

Reflecting on Self-Adaptive Software Systems

Jesper Andersson, Rogerio de Lemos, Sam Malek, *Danny Weyns*

Software Engineering for Adaptive and Self-Managing Software Systems

SEAMS 2009



University of Coimbra



Outline

- I. Motivation**
- II. Reflection**
- III. Reflection prism**
- IV. Challenges**
- V. Conclusions**

Motivation

- **Understand the fundamental underlying concepts of self-adaptive systems**
- **What is the “self” in self-adaptive systems**
- **Explore the crucial role of reflection in self-adaptive systems**
- **Propose topics for future research**

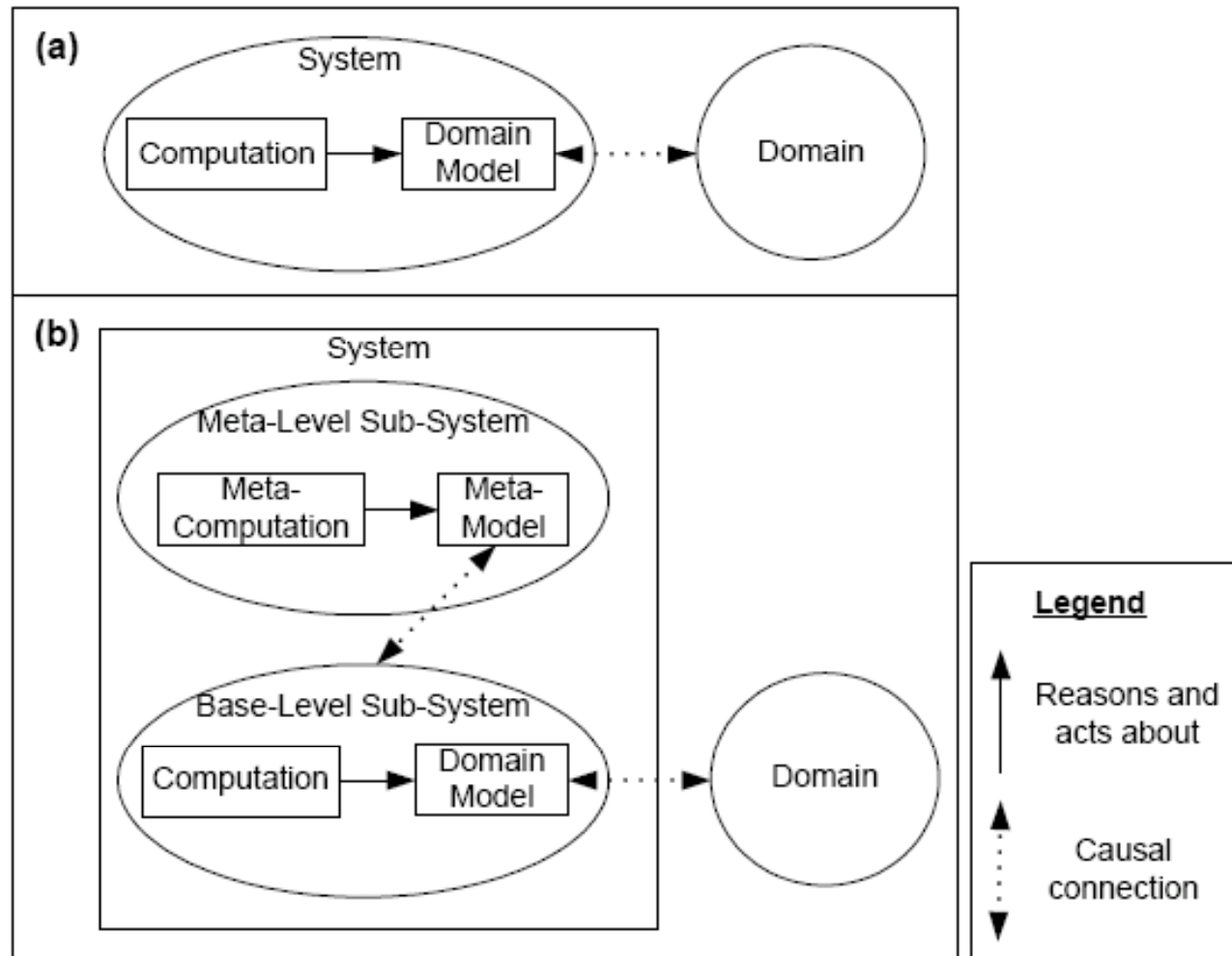
Outline

- I. Motivation
- II. Reflection
- III. Reflection prism
- IV. Challenges
- V. Conclusions

Reflection

- **Reflection is about meta-computation**
 - Computation about computation
 - “The capability of a system to reason about and act upon itself.”
- **Some milestones**
