An Architecture for Requirements-Driven Self-Reconfiguration

Fabiano Dalpiaz

Tropos Seminar - 20th March 2009

UNIVERSITY
OF TRENTO - Italy

Information Engineering
and Computer Science Department

Outline

1) Motivation and Research Question
2) Background
   - Preliminaries
   - Requirements Models
3) Self-reconfiguration architecture
4) Creating the architecture for an existing system
5) Case study: smart homes

1) Motivation and Research Question

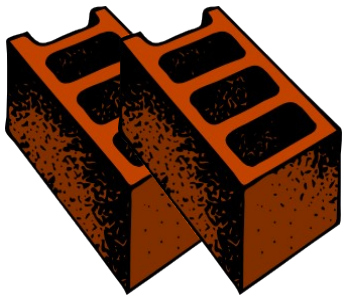
Motivation

- Need for software systems that fulfill their requirements in different operational environments
  - Smart-homes, crisis management, socio-technical systems
- Self-reconfiguration mechanisms are embedded into applications
  - Model-based adaptation [Garlan04]
- Model-based adaptation alone does not guarantee requirements fulfillment

Research Question

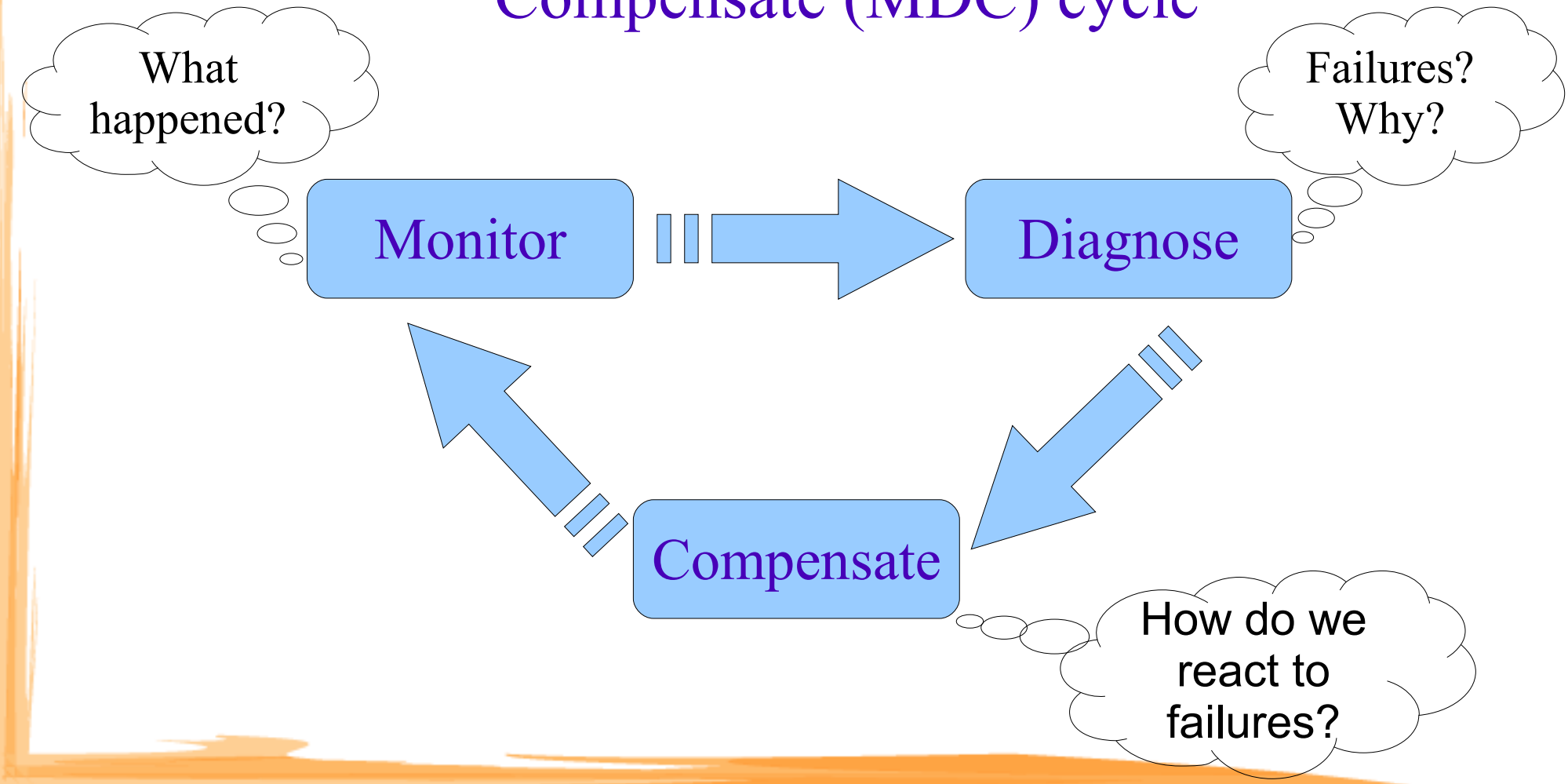
- ”Define an architecture that supports self-reconfiguration at the level of requirements by means of model-based adaptation”
  - Logical structure
  - Select/Define Requirements Models
  - Diagnosis and Reconfiguration algorithms
  - Application to a case study

