IEEE International Symposium on Policies for
Distributed Systems and Networks (POLICY 2011)

OSeml

Tool-Supported Refinement of High-Level Requirements and Constraints into Low-Level Policies

Outline:

- Automated Policy Refinement
- Background
- Application
- Patterns
- Conclusion

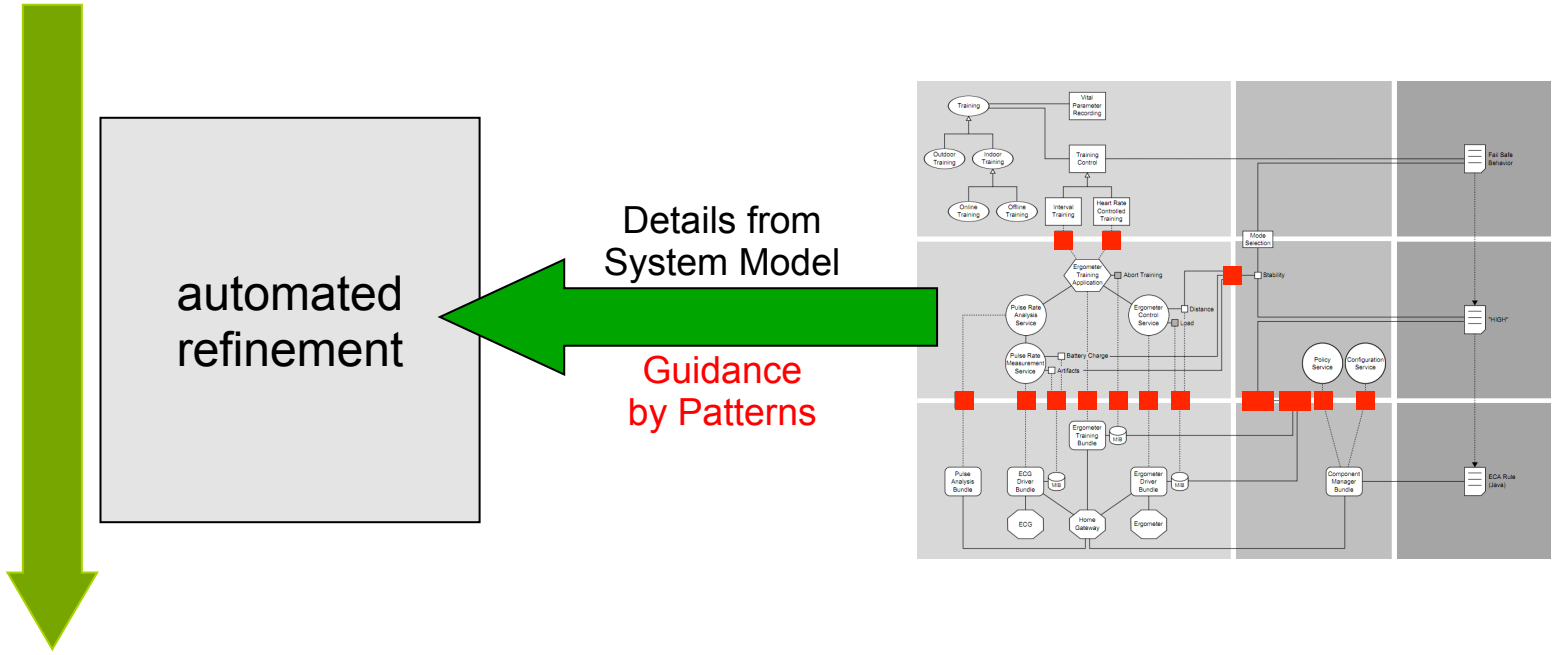
TU Dortmund University
O. Dohndorf, J. Krüger, H. Krumm

MATERNA
C. Fiehe, A. Litvina, I. Lück, F.-J. Stewing

Automated policy refinement

- Principle:

