An Architecture for Requirements-Driven Self-Reconfiguration

Fabiano Dalpiaz, Paolo Giorgini,
John Mylopoulos

June 12th, 2009

UNIVERSITY
OF TRENTO - Italy
Information Engineering
and Computer Science Department

CAiSE'09
AMSTERDAM

Outline

1. Motivation and Research Question
2. Preliminaries
3. Requirements models
4. Self-reconfiguration architecture
5. Creating the architecture for an existing system
6. Case study: smart homes
7. Summary and future work

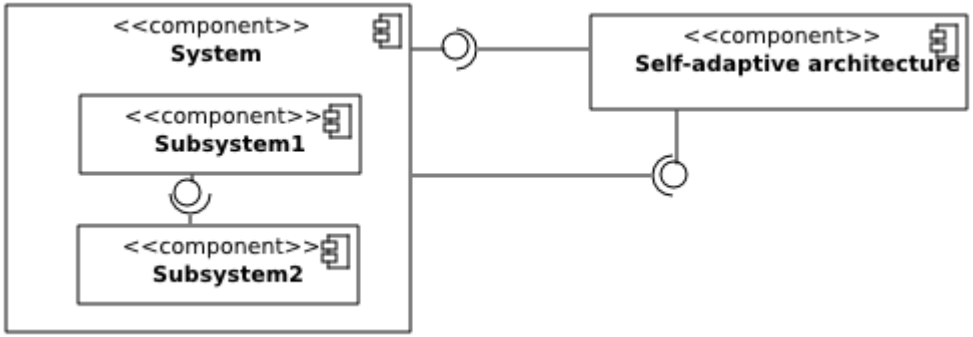
Motivation and Research Question

Motivation

- Need for software systems that
  - **Fulfill their requirements** in different operational environments
    - Changes in the context have an impact on requirements
    - Socio-Technical Systems, Ambient Intelligence
  - **Survive and respond to failures**
    - **Self-reconfigure** by switching to a configuration that is more likely to achieve the requirements
    - Under-performance is a type of failure

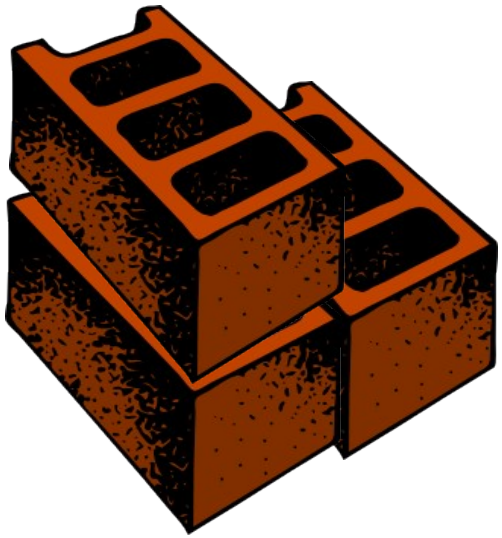
Research Question

- ”Define an **architecture** that supports **self-reconfiguration** at the level of **requirements**”
  - Logical structure
  - Select/Define Requirements Models
  - Diagnosis and Reconfiguration algorithms
  - Apply to a case study