2) Background



Preliminaries

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

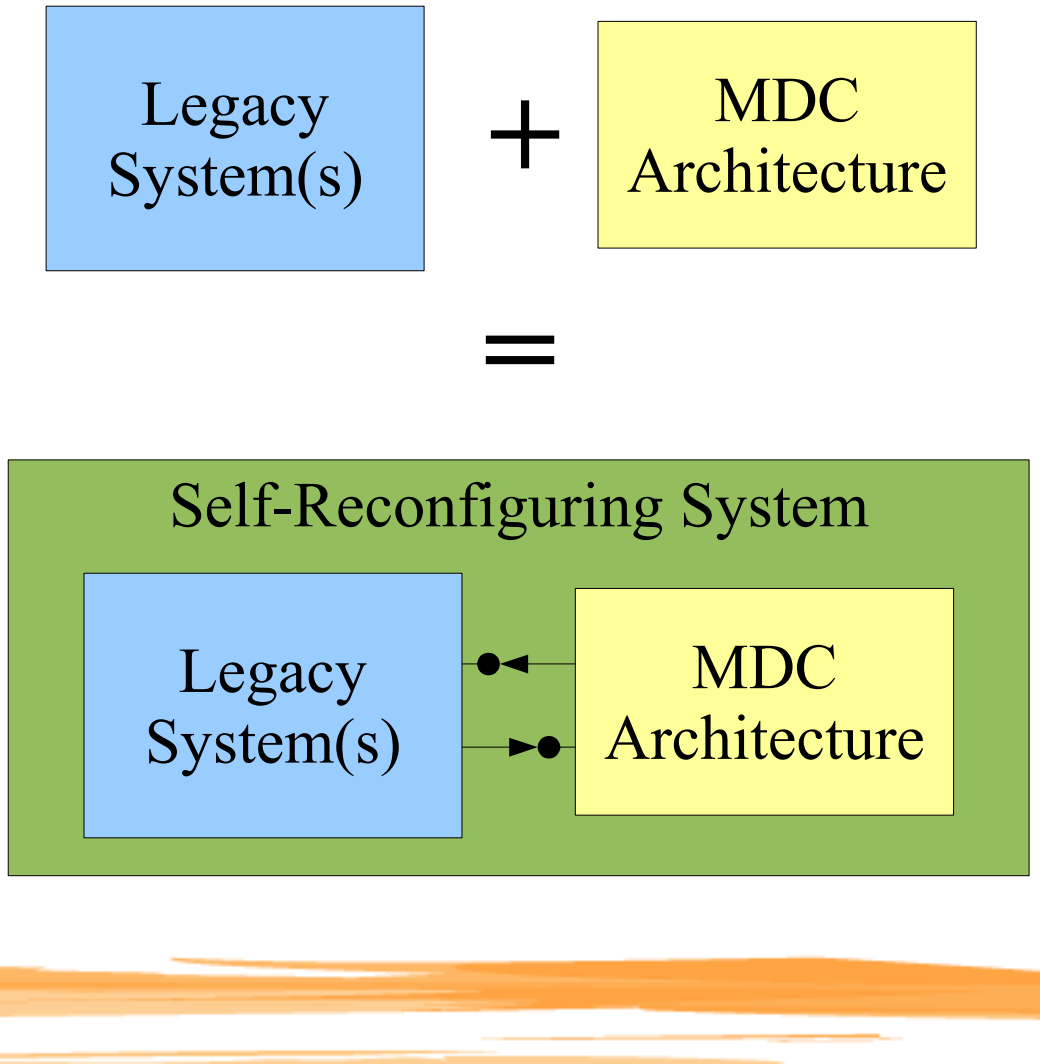


Preliminaries

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

Preliminaries

Externalized adaptation

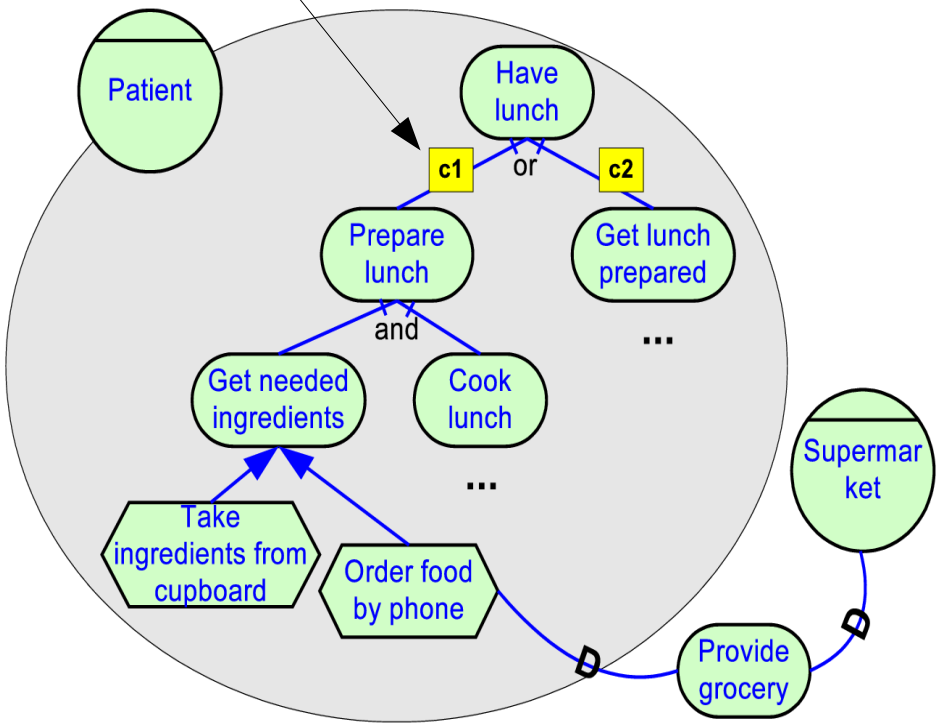


Requirements Models

Extended Tropos [Bresciani04] 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



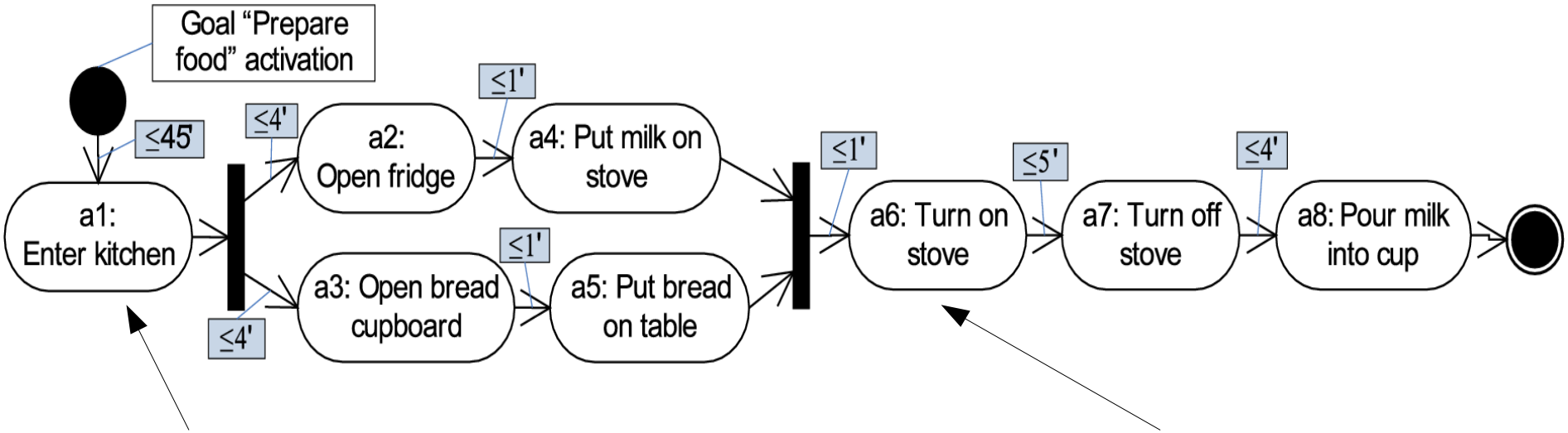
ACTIVATION EVENT:
"Have lunch": it's 12AM
COMMITMENT CONDITION:
"Have lunch": 1 hour since activation

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

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

Requirements Models

Fine-grained characterization for tasks: Timed Activity Diagrams



Activity a1 (enter kitchen) should occur within 45' after goal "Prepare food" is activated

Activity a6 (turn on stove) should within one minute after the last activity between a4 and a5 ended