 - 1984 Smith: Reflection in Lisp
 - 1987 Maes: Reflection in OO languages
 - 1999 Cazolla et al. Architectural reflection
 - 2003 Tanter et al. Reflection and AOP
 - 2008 Coulson et al. Reflection and middleware

Reflection



(a) Traditional System

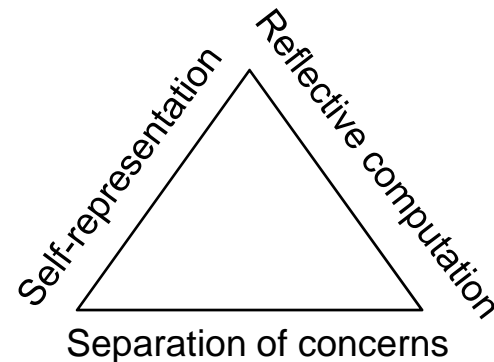
(b) Reflective System

Outline

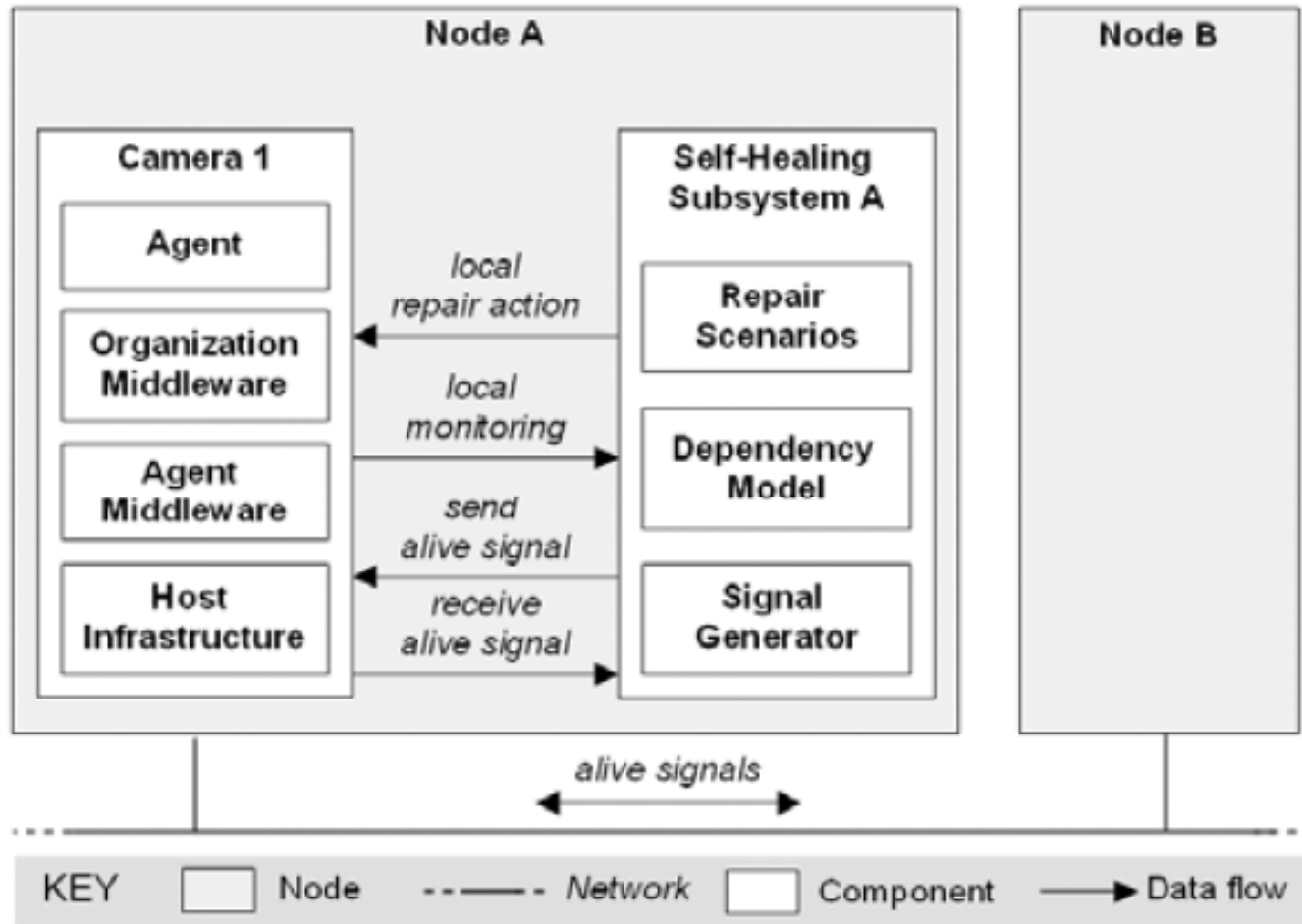
- I. Objective of the paper
- II. Reflection
- III. Reflection prism
- IV. Challenges
- V. Conclusions