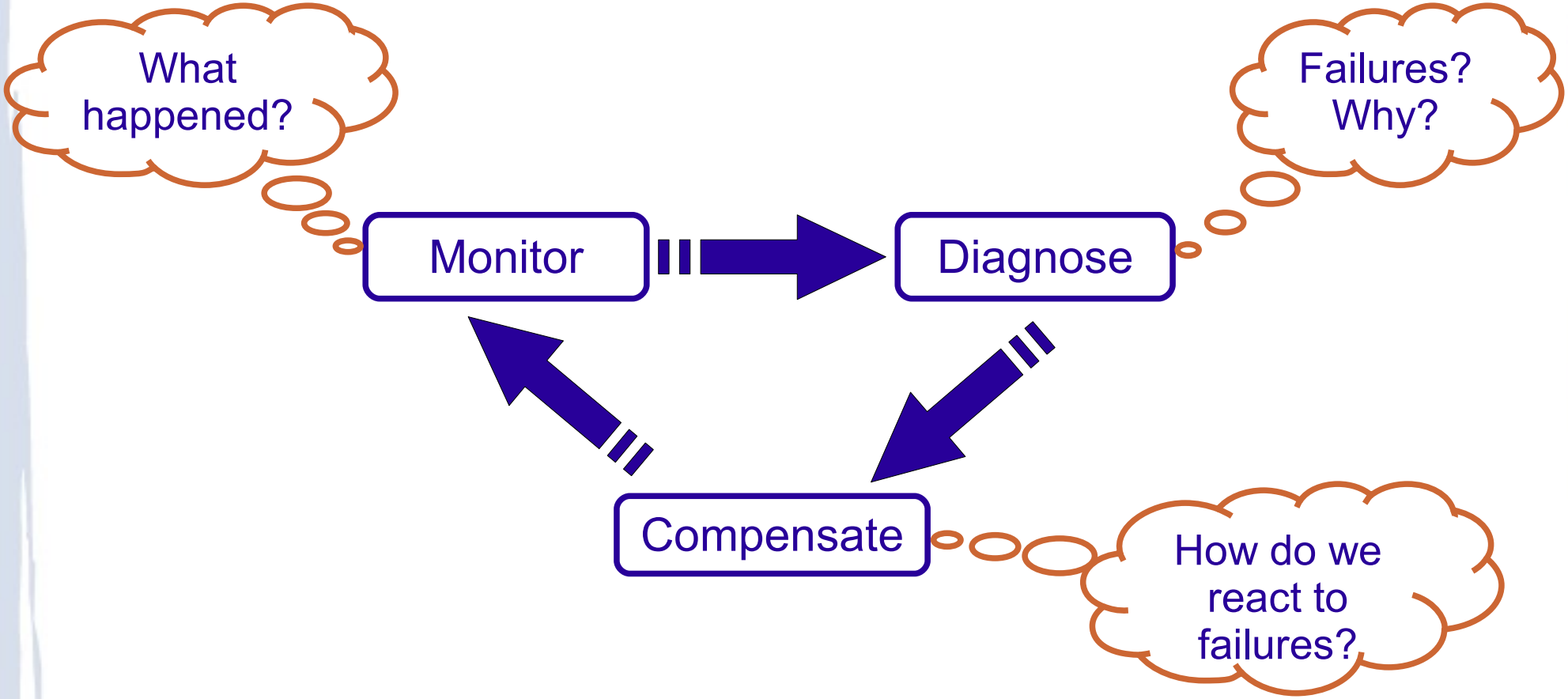
The architecture should be based on externalized adaptation [Garlan04]

Preliminaries



MDC loop

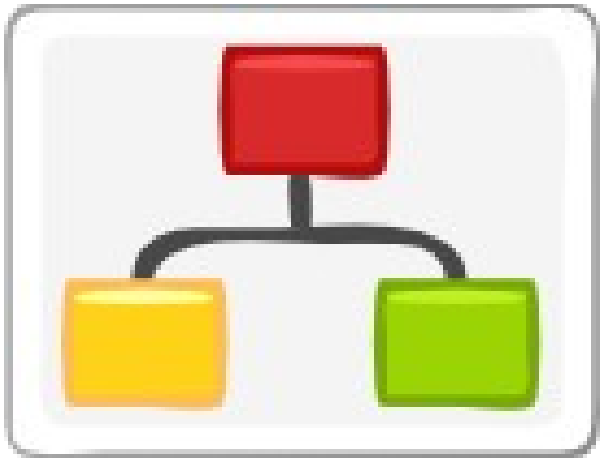
Reconfiguration follows a Monitor-Diagnose-Compensate (MDC) loop



