FONDAZIONE
BRUNO KESSLER

On the use of the Goal-Oriented Paradigm for System Design and Law Compliance Reasoning

Authors: M. Morandini, L. Sabatucci, A. Siena,
J. Mylopoulos, L. Penserini, A. Perini, A. Susi

Speaker: Luca Sabatucci

Hammamet, 08/06/2010, i* workshop

Common concerns and paradigms in three different contexts

- (RE) when analysts have to build a requirements specification compliant with a set of laws
- (DESIGN) when designers have to choose a suitable design pattern
- (GD EXECUTION) when adaptive software agents have to take run-time decisions

Law Compliance Reasoning

contact: Alberto Siena
(siena@fbk.eu)

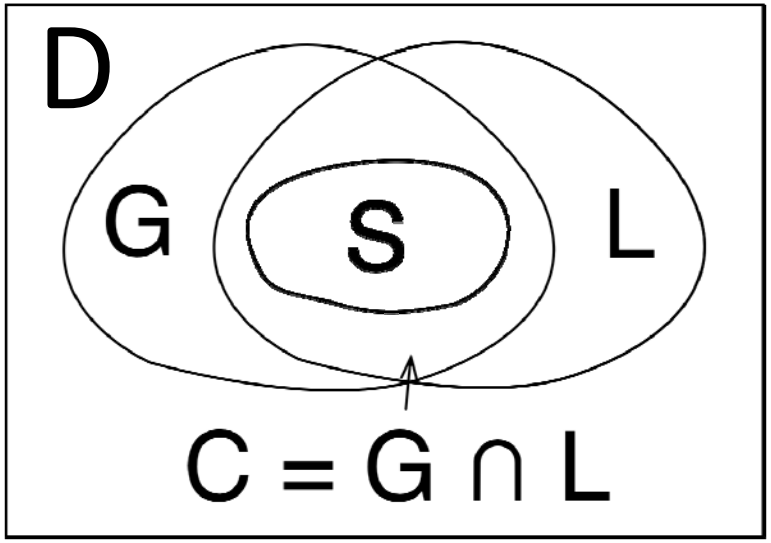
NOMOS: The Problem of Law in Requirements Engineering

- No clear-cut separation from the “software” and the “physical” world
- A choice in the software world may have effects on the physical world, and viceversa
- New laws trying to regulate this reality
- New effects of old laws

NOMOS: Problem statement

- Laws regulate the increased pervasiveness of IS
- Laws are source of requirements
- However law prescriptions are NOT goals
  - Stakeholders ***want*** to achieve goals,
  - law prescriptions are ***imposed to*** stakeholders
  - Law prescriptions can contradict goals