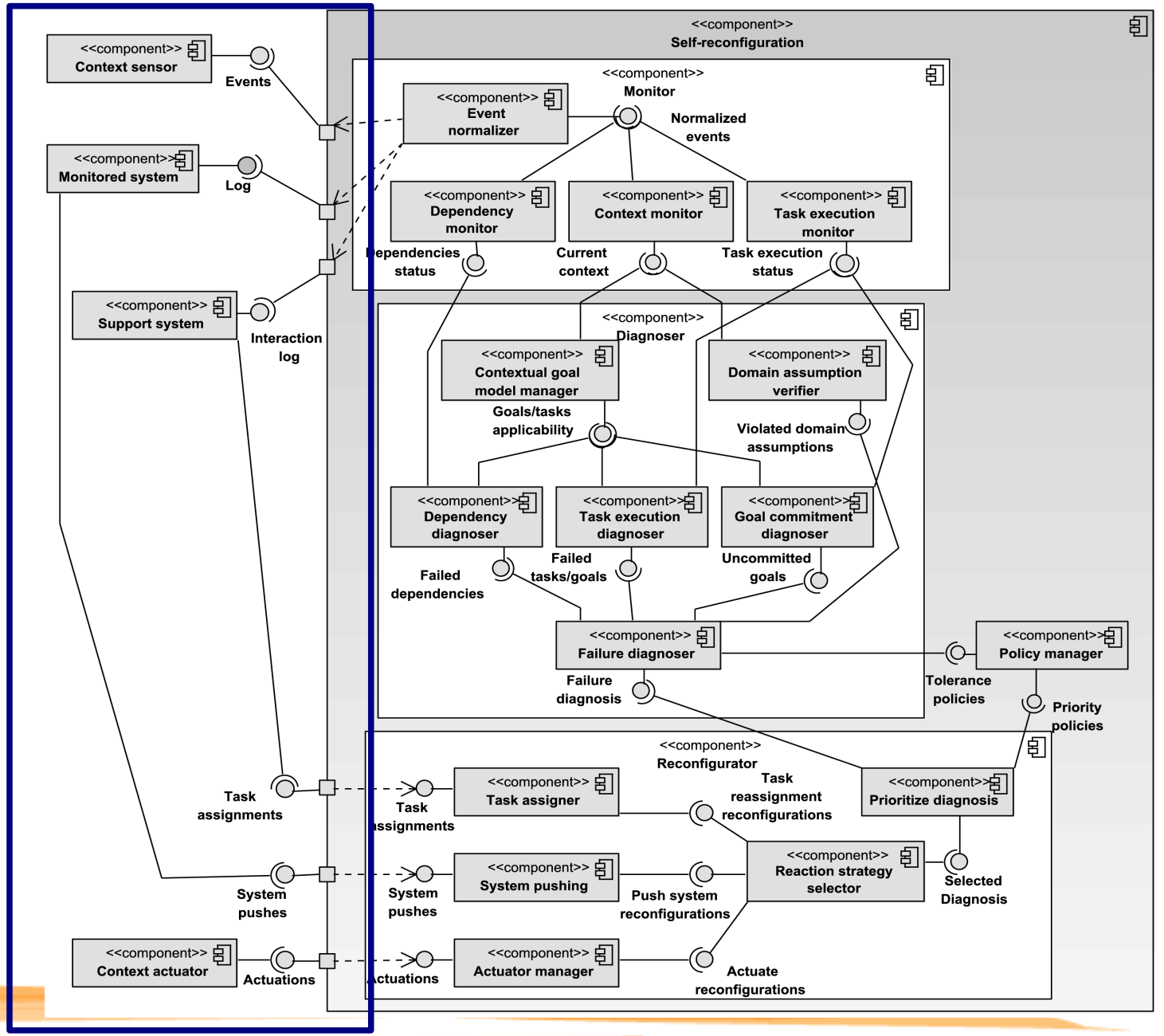
Requirements Models

- Monitoring tasks
  - Timed activity diagrams are quite procedural and inflexible
  - On the contrary, simple precondition-postcondition is not sufficient in many cases
  - A new formalism is under development
    - Based on a simplified version of event calculus
    - Timeouts for events
    - The approach is not procedural and more flexible