Reflective Prism

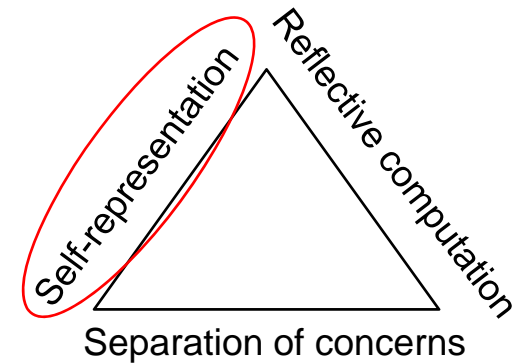
- **Crucial properties of reflection for self-adaptive systems**
- **Basis**
 - Work of Maes, Cazolla, Coulson (among others)
 - Our experience with self-adaptive systems
- **Prism with 3 sides**
 - Self-representation
 - Reflective computation
 - Separation of concerns



Case study: self-healing



Self-representation

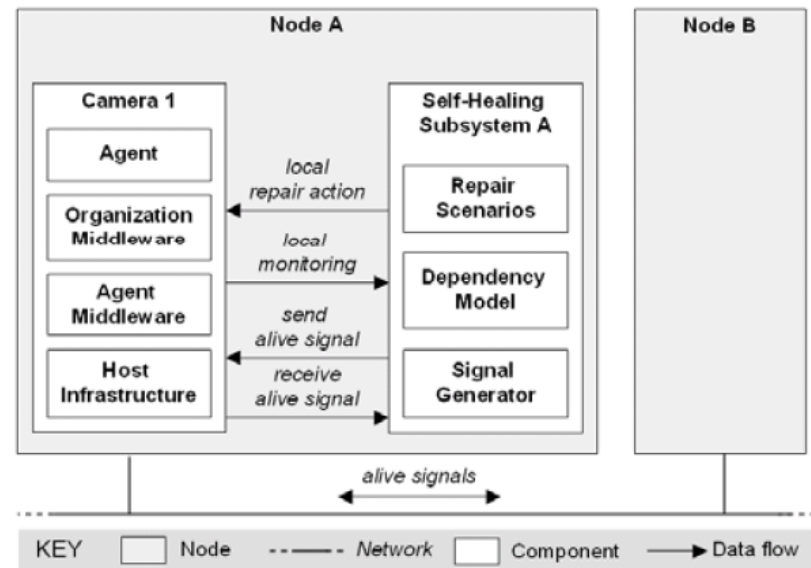


- **Type of representation**

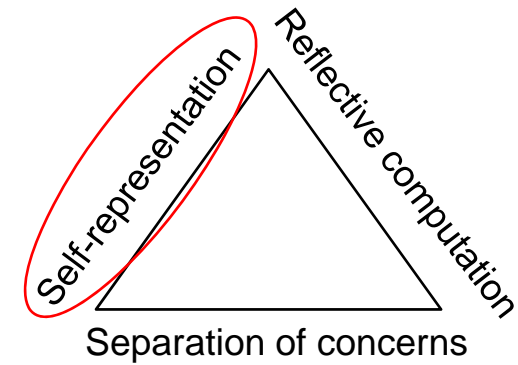
- Procedural = self-representation is part of implementation
- Declarative = self-representation is independent entity

- **Case study**

- Declarative
- Dependency model



Self-representation



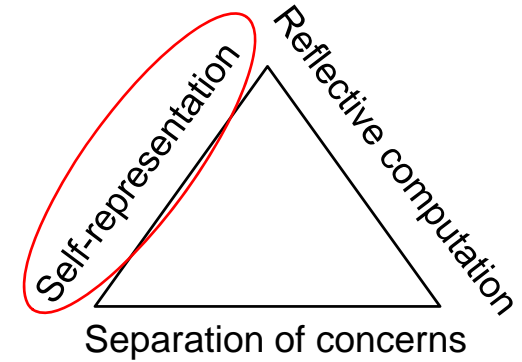
- **Granularity**

- Smallest element that is reified
 - Programming language: class, object, method..
 - Self-adaptive systems typically architectural elements

- **Case study**

- From data types (id's of cameras) to high-level concepts (such as role contracts of agents)

Self-representation



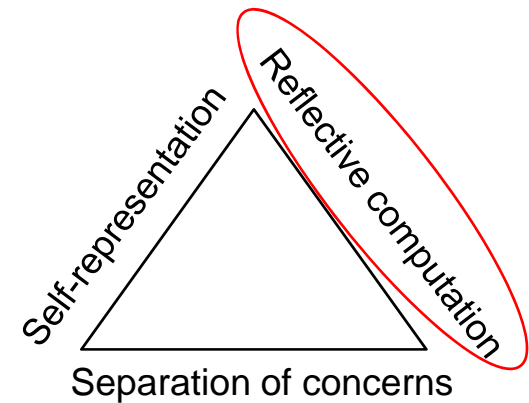
- **Completeness**

- What is reified at meta-level
 - 3-KRS (Maes) complete
 - Self-adaptive frameworks: from partial to complete architecture model

- **Case study**

- Partial: dependencies relevant for self-healing

Reflective computation



- **Type of reflection**

- o Structural

- static structures – classes, method def. etc.

- o Versus behavioral

- dynamic structures – objects' state, messages, etc.