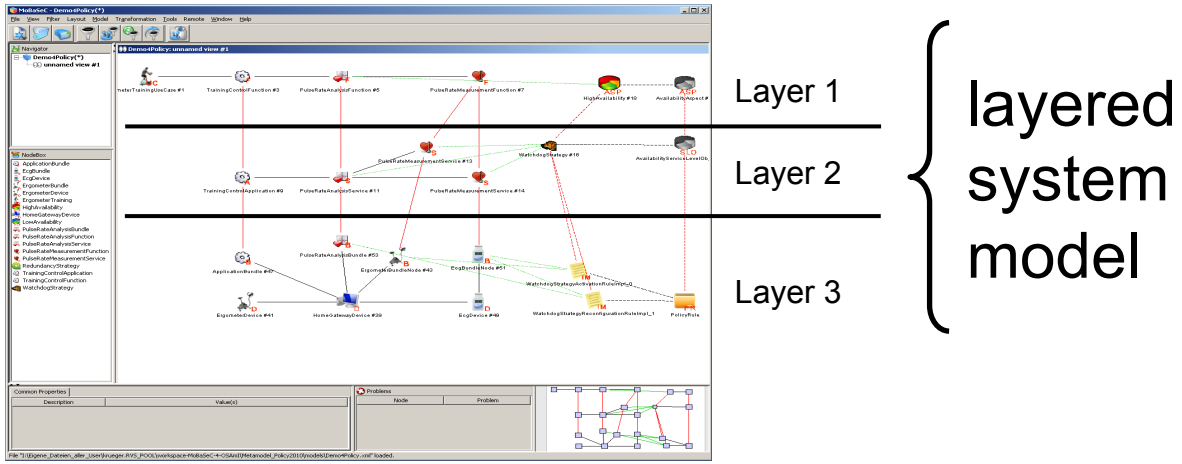
Abstract Policies: Constraints



Detailed Policies: Executable ECA-Rules

Background

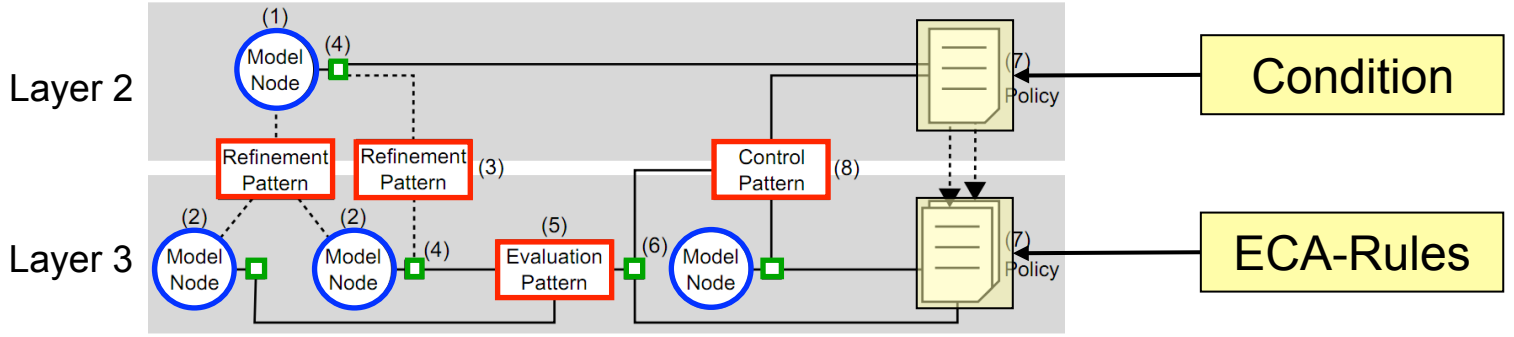
- Tool **MoBaSeC**



- Model-Based Management (MBM)
  - supported by MoBaSeC
- Project OSAmI
  - AAL domain
  - not necessarily technicians
  - resource-constraint devices