3) Self-Reconfiguration Architecture

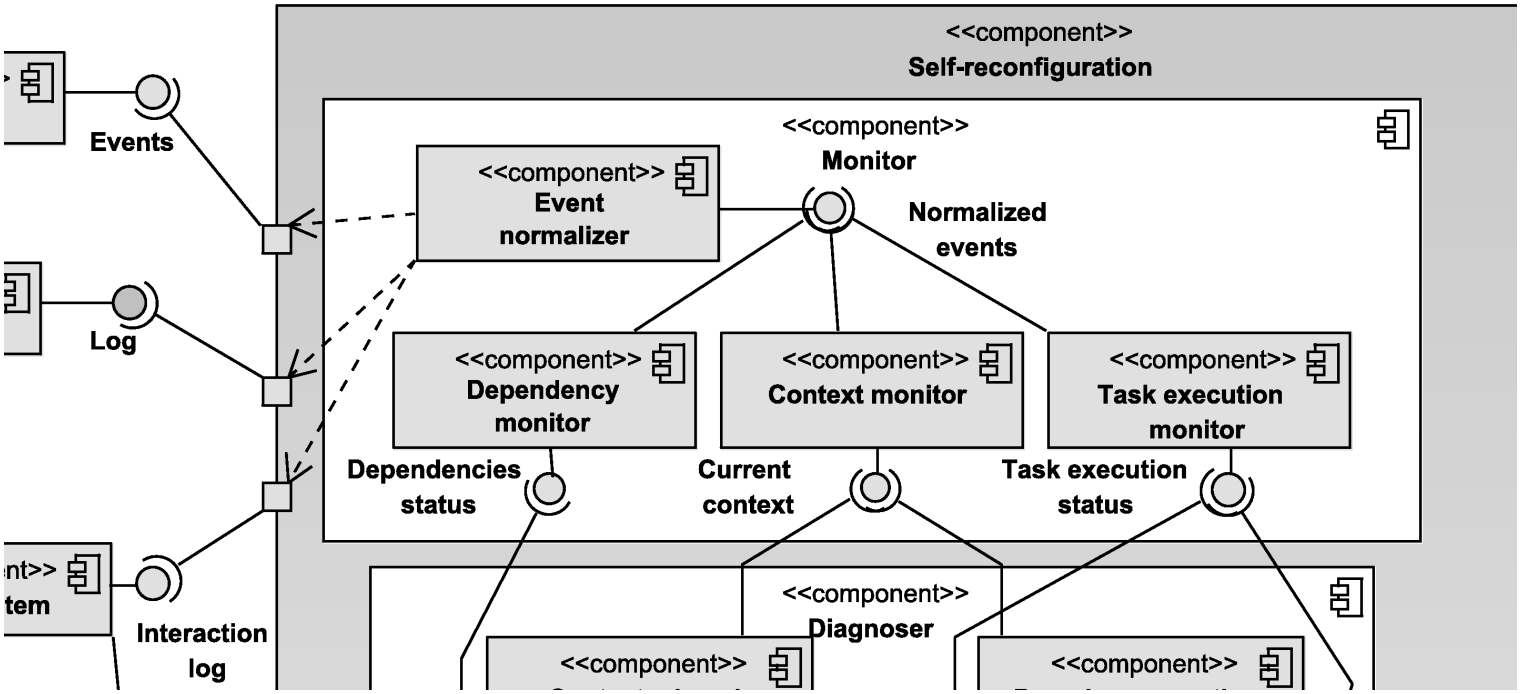
Overall view

External components: context sensors, monitored system, support systems, context actuators



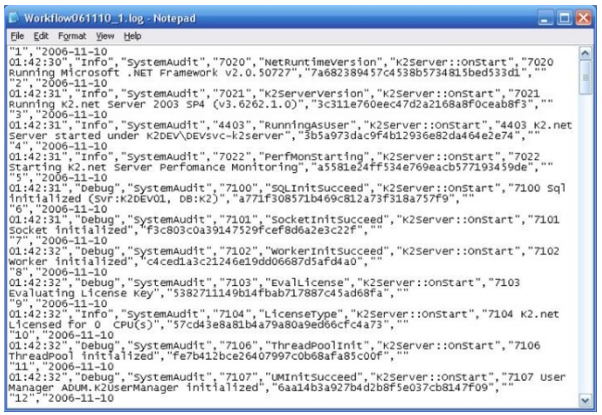
Monitoring component

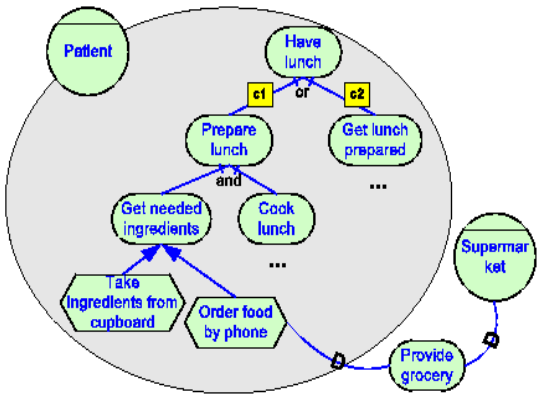
- The architecture monitors task execution, dependency status, and changes in the context