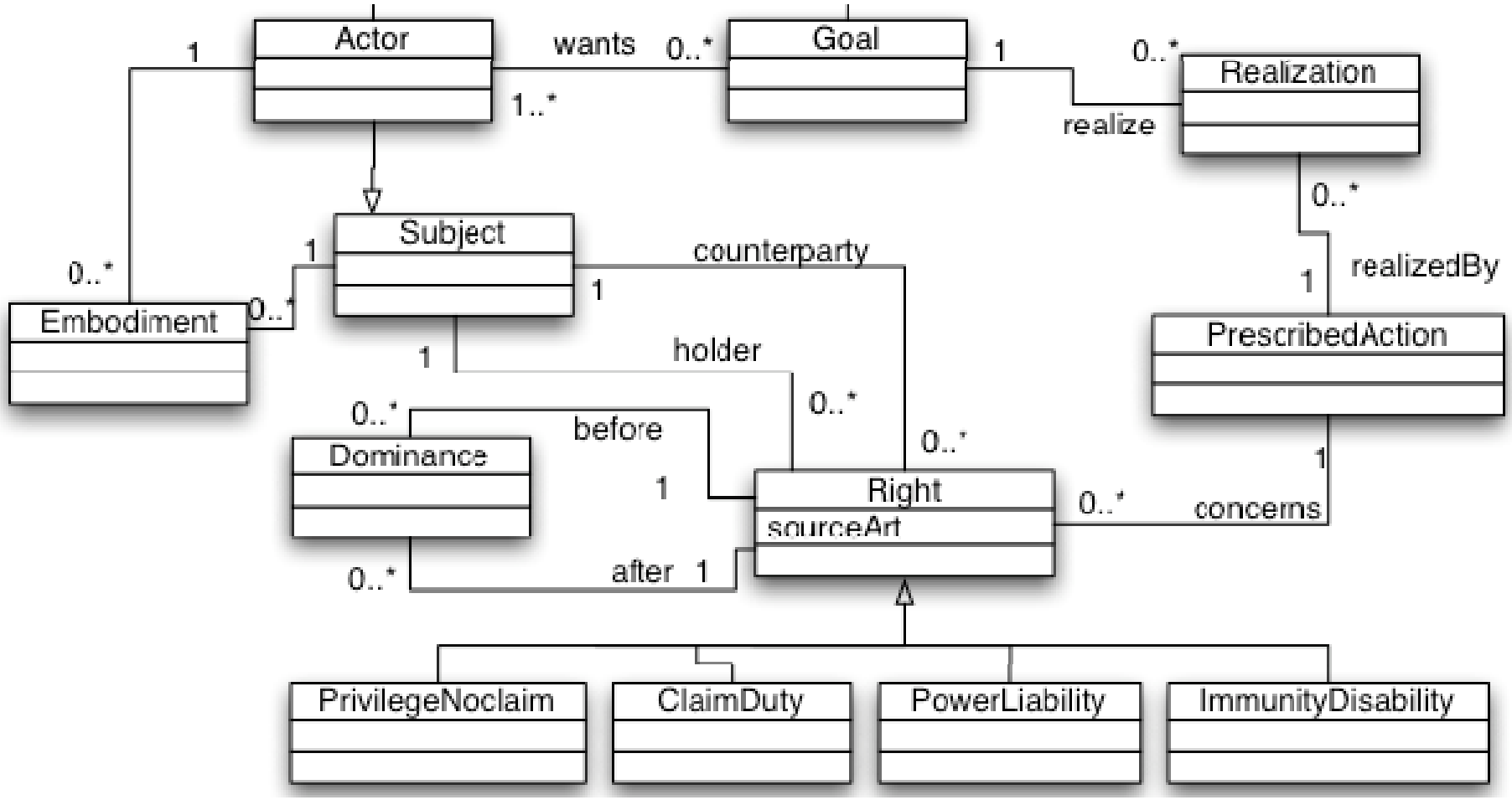
D = Domain assumptions
G = Set of states of the world represented by the stakeholders goals
L = Set of states of the world described by law sentences
S = Set of states of the world specified to the system



The NOMOS Framework

- A **modeling language** for legal concepts and software requirements
- A **modeling process** for systematically going from a model of law to a model of law-compliant requirements
- A **set of properties** for analyzing models of requirements with respect to:
  - completeness;
  - traceability;
  - audit-ability;
  - vulnerability;

Meta-model of a NP



Publications

- A. Siena. Engineering Law-Compliant Requirements: the Nomos Framework. PhD Thesis
- A. Siena, J. Mylopoulos, A. Perini, and A. Susi. The Nomos framework: Modelling requirements compliant with laws. Technical Report TR-0209-SMSP, FBK – Irst, 2009.
- A. Siena, J. Mylopoulos, A. Perini, A. Susi: Designing Law-Compliant Software Requirements. ER 2009: 472-486
- A. Siena, A. Perini, A. Susi, J. Mylopoulos: Towards a framework for law-compliant software requirements. ICSE Companion 2009: 251-254
- A. Siena, N. Maiden, J. Lockerbie, I. Karlsen, A. Perini, A. Susi: Exploring the Effectiveness of Normative i* Modelling: Results from a Case Study on Food Chain Traceability. CAiSE 2008: 182-196

Future Work

- Argumentation-based compliance evidence
- Integration with
  - natural language processing; security analysis; risk analysis; ...
- Qualitative and quantitative analyses
  - Model complexity; readability;
- Compliance in the Internet of services

System Design Reasoning

Contact: Luca Sabatucci
(sabatucci@fbk.eu)

DESIGN PATTERN

- Design Patterns are more than solutions
- Motivations describe ‘why’ to apply the pattern
- The reuse is described as a general context and a set of forces to balance
- The applicability
  - conditions to meet for applying the pattern
  - consequences of reuse, result of force balance
- Implementing issues describes design alternatives

Proxy

Object Structural

Intent

Provide a surrogate or placeholder for another object to control access to it.