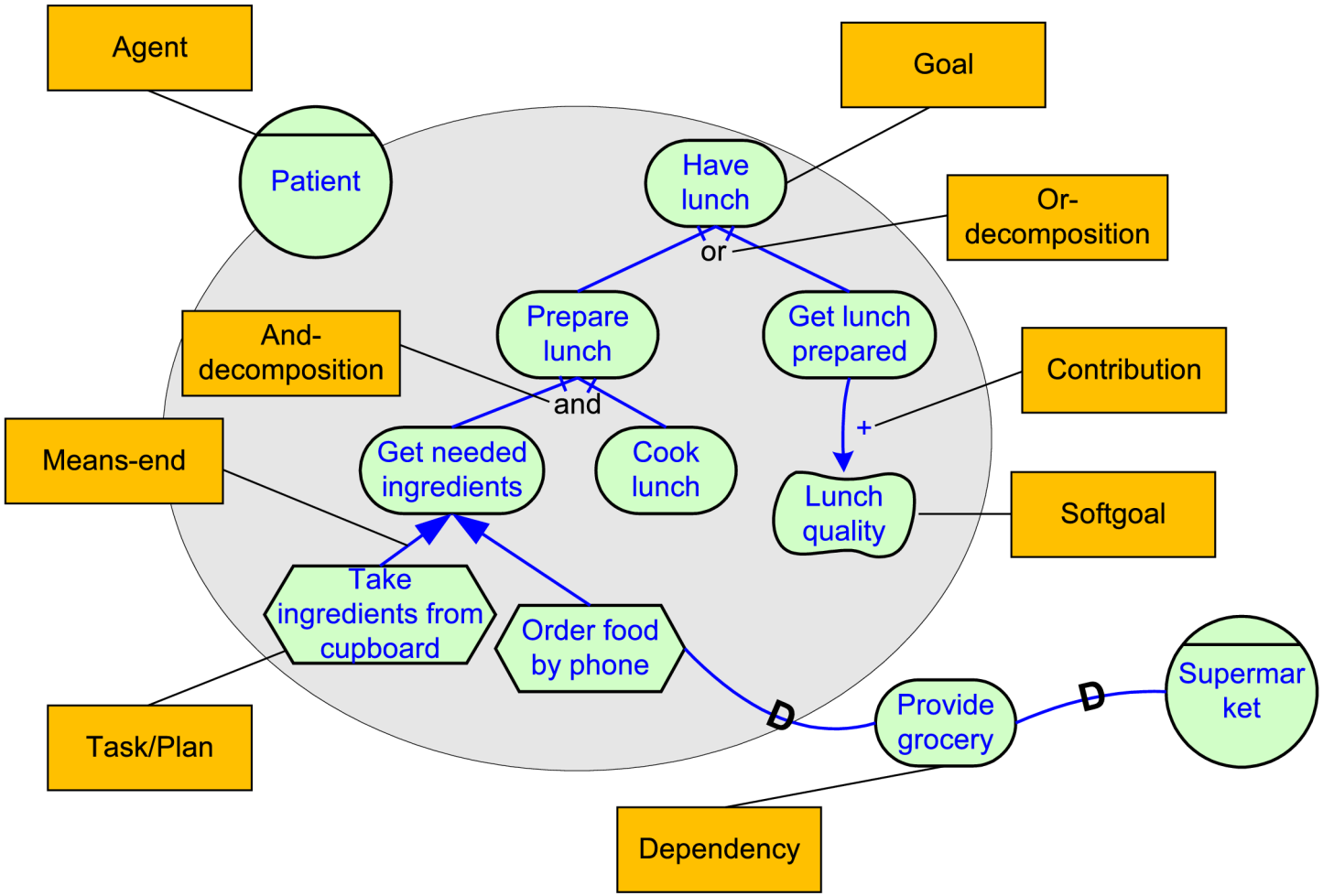
BDI-compliant behaviour

- We assume the system should behave according to the Belief-Desire-Intention (BDI) paradigm [Rao92]
  - The system is a set of agents
  - Each agent has goals (**desires**)
    - Agents goals define system requirements
  - Whenever an agent adopts a goal, it will commit to its achievement by starting an **intention**
    - An intention is an instantiated plan
  - Plans are chosen in accordance to current **beliefs**

Requirements Models



Tropos goal models

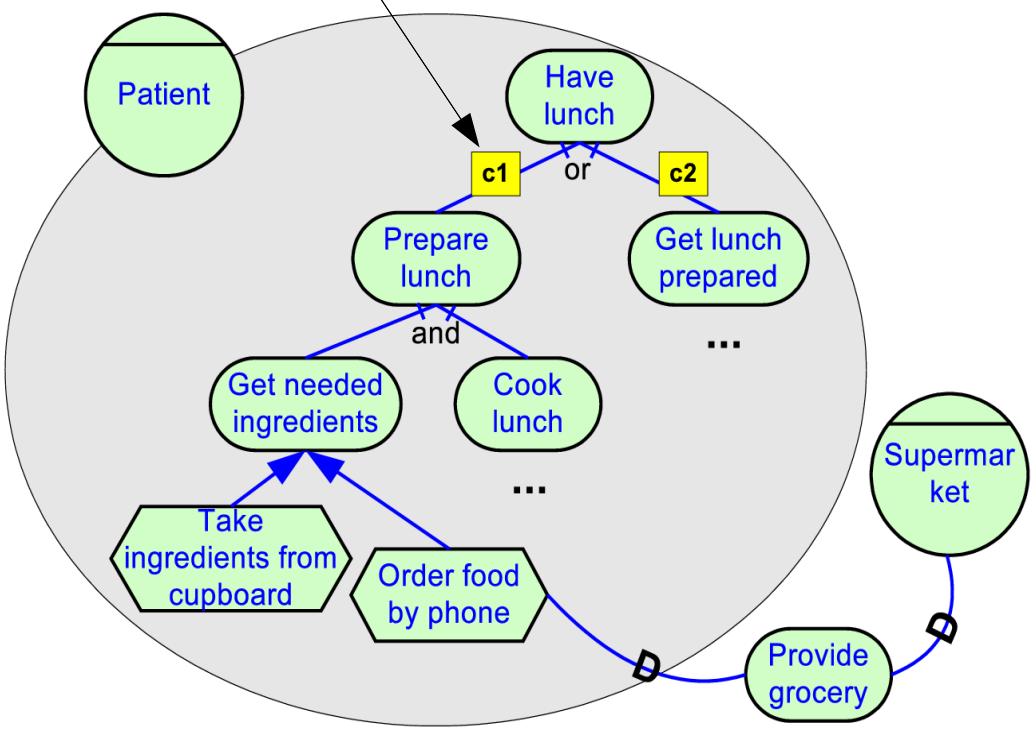


[Bresciani04]

Extended Tropos goal models

Contexts constrain variation points [Ali08]: goal ”Prepare lunch” can be achieved only in context c1