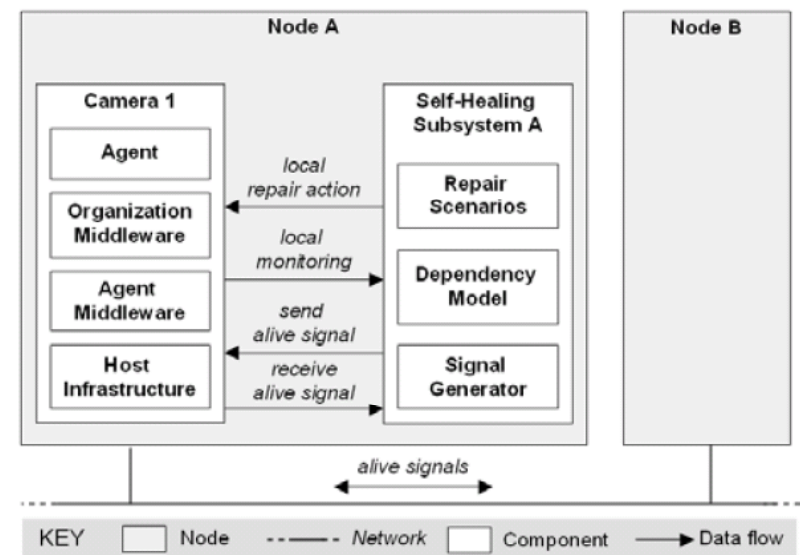
- **Case study**

- o Behavioral aspects

- organizations' state

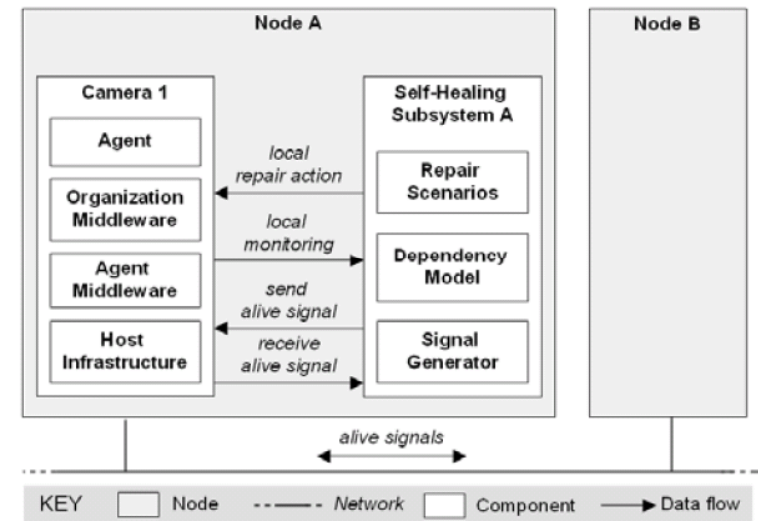
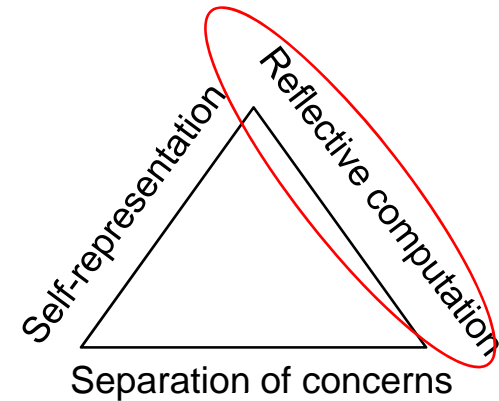
- o Structural aspects

- structure of camera network

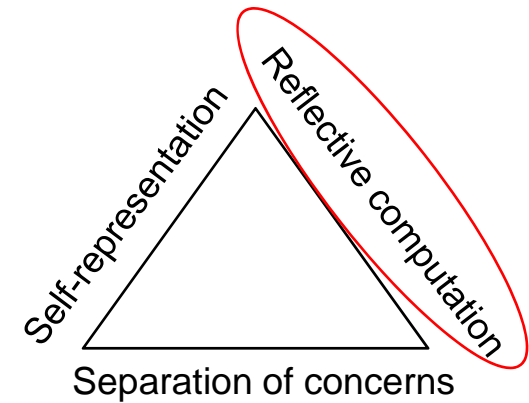


Reflective computation

- **Causality**
 - Domain – domain model
 - System – self-representation
- **Case study**
 - Traffic – domain model
 - monitoring traffic
 - Camera – self-representation
 - dedicated interfaces
 - + alive messages



Reflective computation



- **Level shifts**

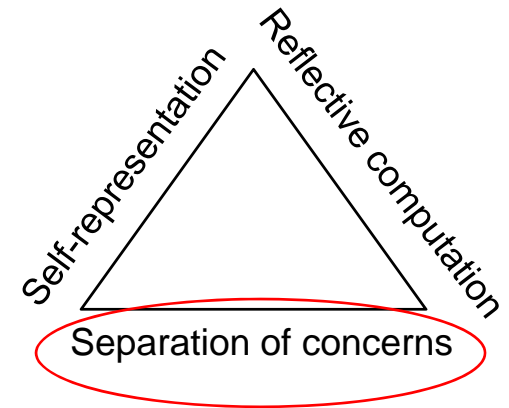
- Computation switches from base to meta level and back
- Frequency: performance issue