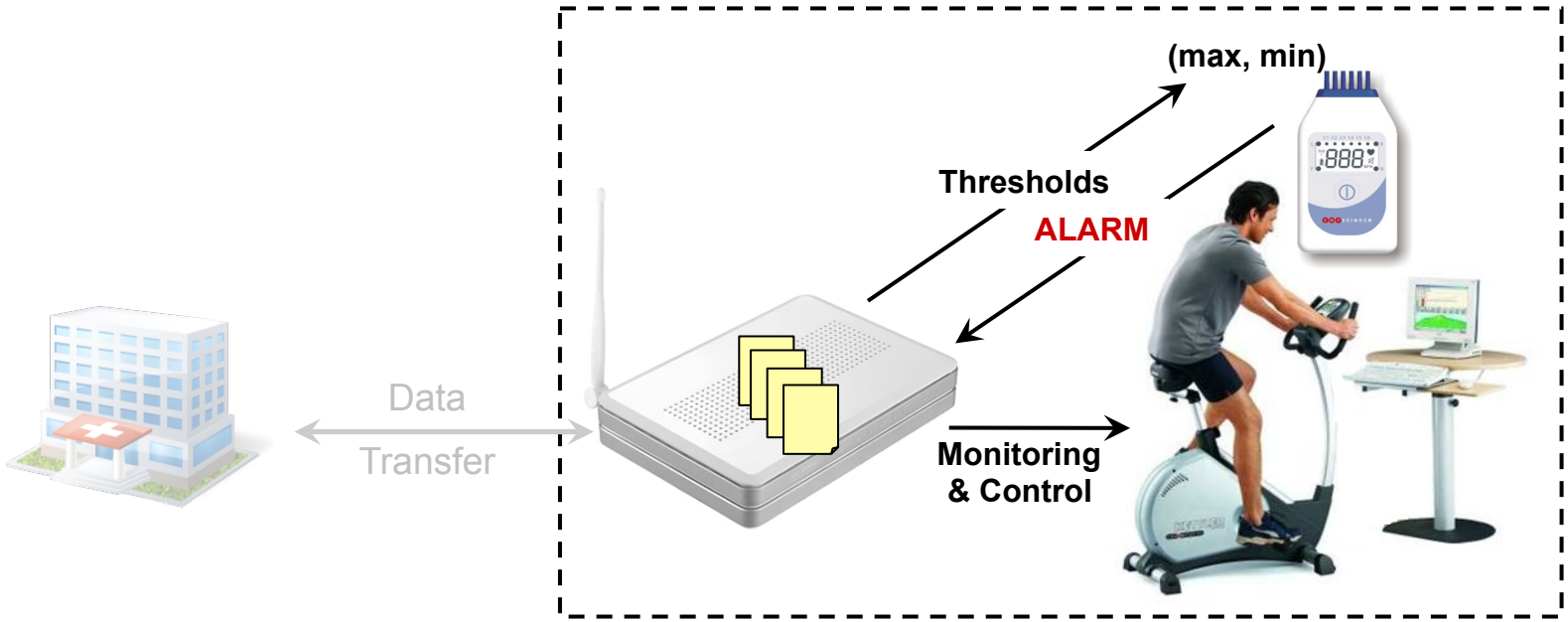
Policy refinement

- Basis: the 3 layer system model
- Modeled by means of UML-like object diagram
- Model consists of
  - Model **nodes** representing the system elements
  - Management **variables** being assigned to the nodes
  - **Associations** between model nodes
- Policy refinement process
  - exploits model structure; is governed by general and domain-specific **pattern** instances
- Pattern types: refinement, evaluation, control