Diagnosis component

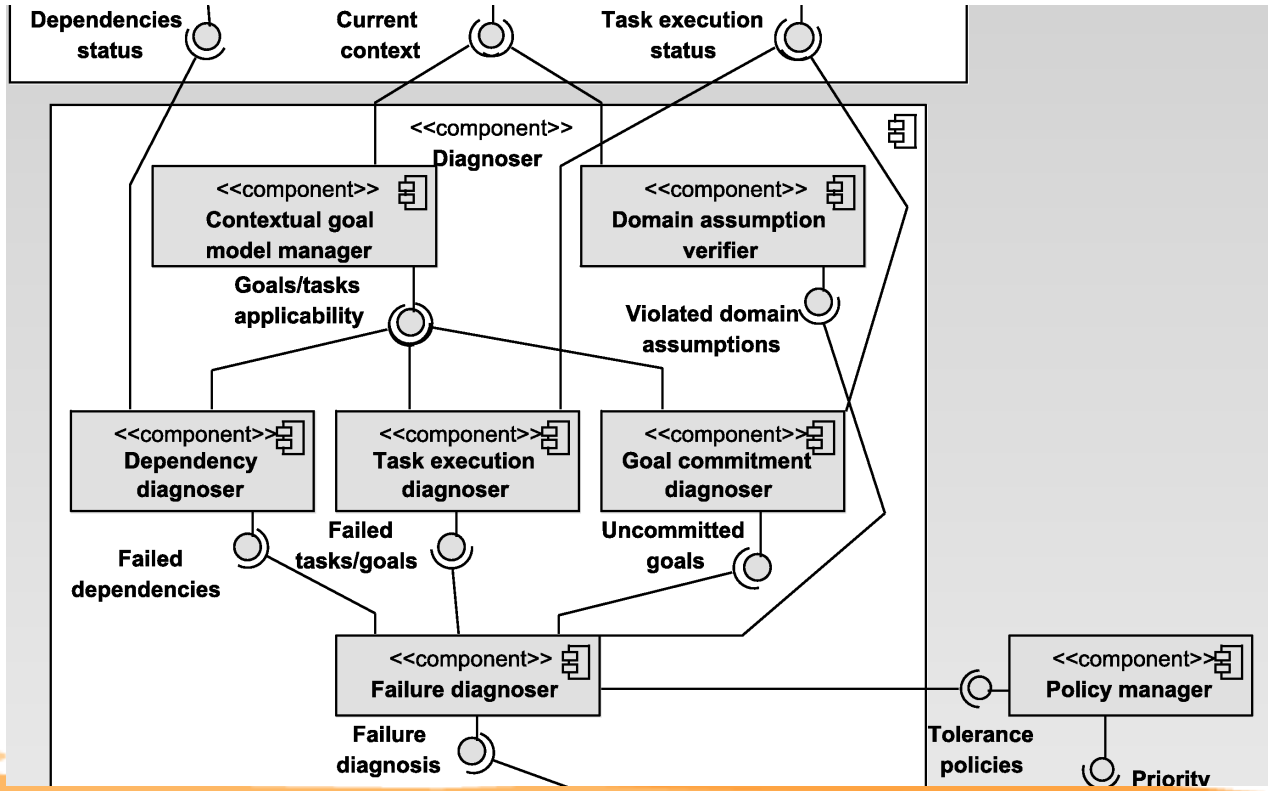
- How to diagnose failures?
  - Check monitored events against requirements models
- A failure occurs if
  - Something that should happen does not occur
  - Something that should not happen does occur