Goal instances: activation events, commitment condition, achievement condition, parameters



ACTIVATION EVENT:
“Have lunch”: it’s 12AM
COMMITMENT CONDITION:
“Have lunch”: 1 hour since activation

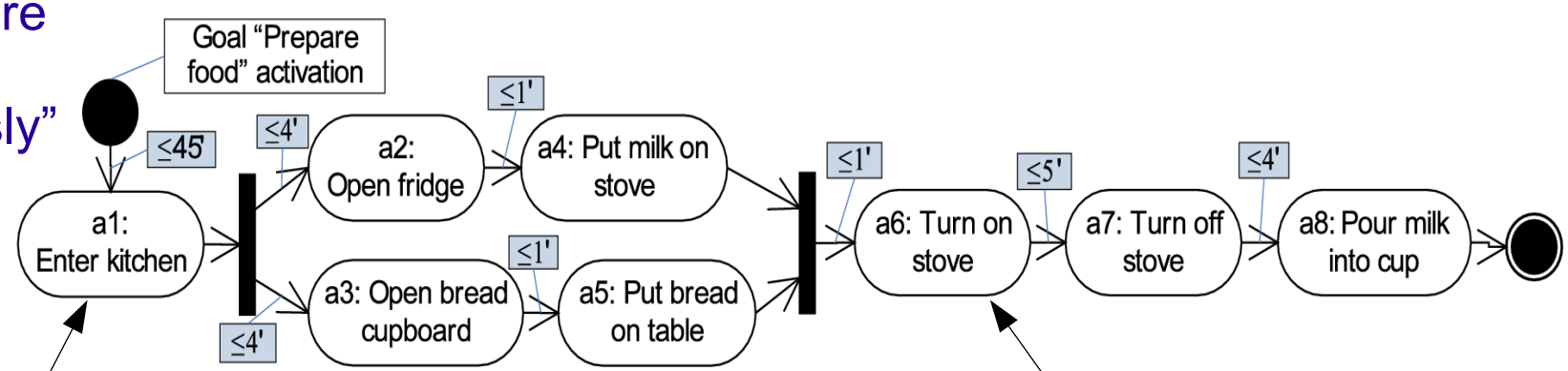
CONTEXTS:
c1: patient is autonomous
c2: patient is not autonomous

TASK PRECONDITIONS:
“Order food by phone”: Patient.house.hasPhone = true

Tasks specification

Each task is specified via a Timed Activity Diagram

Task ”Prepare breakfast autonomously”

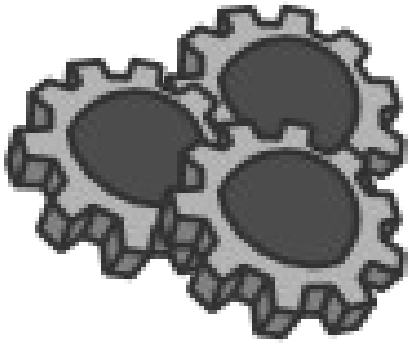


The patient should enter the kitchen within 45' after goal ”Prepare food” is activated

The patient should turn on the stove within one minute after the last activity between a4 and a5

Update: We are working on a new formalism more flexible than timed activity diagrams