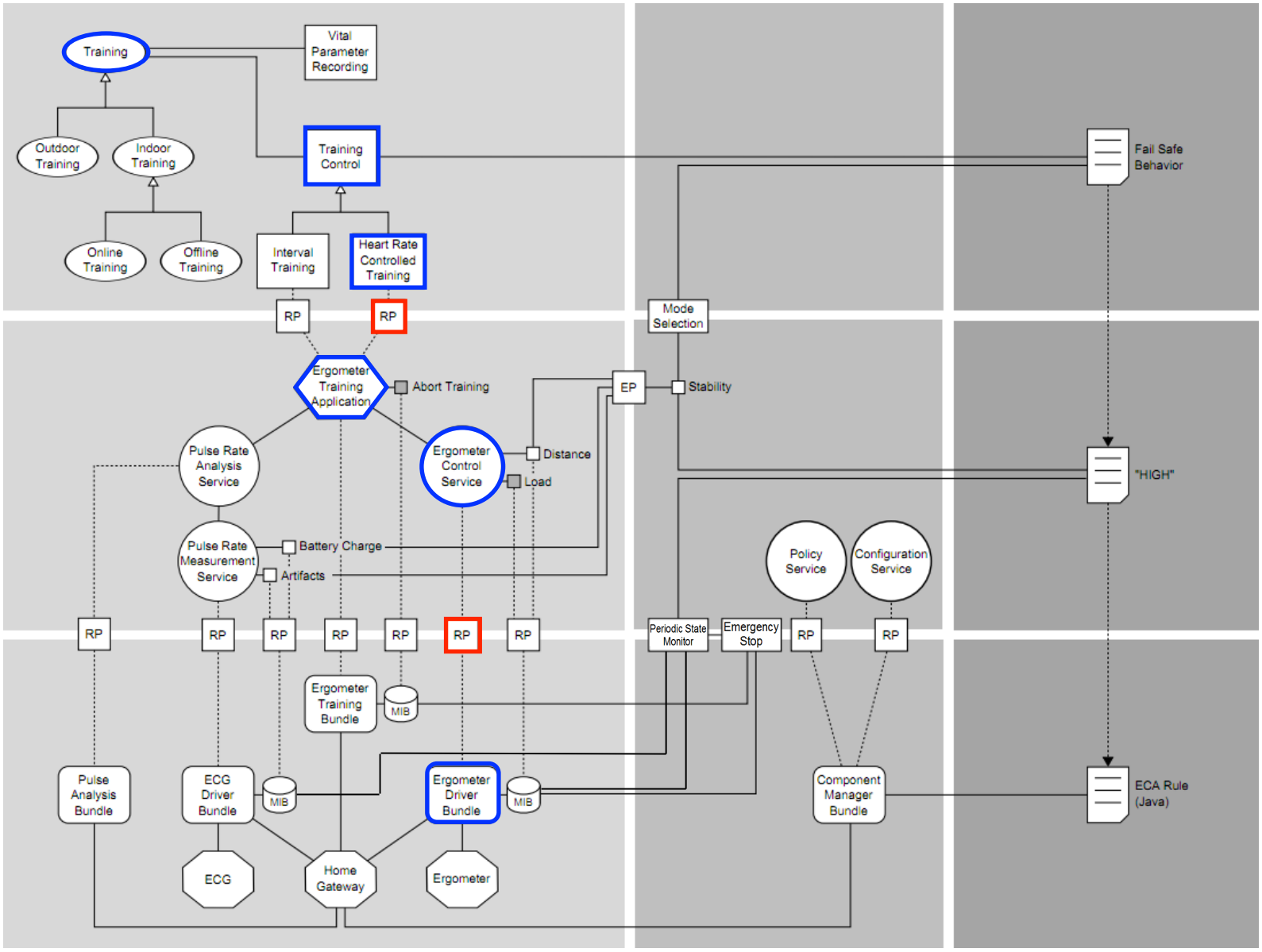
Field of application

- Used to generate policies for the use in the OSAmI project
- Scenario:



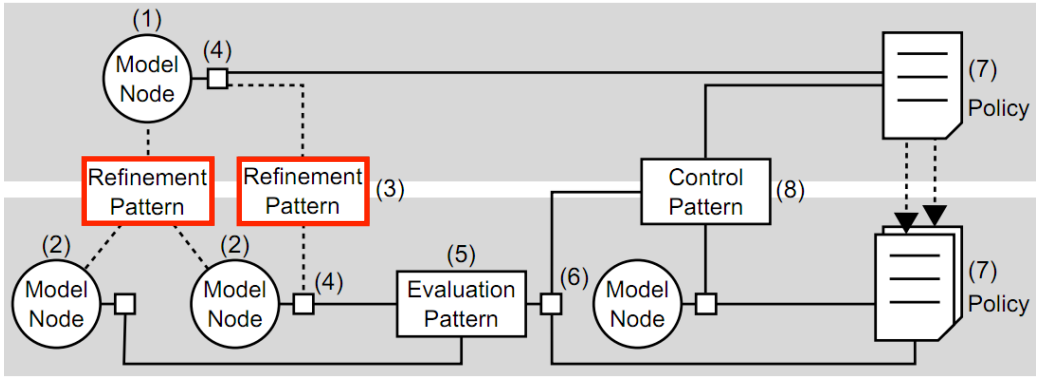
- Policies to control the system, especially to
  - keep it stable & running
  - ensure the safety of the patient

