} \ \text{with} \ \text{state}(c) = \text{MASTERWITHSLAVES}) \)

- Adaptation goals:

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

  **Robustness**
  
  R1: \( \text{ag}((\text{state}(c_i) = \text{FAILED} \ \text{and} \ \text{slaves}(c_i, c_{i+1})) \ \text{implies} \ \text{ef}(\neg \text{slaves}(c_i, c_{i+1}))) \)
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

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