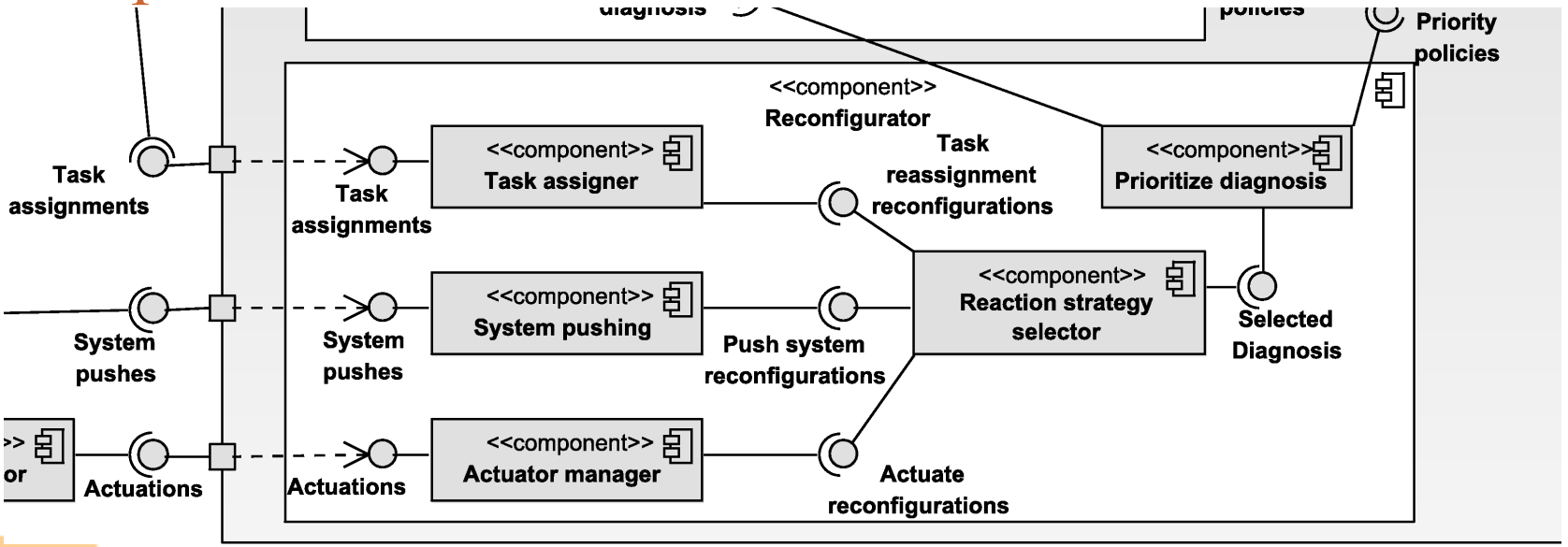
Diagnosis component

- Diagnosis checks monitored data against contextual goal models and domain assumptions
  - Failures are identified after checking policies

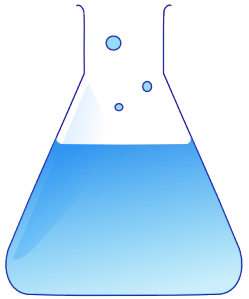


Reconfigurator component

- Reconfiguration types: task assignment to supporting systems, pushing the monitored system, control actuators in the context
  - Diagnosis are prioritized
  - Compensation actions to enact semantic undo



4) Creating the architecture for an existing system



A process to create the architecture

1) Define the context model
   - Which are the basic entities we talk about?
2) Define requirements models
   - Tropos goal model, task specification, domain assumptions
3) Establish traceability links for monitoring
   - Relate information from sensors to requirements models

A process to create the architecture

4) Select tolerance policies for diagnosis

- Define when failures do not require reaction

5) Choose reconfiguration and compensation mechanisms

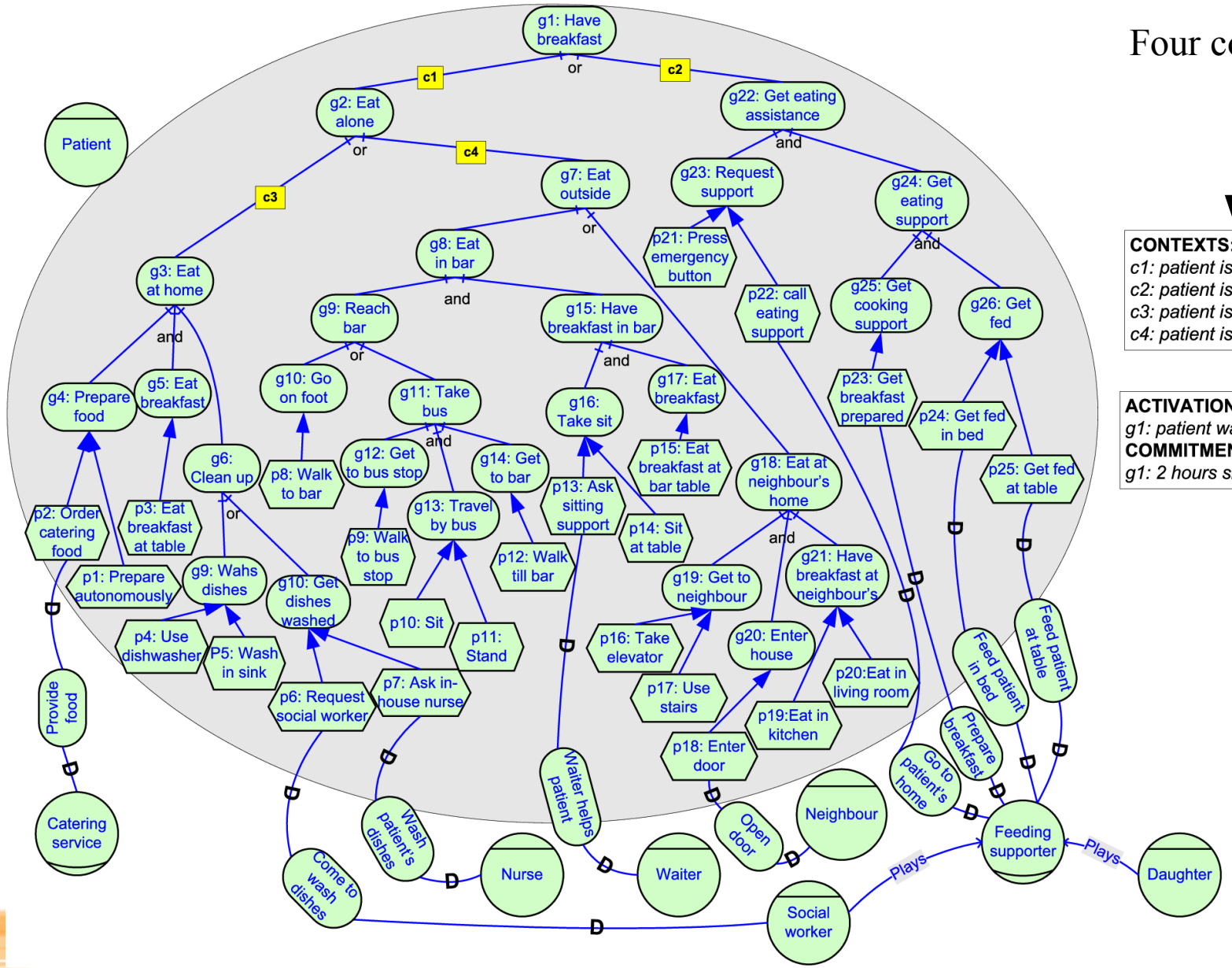
- Depends both on analyst decisions and on domain feasibility issues