Application: Resulting model *(simplified)*

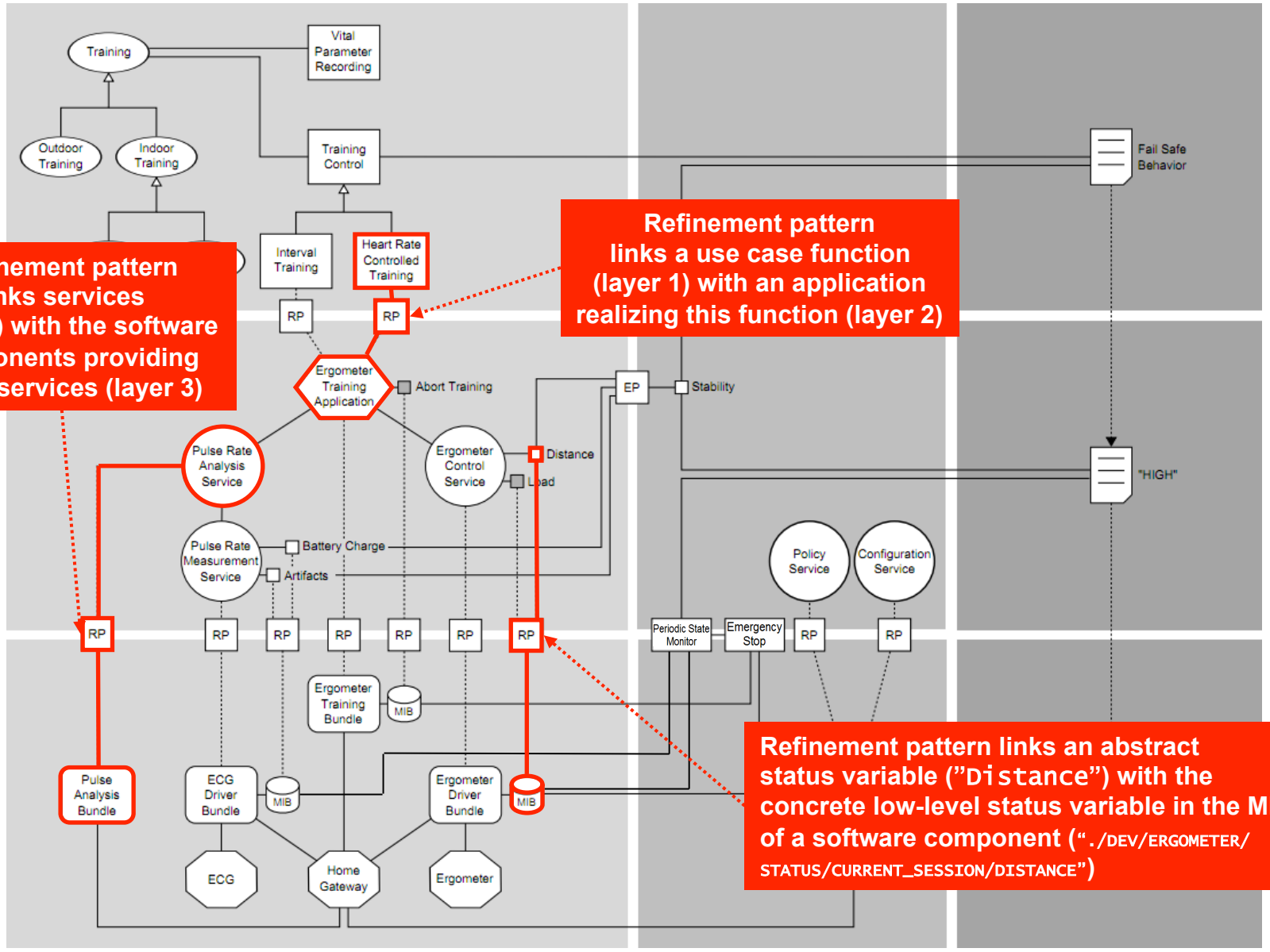


Refinement Patterns: Purpose

- Specify refinement relations between model elements of *adjacent layers*
- RPs between system elements illustrate *implementation strategies*
- RPs between management variables of adjacent layers depict the *mappings* between the *abstract and the more concrete variables*