Motivation

One reason for controlling access to an object is to defer the full cost of its creation and initialization until we actually need to use it. Consider a document editor that can embed graphical objects in a document. Some graphical objects, like large raster images, can be expensive to create. But opening a document should be fast, so we should avoid creating all the expensive objects at once when the document is opened. This isn't necessary anyway, because not all of these objects will be visible in the document at the same time.

[...]

Applicability

Proxy is applicable whenever there is a need for a more versatile or sophisticated reference to an object than a simple pointer. Here are several common situations in which the Proxy pattern is applicable:

1. A **remote proxy** provides a local representative for an object in a different address space. NEXTSTEP [Add94] uses the class NXProxy for this purpose. Coplien [Cop92] calls this kind of proxy an "Ambassador."

2. A **virtual proxy** creates expensive objects on demand. The ImageProxy described in the Motivation is an example of such a proxy.

[...]

Consequences

The Proxy pattern introduces a level of indirection when accessing an object. The additional indirection has many uses, depending on the kind of proxy:

1. A remote proxy can hide the fact that an object resides in a different address space.

2. A virtual proxy can perform optimizations such as creating an object on demand.

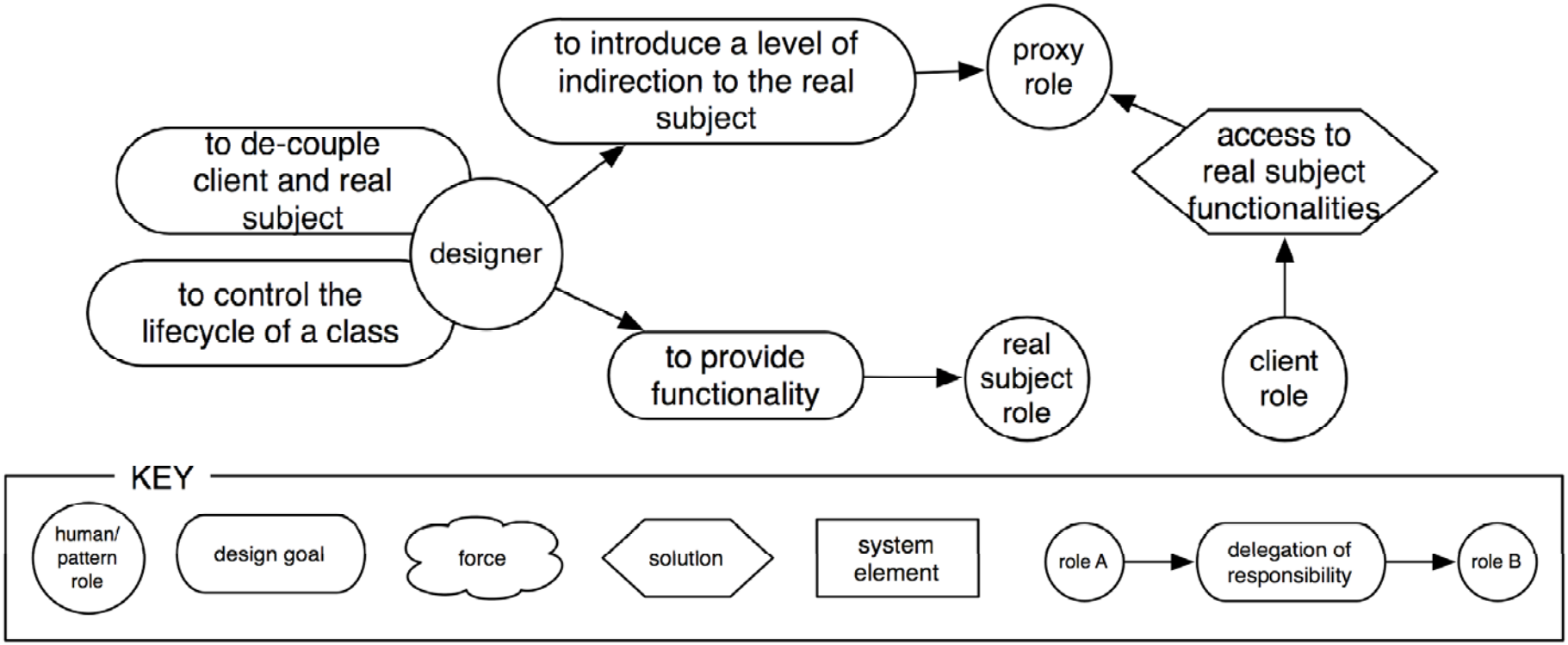
The i* framework and design patterns

- the designer is the main actor who delegates design problems to the pattern
- pattern roles are actors, which hold design responsibilities
- a design goal is a condition of the modeling activity to achieve
- the solution is provided as a collection of tasks
- system elements are resource to manipulate

DESIGNER'S NEEDS FOR THE PROXY PATTERN

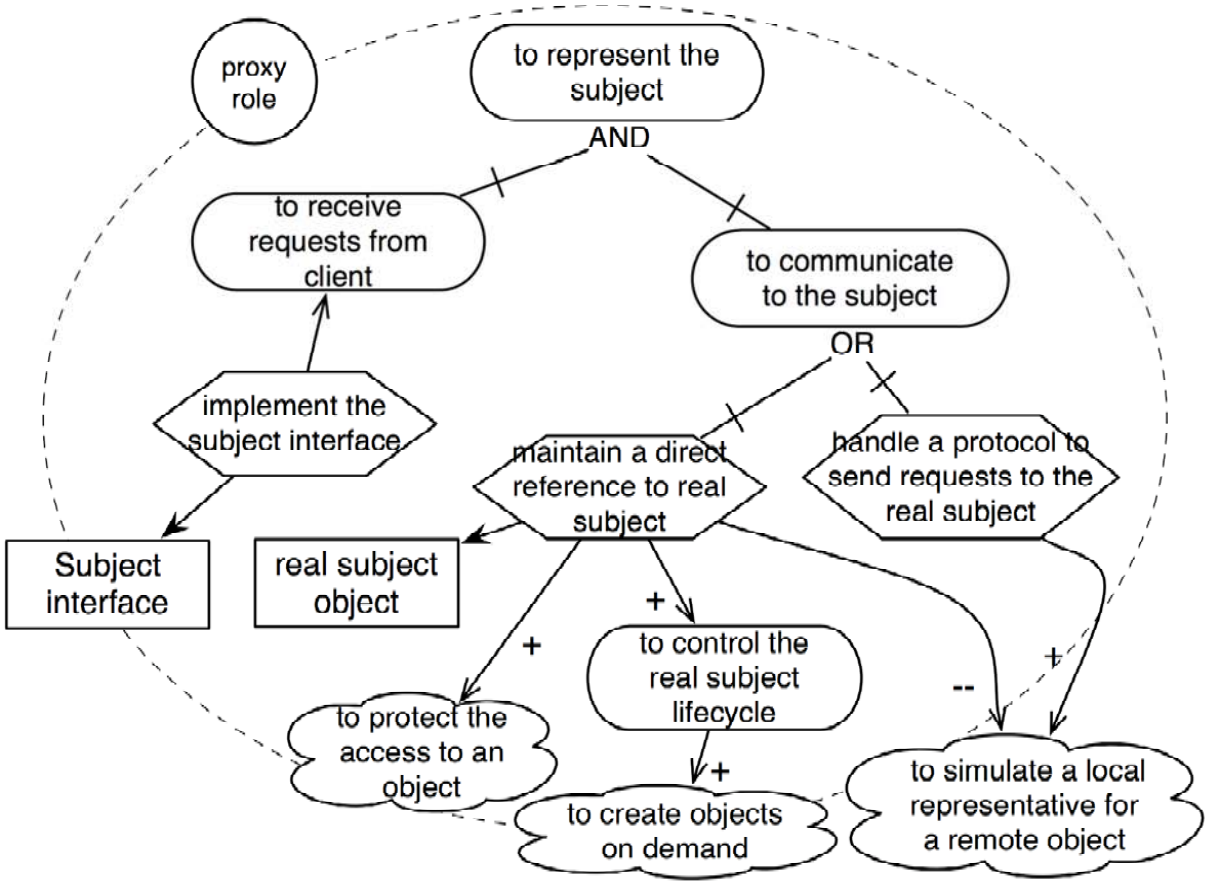
Design Goals	Quality Properties
to decouple a class from its clients	distributed subsystem
to control the lifecycle of a class	speed up the class instantiation
to delay the creation of a class	reduce the memory allocation