5) 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
- Both 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



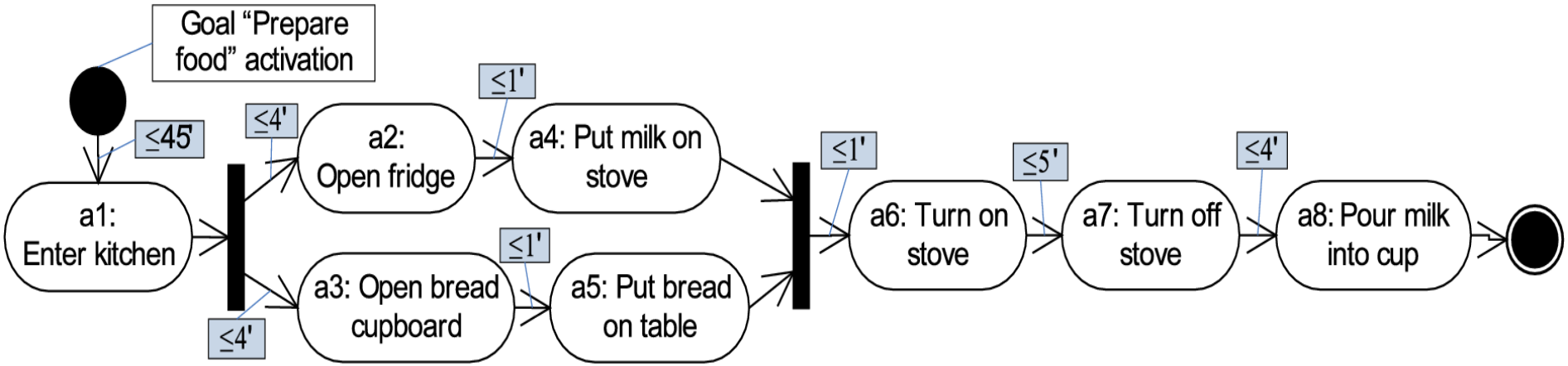
Four contexts (c1-c4)

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

Case study: timed activity diagram

- Task ”Prepare (food) autonomously” is described as follows



If the event corresponding to activity a5 does not occur within 1 minute after a3, a task failure is identified.

Case study: reconfiguration scenario

- Patient Mike wakes up at 8.00 am. Mike is autonomous (context *c1*) and at home (context *c3*).
- Mike is supposed to have breakfast (goal *g1* is activated as soon as Mike wakes up)
- The subtree of *g3* (Eat at Home) is the only allowed one, because of the current context. Thus, we do not monitor for the other sub-trees
- At 8.20 am Mike enters the kitchen: checking the activity diagram for task *p1* against this event changes the status of the goal *g4* to in_progress.
- At 8.25 Mike hasn't neither opened the fridge nor opened the bread cupboard. This violates the specification of *p1* (see previous slide), whose state is now fail
- The policy manager component says not to ignore this failure

Case study: reconfiguration scenario

- The reconfiguration strategy selector component selects to push the system, and the system pushing component sends a notification to the patient through an SMS message
- This changes the mind of Mike, which opens the fridge (*a2*), opens the bread cupboard (*a3*), and puts bread on table (*a5*). These events are compliant with the task specification, thus the task is in_progress.
- Anyhow, Mike does not put milk on stove (*a4*) within one minute since *a2*, therefore a new failure is diagnosed by the task execution diagnoser component.
- The compensation to address this failure is to automate *p2*, and the task assigner component assigns it to a catering service.
- An alternative scenario evolution is that Mike exits house (the context c4 is true, c3 is not valid anymore).

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
  - Implement the architecture (ongoing)
  - Apply to a wide case study (a real smart-home)
  - Examine monitoring, diagnosis, and reconfiguration in case of dependencies on external agents

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