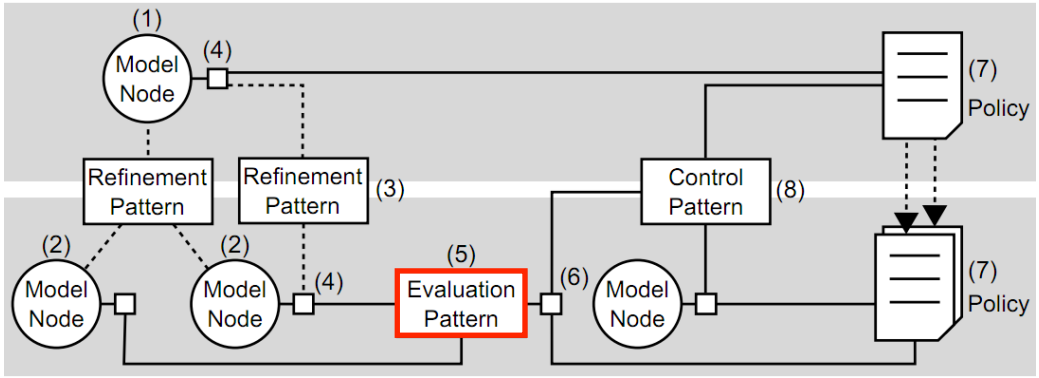


Refinement Patterns: Example

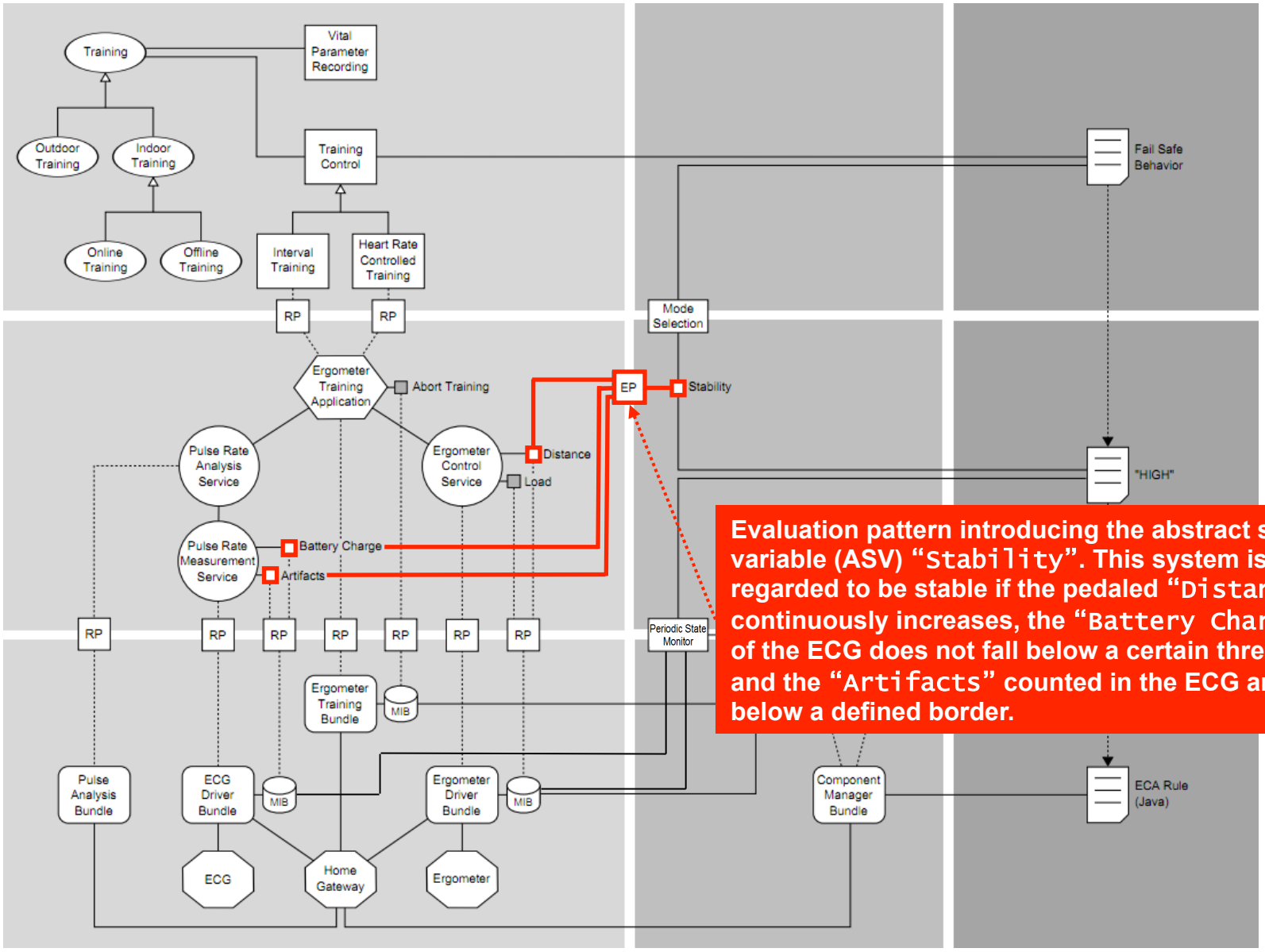


Evaluation Patterns: Purpose

- Introducing *abstract status variables* (ASV)
- ASVs allow to define *domain-specific* quantities
- ASVs aggregate the values of a set of status variables according to a *specific point of view*
- EPs define the corresponding (arithmetic) *expressions*