Architecture

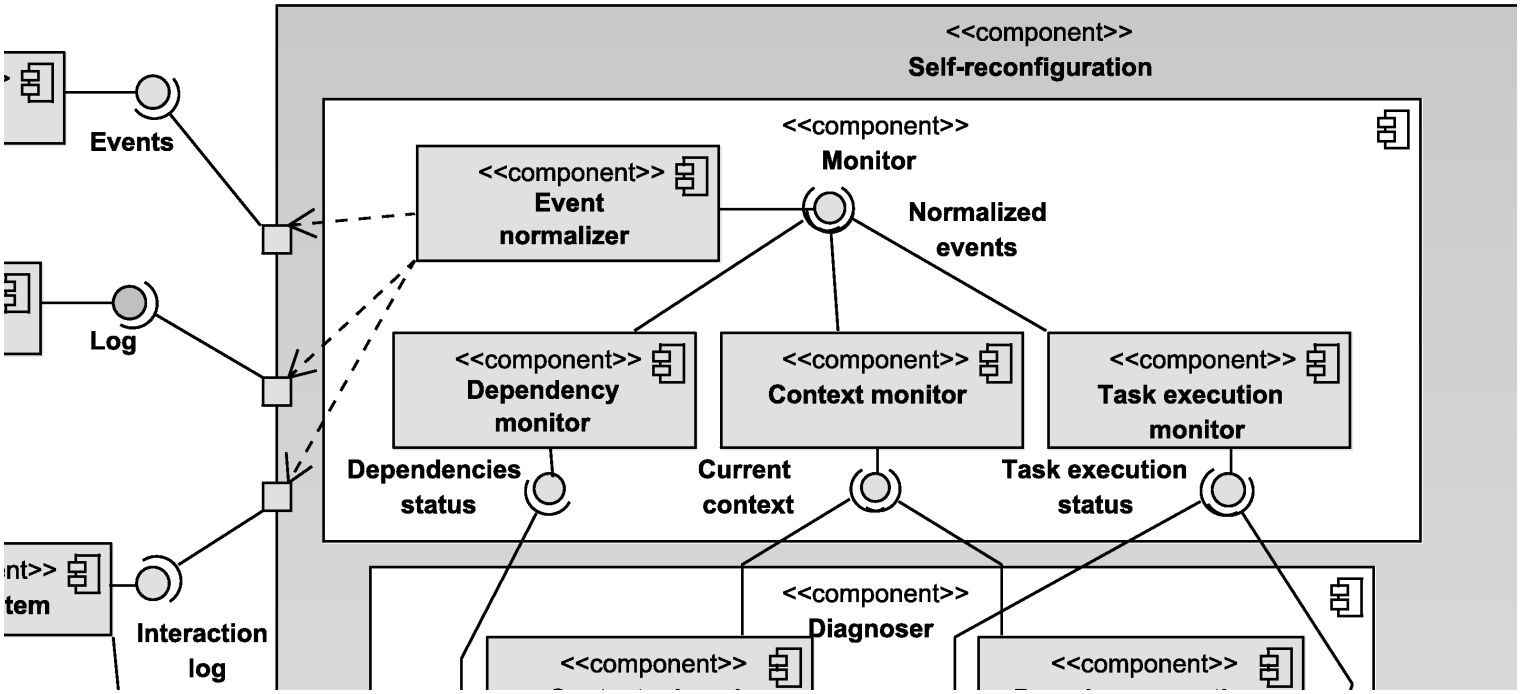


Self-reconfiguration architecture

- Internal components
  - Monitoring
  - Diagnosis
  - Reconfiguration
- External components
  - **Monitored system**: the system the architecture assists
  - **Support systems**: external agents supporting the main system
  - **Context sensors and actuators**: contextual sensors/effectors

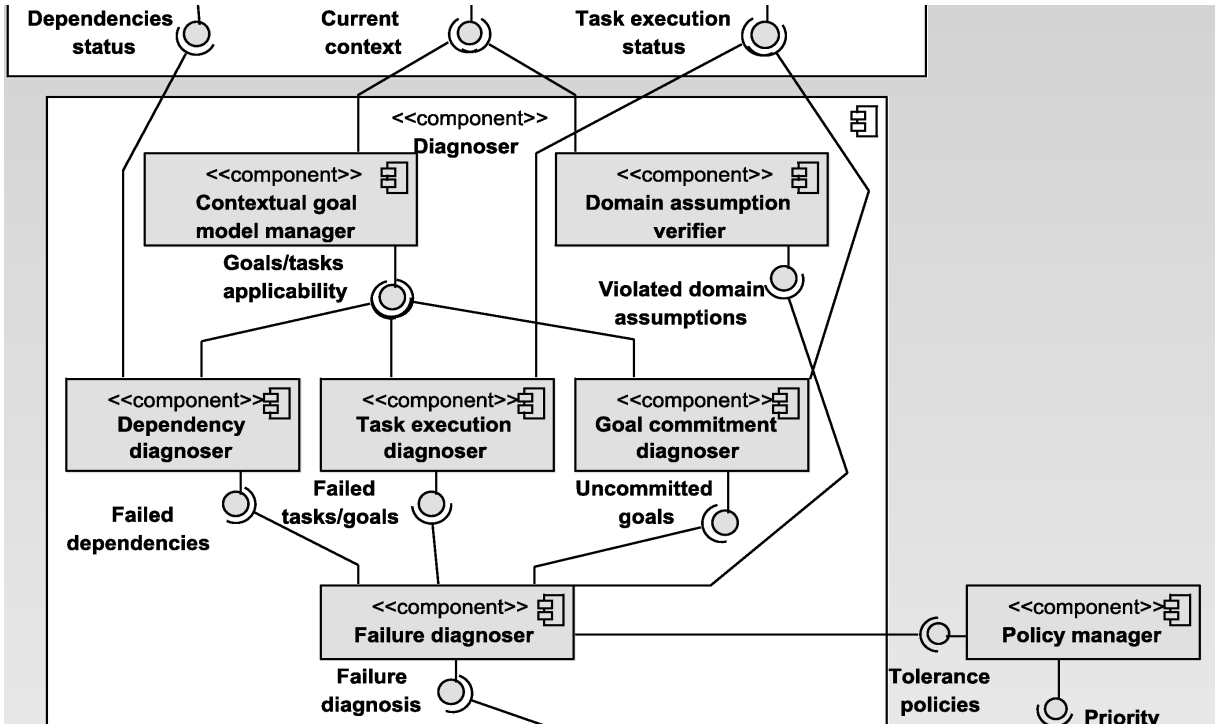
Monitoring component

- The architecture monitors **task execution**, **dependency status**, and **changes in the context**