- **Case study**

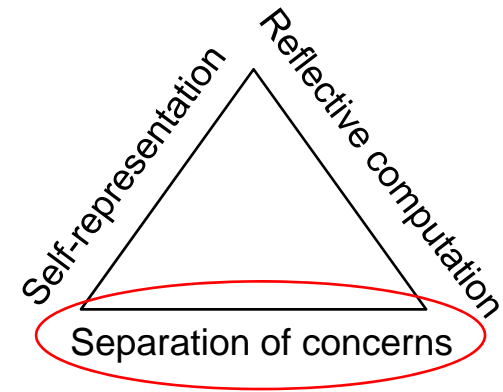
- When failure is detected (background activity)

Separation of concerns

- **Disciplined split**
 - Base level is about the domain
 - Meta level is about the system
 - Different concerns
- **Case study**
 - Domain concern: monitoring traffic
 - Meta concern: self-healing



Separation of concerns



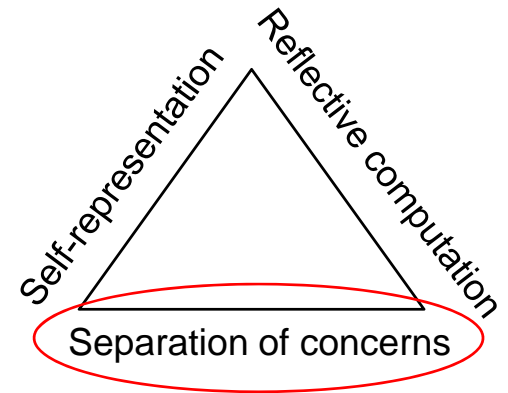
- **Transparency**

- Extent to which base level is “unaware” of reflective levels
- E.g. Touchpoints in IBM autonomic architecture: automatic generation based on description of manageability capabilities of managed resource

- **Case study**

- Dedicated interfaces
- Invasiveness of self-healing concern

Separation of concerns



- **Hierarchy**

- Meta level layers
- Kramer & Magee 2 levels:
 - Change layer: plans that react to exceptions in base level
 - Goal layer: if no plan is available, derive new plan

- **Case study**

- Single level reflective system

Other Case Studies

- **High performance computing**
 - Sensor network scenario
 - Scheduling data-parallel programs on hardware architecture
 - Dynamic Model-Driven Architecture (DMDA platform)
- **Learning fault detectors**
 - Genetic approach

Outline

- I. Objective of the paper
- II. Reflection
- III. Reflection prism
- IV. Case study
- V. Challenges**
- VI. Conclusions**

Challenges

- **Expressiveness of self-representation**
 - State of the art ADLs are not designed for providing runtime representation of the system
 - Need for expressing dynamism
 - Need for modeling domain characteristics
- **Uncertainty**
 - Abstraction introduces certain inaccuracies
 - Domain is uncertain
 - Model need to take into account uncertainty explicitly

Challenges

- **Autonomy**
 - Decentralization <> system behavior
- **Transparency**
 - To what extent can probes and gauges be realized in a transparent manner?
 - Invasiveness of certain concerns of self-adaptation
 - Disciplined approaches for integration
- **Performance**
 - Selective/dynamic interception (Fractal, OpenCom)

Outline

- I. Objective of the paper
- II. Reflection
- III. Reflection prism
- IV. Case study
- V. Challenges
- VI. **Conclusions**

Conclusions

- **Reflection is a (the?) key issue of self-adaptation**
- **Original theory for computational reflection (level of programming languages) needs to be extended**
 - Domain issues
 - Uncertainty
- **Numerous challenges**