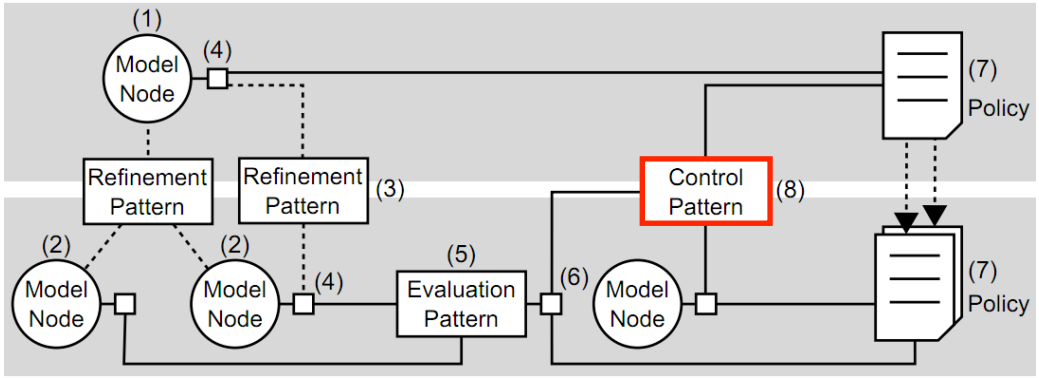
Evaluation Patterns: Example



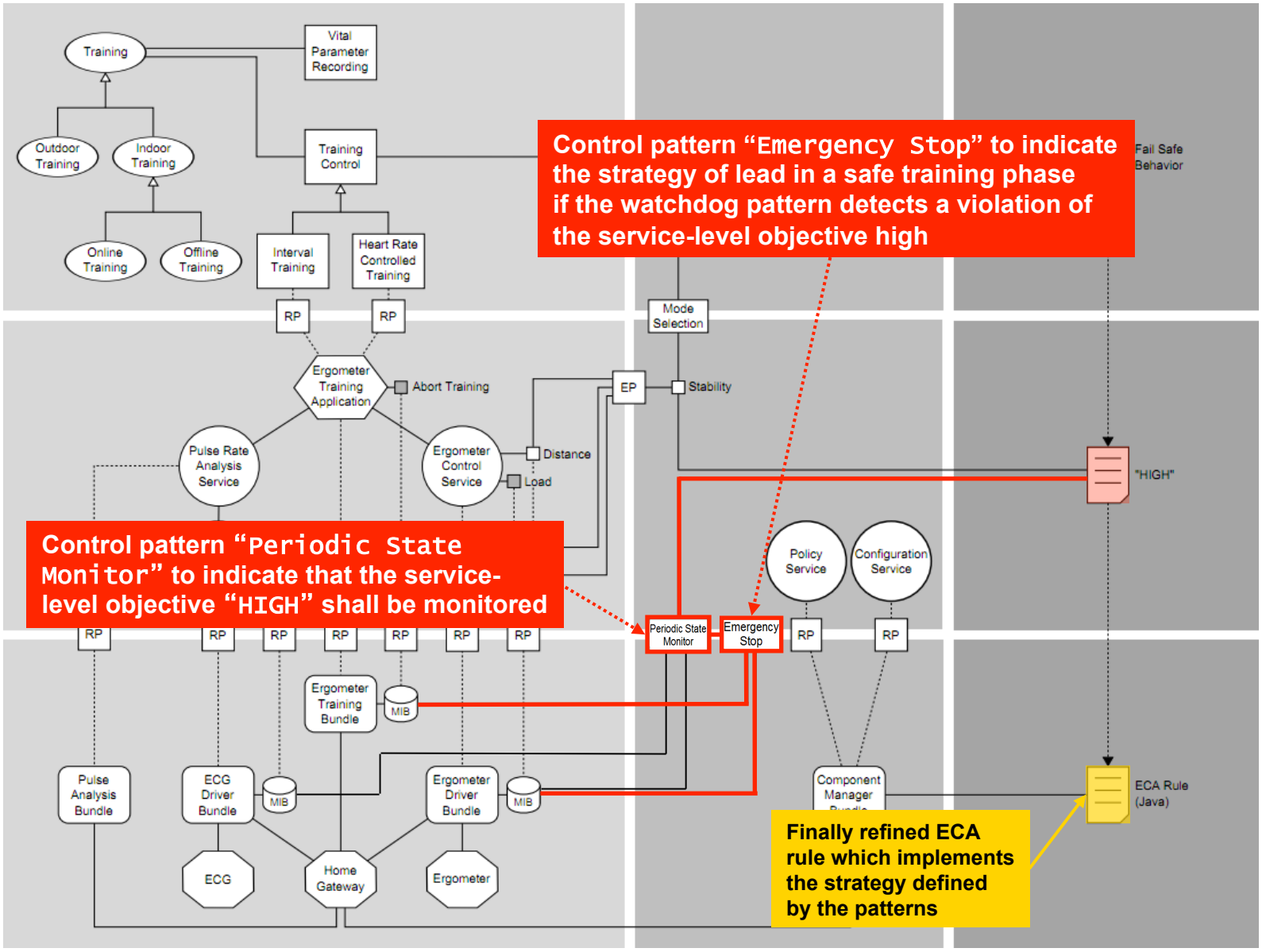
Evaluation pattern introducing the abstract status variable (ASV) "`Stability`". This system is regarded to be stable if the pedaled "`Distance`" continuously increases, the "`Battery Charge`" of the ECG does not fall below a certain threshold, and the "`Artifacts`" counted in the ECG are below a defined border.

Control Patterns: Purpose

- Specify how higher-level policies are to be enforced by lower-level ones
- CPs specify the detailed measures to be applied by
  - mapping of declarative abstract requirements to sets of more concrete conditions and objectives
  - the definition of the behavior of control components by relating declarative objectives with imperative enforcement mechanism



Control Patterns: Example



Conclusion

- Automated policy refinement
  - from abstract policy constraints to executable ECA rules
  - detail enrichment from model, guidance by patterns
- Pattern types: refinement, evaluation, control
- Tool-support for modeling & refinement
  - Tool MoBaSeC
- Application within the project OSAmI
  - AAL domain
  - Challenge: resource-restricted devices
  - Challenge: not necessarily technicians for planning