Diagnosis component

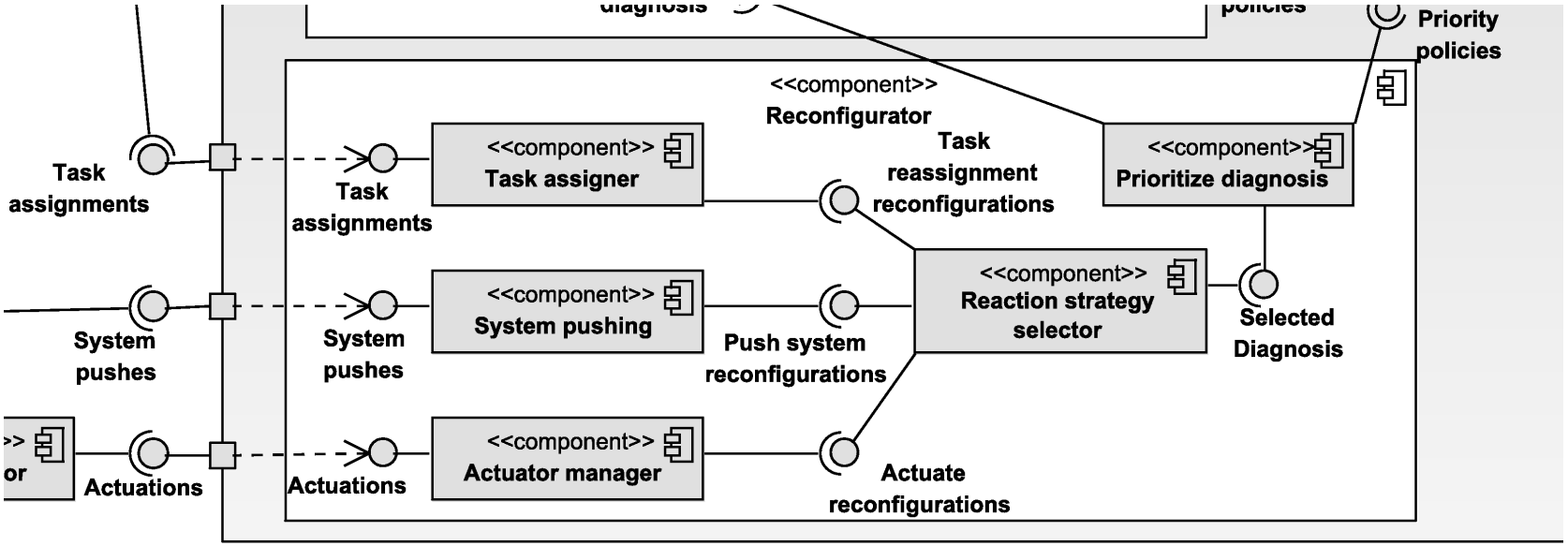
- Diagnosis checks **monitored data** against **contextual goal models** and **domain assumptions**



- A **failure** occurs when
  - Something that should happen does not occur
  - Something that should not happen does occur

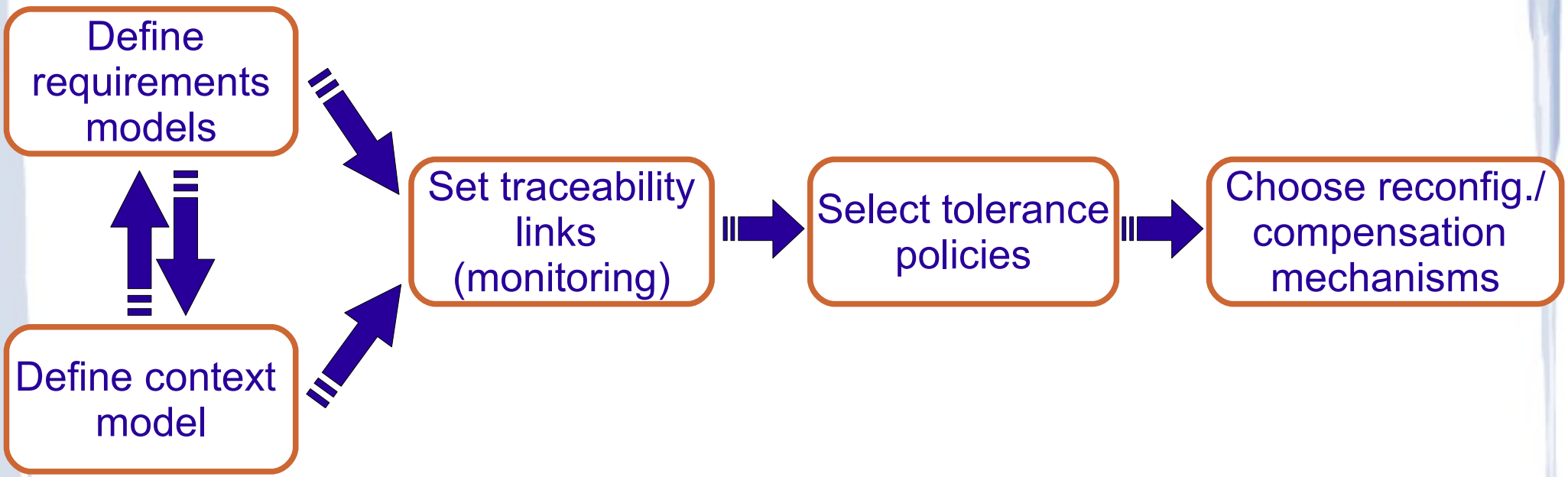
Reconfigurator component

- Reconfiguration types: **assign tasks** to supporting systems, **push** the monitored system, **control actuators** in the context
  - Diagnosis are prioritized
  - **Compensation actions** to enact semantic undo



A process to create the architecture

How can we build the architecture for an existing system?

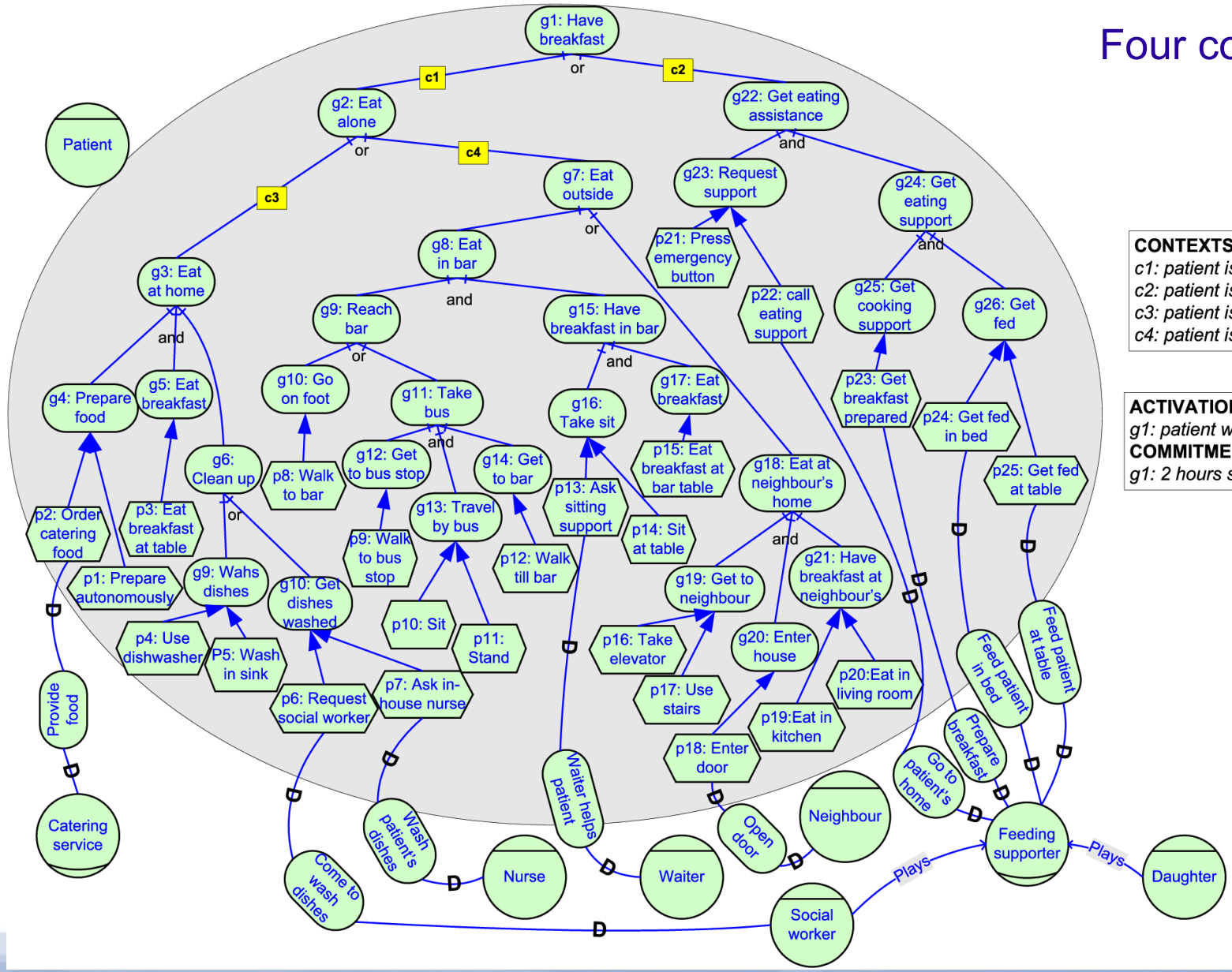


Case study: smart homes

Case study description

- **A patient is living in a smart-home**
- A smart-home is a **socio-technical system supporting the patient** in everyday activities
  - eating, sleeping, taking medicine, being entertained, visiting doctor
- Smart home and patient are equipped with **AmI devices** that
  - **gather data** (e.g., patient's health status, temperature in the house)
  - **change the context** (e.g., open the door).

Case study: goal model



Four contexts (c1-c4)

CONTEXTS:
c1: patient is autonomous
c2: patient is not autonomous
c3: patient is at home
c4: patient is not at home

ACTIVATION EVENT:
g1: patient wakes up
COMMITMENT CONDITION:
g1: 2 hours since activation event

Reconfiguration scenario (1)

8.00 AM: Mike wakes up

Goal ”**Have breakfast**” should be achieved within one hour
The architecture monitors only goals and plans where Mike is at home and autonomous.

8.20 AM: Mike enters the kitchen

The specification of task ”**Prepare breakfast autonomosuly**” makes the task status move to **in progress**

Reconfiguration scenario (2)

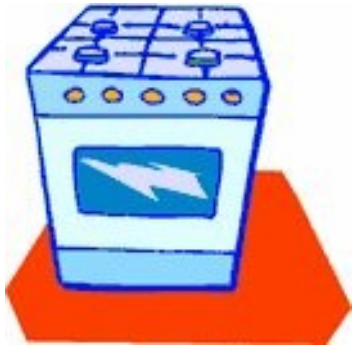
Task ”Prepare breakfast autonomously” **fails**
The **policy manager** says the **failure can't be ignored**
The **reconfigurator** chooses to **push the system** by sending an SMS to the patient

8.30 AM: Mike hasn't done any progress

Task ”Prepare breakfast autonomously” is now **in progress**

8.32 AM: Mike opens fridge and cupboard

Reconfiguration scenario (3)



The **task execution diagnoser** identifies another **failure**
The **task assigner** delegates **food provision to a catering service**

8.34 AM: Milk is not on the stove

Goal ”Have breakfast” is **achieved**
Mike has breakfast.

8.45 AM: Catering service delivers breakfast

Implementation

- Used technologies
  - Development in **Java 6**
  - Diagnosis engine: DLV-complex [Calimeri08]
    - based on **disjunctive datalog**
  - Database: H2[1] in embedded mode
  - Requirements meta-models defined using **Eclipse EMF**[2]
- Current execution mode
  - A **simulator** allows for testing the MDC loop
    - Input: requirements models, events, and compensation strategies

[1] http://www.h2database.com/
[2] http://www.eclipse.org/modeling/emf/

Comparison with related work

- Rainbow [Garlan04]
  - Reconfiguration based on **architectural models**
  - *Requirements fulfillment is not guaranteed*
- Monitoring and Diagnosis requirements [Wang07]
  - Requirements specified as **one goal model**
  - *No support for multiple actors and context*
  - *Focuses only on failures*
- Reconciling system requirements and runtime behaviour [Feather98]
  - Based on KAOS goal models
  - *Not targeted at reconfigurations of a set of agents*
  - *Exploits **predefined problem/remedy strategies***

Summary and Future Work

- We propose an architecture for self-reconfiguration
  - Takes a **distributed legacy system** as input
  - Adds **self-reconfiguration** by means of a Monitor-Diagnose-Execute cycle
  - Aims at maintaining **requirements fulfillment**
- Future work
  - Complete the architecture implementation (ongoing)
  - Apply to a wide case study (smart-home, crisis management)
  - Examine monitoring, diagnosis, and reconfiguration in case of dependencies on **external agents**

Thank you!

Questions?

dalpiaz@disi.unitn.it

References

- [Garlan04] Garlan, D., Cheng, S.W., Huang, A.C., Schmerl, B., Steenkiste, P.: Rainbow: architecture-based self-adaptation with reusable infrastructure. Computer 37(10) (Oct. 2004) 46–54
- [Ali08] Ali, R., Dalpiaz, F., Giorgini, P.: Location-based software modeling and analysis: Tropos-based approach. ER 2008 (2008) 169–182
- [Bresciani04] Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agentoriented software development methodology. JAAMAS 8(3) (2004) 203–236
- [Rao92] Rao, A., Georgeff, M.: An abstract architecture for rational agents. Proceedings of Knowledge Representation and Reasoning (KR&R-92) (1992) 439–449
- [Calimeri08] F. Calimeri, S. Cozza, G. Ianni, N. Leone, "Computable Functions in ASP: Theory and Implementation", Proceedings of the 24th International Conference on Logic Programming (ICLP 2008), 407--424

References

- [Wang07] Y. Wang, S. McIlraith, Y. Yu, and J. Mylopoulos: An Automated Approach to Monitoring and Diagnosing Requirements. 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE'07).
- [Feather98] Feather, M., Fickas, S., Van Lamsweerde, A., Ponsard, C.: Reconciling system requirements and runtime behavior. In: IWSSD '98. (1998) 50–59