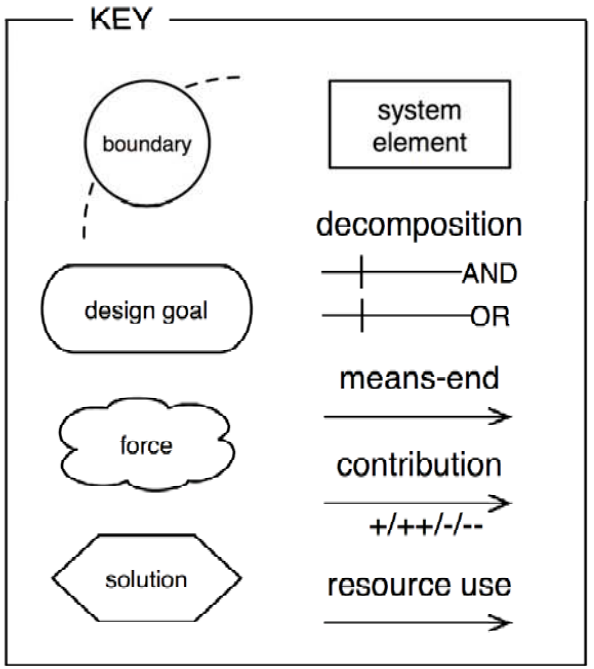
The Actor Model (Proxy Pattern)



A **Dependency** describes how a source actor depends on a destination actor, for a responsibility

The Goal Model (Proxy Pattern)





OR **Decompositions** used for detailing alternatives

A **Means-End** for providing plans to goals

A **Contributions** as a mean for choice selection

Benefits

- Understandability
  - a couple of compact diagrams for reporting the most relevant information
- Quick Browsing of the Repository
  - explicit structure where intent, applicability and consequences are highlighted
- Reuse and Traceability
  - documenting motivations for design choices
  - a pattern is not represented as a rigid template, but as a reasoning process to customize for the specific context

Publications

- L. Sabatucci, M. Cossentino, A. Susi. Introducing Motivations in Design Pattern Representation. In Formal Foundations of Reuse and Domain Engineering (ICRS'10), Washington DC, 2010.

Future Works

- Design Pattern Composition
- Empirical Study
- Tool for automatic support of pattern reuse

Adaptive Agent Systems

Contact: Mirko Morandini
(morandini@fbk.eu)

TROPOS FOR ADAPTIVE AGENTS

- Goal models are used in many agent-oriented methodologies
- BUT most AOSE methodologies loose the concept of goal in the later development phases

How can we deal with goal models at run-time?

Modeling *Self*-Adaptivity by BDI Agents

Self-adaptive systems, we consider, are able to

- identify changes in the environment and dynamically adapt their behaviour to reach their goals
- prevent goal failure, managing error recovery.

Approach

Main idea: ***preserve goals and high-level alternatives*** in all development phases until **implementation** and **run-time**.

- *Provide a framework for the modelling of adaptive systems, in which **goals**, **failures** and the **environment** are treated as first-class abstractions.*
- Define a (automated) mapping from goal models to software (BDI) agents implementing the desired run-time behaviour.

Extending Tropos Goal modelling for Self-Adaptivity

(examples along 3 main concerns)

1) Goal types:

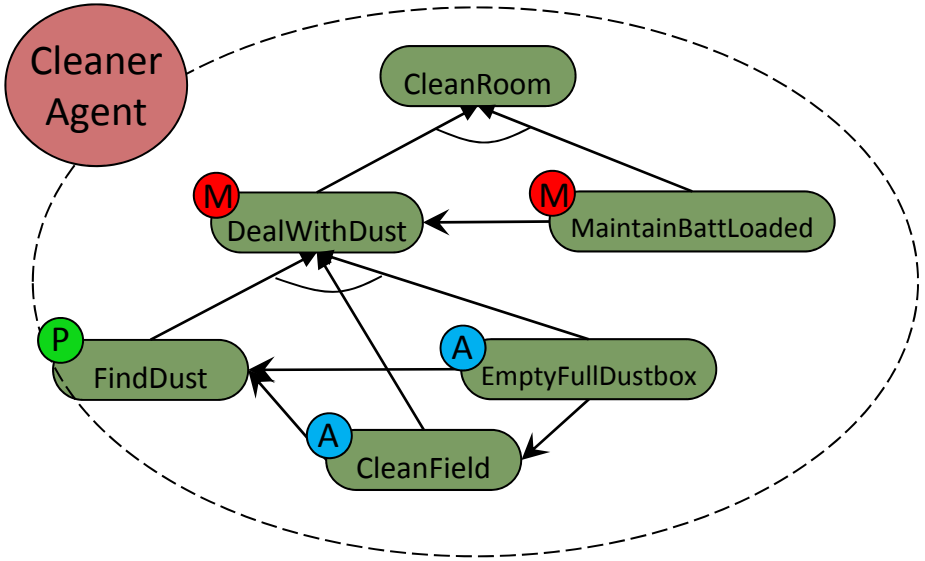
Maintain: maintain a state (M)

Achieve: reach a state (A)

Perform: do some action (P)

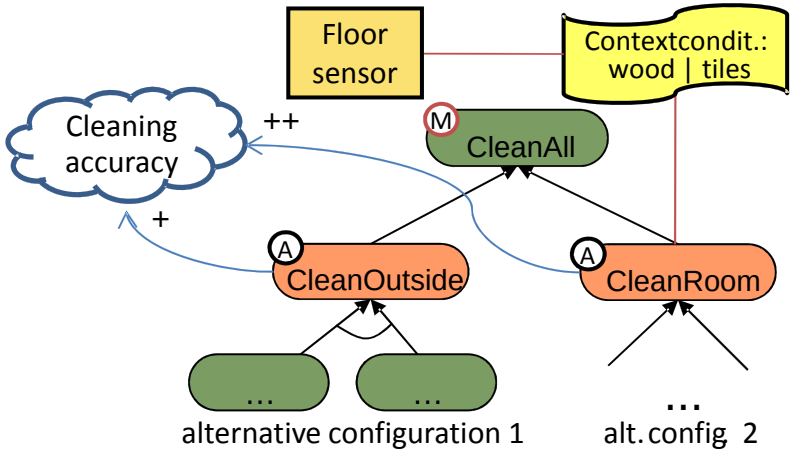
← Inhibition Relation

Sequence Relation



2) Correlation of the environment to goal achievement

3) Prevent failures



Publications

- M. Morandini, L. Penserini and A. Perini, Operational Semantics of Goal Models in Adaptive Agents, AAMAS'09, Budapest, Hungary, May 2009.
- M. Morandini, L. Penserini, and A. Perini. Towards Goal-Oriented Development of Self-Adaptive Systems. SEAMS at ICSE08, Leipzig, Germany, May 2008.
- M. Morandini, L. Penserini and A. Perini, Automated Mapping from Goal Models to Self-Adaptive Systems, In Proc. of the 23rd Conference on Automated Software Engineering (ASE08), L'Aquila, Italy, 2008

Future Work

- Experimental evaluation for the effectiveness
- Writing the PhD Thesis

Conclusion

Goal-orientation supports decision making in different contexts

- (LAW COMPLIANCE) goals represents the states that the "actions" induced by laws aim to achieve
- (DESIGN PATTERNS) goals represent designer objectives that will be met by reusing a pattern
- (ADAPTIVE AGENTS) goals drive agent decisions on which behavior to select, that is, they have an operational semantics.

Any Questions?