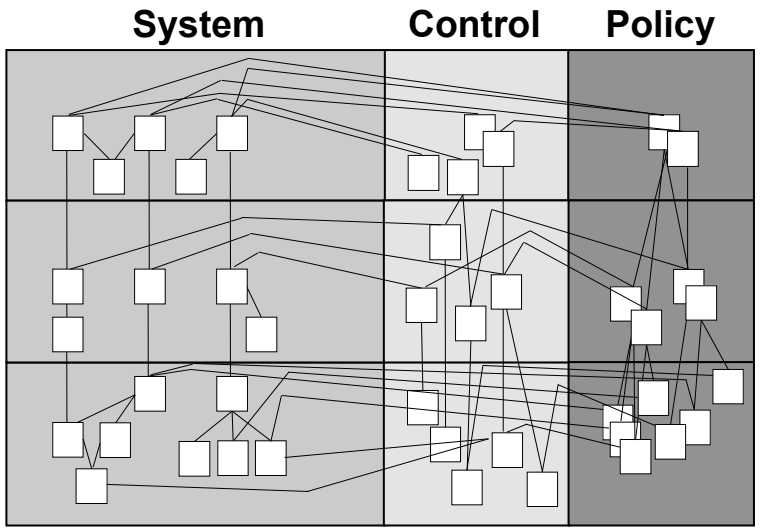
Thanks for the attention!

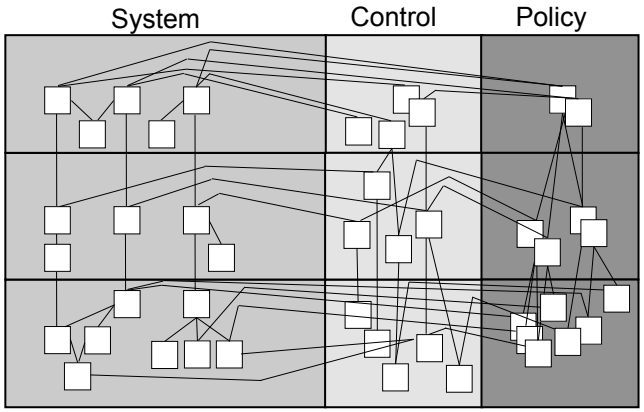
Questions?

Refinement Patterns: Example

- For example, the relation between a service and a software component providing it
- For example, a service level parameter is mapped to the variable in the MIB of the providing component

… ab hier:
ausgeblendete Folien!


System
Control
Policy


System
Control
Policy

Motivation

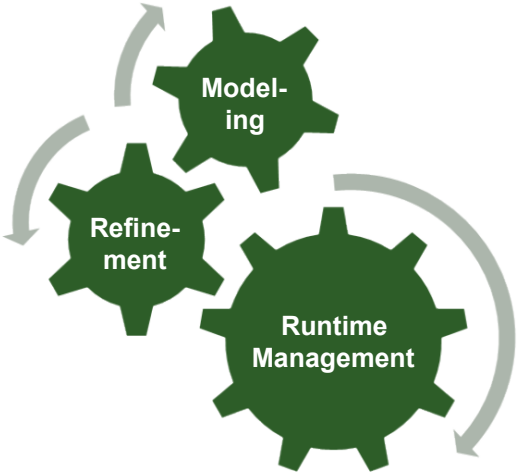
- Use-case context: mobile devices in healthcare application scenarios
  - Resource restrictions of used devices
  - Spontaneously changing conditions
  - Sometimes critical exceptions
- Furthermore: users (health professionals) typically not familiar with technical system management

Basics of our work

- **Model-based** management
- **Layered** System Model
- Model-based, automated policy **refinement**
- Policy refinement supported by **patterns**:
  - **Refinement** patterns
  - **Evaluation** patterns
  - **Control** patterns
- **Tool** Support (MoBaSeC)
- Low-level management rules by means of **ECA**

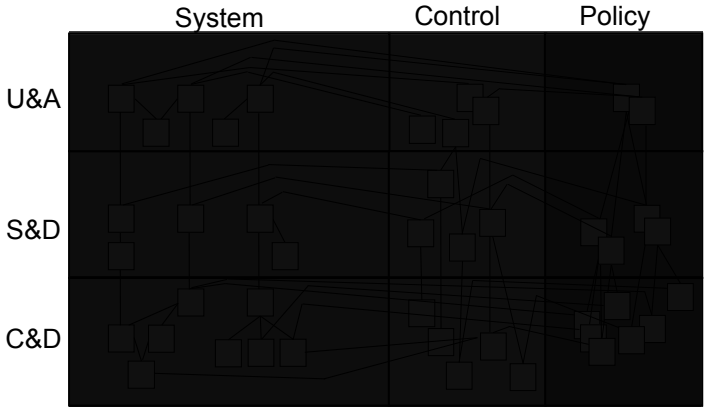
Model-based management

- Combination of management policies and policy hierarchies with a layered system model
- Management policies:
  - Additional control level above the programming code
  - Used for configuration, adaptation, correction
- