Policy Modeling and Compliance Verification in Enterprise Software Systems: a Survey

George Chatzikonstantinou, Kostas Kontogiannis
National Technical University of Athens

September 24, 2012
MESOCA’12, Riva del Garda, Trento, Italy
Goals of this survey

Survey a number of policy modeling and policy compliance verification techniques

Propose a corresponding basic classification for these

Outline advantages and disadvantages of each class of techniques
Ambiguity of the term "policy"

Many aspects and features of ESS are described by the term *policy*

It mainly depends on the context

Possible interpretations include:
- requirement
- process
- condition
Definition and Types of Policy

**Definition**

*Policy* is a rule that defines an intentional, expected or mandatory behavior or property of a system.

We focused on four policy types, mainly because of their importance as this is documented in the surveyed literature:

- Security Policies
- Business Process Policies
- Regulatory Policies
- Design Policies
Table of contents

1. Policy Modeling
   - Issues related to Policy Modeling
   - Classification of the techniques proposed

2. Policy Compliance Verification
   - Issues related to Policy Compliance Verification
   - Classification of the techniques proposed

3. Conclusion

4. Future Avenues of Research
Introduction

Policy Modeling

The process of creating an abstract and formal representation of a policy.

Limitations of natural language (NL):
- NL is inherently ambiguous
- Processing documents written in NL with CASE tools is not a trivial task

This leads to the necessity for defining formal or semi-formal languages/notations
The ideal modeling notation

Ideally a modeling notation should:
- not leave room for ambiguities
- be expressive enough to denote any policy required
- be easily understood by all stakeholders
- allow the use of CASE tools
The ideal modeling notation

Ideally a modeling notation should:

- not leave room for ambiguities
- be expressive enough to denote any policy required
- be easily understood by all stakeholders
- allow the use of CASE tools

There is always a trade-off between expressiveness and complexity
Classification of the techniques proposed

Classification

Policy Modeling Classification
  \arrow{Graphical Notations}
  \arrow{Formal Languages}
Classification

Policy Modeling Classification

Graphical Notations
- UML Profiles
- Sequence Charts
- Directed Graphs
- Agent Based

Formal Languages
Classification of the techniques proposed

Classification

Policy Modeling Classification

Graphical Notations
- UML Profiles
- Sequence Charts
- Directed Graphs
- Agent Based

Formal Languages
- Logic Based
- High Level
Classification of the techniques proposed

Graphical Notations - UML Profiles

Pros:

- the majority of IT experts are familiar with UML and the new notation can be easily adapted
- CASE tools for UML profiles

Cons:

- any deviation from the standard may lead to interoperability problems
Graphical Notations - UML Profiles - Examples

- SecureUML [Lodderstedt et al. 2002]
- UML4SOA [Bruni et al. 2009]
Graphical Notations - Sequence Charts

Pros:
- visualize the behavior of the system in terms of scenarios/anti-scenarios

Cons:
- policies must be formulated as sequences of messages exchanged between entities
Classification of the techniques proposed

Graphical Notations - Sequence Charts - Examples

- UML Profiles
- **Sequence Charts**
- Directed Graphs
- Agent Based

**TMSC [B. Sengupta et al. 2006]**
- MSC extension
- supports conditional/partial scenarios

**Property Sequence Charts [M. Aytili et al. 2006]**
- messages enriched with temporal properties
Graphical Notations - Directed Graphs

Pros :
- models can be easily used and understood from most stakeholders regardless of their scientific background
- can be used in model checking techniques

Cons :
- policies must be transformed into event-driven transitions between states
Graphical Notations - Directed Graphs - Examples

Visual Timed event Scenarios
[A. Alfonso et al. 2004]
- limited set of graphical elements
- model complex scenarios for real time systems

YAWL [W. van der Aalst et al. 2003]
- Petri-nets extension
Classification of the techniques proposed

Graphical Notations - Agent Based

Pros:
- assists in developing a deeper understanding of the system
- provides a more clear view of the environment the system operates in

Cons:
- lacks in describing the exact sequence of activities and the responsible actors for each activity
Graphical Notations - Agent Based - Examples

- **UML Profiles**
- **Sequence Charts**
- **Directed Graphs**
- **Agent Based**

**Active-i* [T. Xu et al. 2010]**
- combines i* with UML activity diagrams

**Agents with Commitments [A. Chopra et al. 2010]**
- extends Tropos modeling notation
Visual notations have the advantage of being easily understood by most stakeholders but they sometimes lack in formality.

When formality is more important than the ease of use, formal languages can be used.

These languages are mostly based on:
- mathematical logic
- high level programming languages
Formal Languages - Logic Based

Pros:
- enable processing and reasoning
- can be used to express semantics of abstract representations

Cons:
- high complexity
- a mathematical background is required
Classification of the techniques proposed

Graphical Notations - Logic Based - Examples

 MFOTL [D. Basin et al. 2010]
  - based on temporal logic
  - used to describe complex security policies

 FLAVOR [R. Thion et al. 2010]
  - based on deontic logic
  - express obligations and permissions
Formal Languages - High Level Languages

Pros:
- easy for software professionals to learn and use
- do not have the complexity of logic based ones

Cons:
- cannot be used from all stakeholders (e.g. business analysts)
Graphical Notations - Logic Based - High Level Languages

- PROPOLS [J. Yu et al. 2006]
  - encoded in OWL
  - specify policies for BPEL schemas

- Rei [L. Kagal et al. 2003]
  - it is implemented in Prolog
  - policies in systems that change dynamically
  - priorities among policies (resolution of conflicts)
Introduction

Policy Compliance Verification

The process of assessing whether a software system satisfies a certain policy or not.

Applications of compliance verification

- during development to assist design decision making
- during runtime to control system maintenance and evolution
- in combination with policy enforcement techniques to build self-adaptive systems

Focus on automatic or semi-automatic methods
Classification of the techniques proposed

Classification

- Model Checkers
- Probabilistic Model Checkers
- Theorem Provers
Model Checkers

Pros:
- fully automated verification process
- generate counterexample

Cons:
- can not easily apply to runtime analysis
- demanding in terms of execution time and memory
Model Checkers - Examples

Osman et al. 2006
- runtime verification
- uses local model checking techniques
Probabilistic Model Checkers - Examples

A. Filieri et al. 2011
- evaluate the satisfaction of reliability requirements
- runtime verification
- system model and properties are transformed in symbolic expressions at design-time
Theorem Provers

Pros:
- same notation for the system and the properties
- generate the sequence of steps of the proof

Cons:
- strong mathematical background is required
- low flexibility for fully automated provers
Conclusion

Traditional model checking techniques (MCTs) do not manage to fulfil the execution time constraints imposed by runtime analysis.

Modifications must be made to traditional MCTs (e.g. local model checking)

While theorem provers solve the problem of state explosion they also have limitations

There are approaches that try to combine MCTs with theorem provers [e.g. W. Kong 2005]
Future Avenues of Research

Combine reverse engineering with monitoring techniques to verify that the system at hand complies with a set of policies.

Tracing events against compliance constraints to identify deviations from service level agreements.

Tracing actual resource usage patterns so that the system can be re-configured to meet dynamically changing needs.
Acknowledgements

This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program ”Education and Lifelong Learning” of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.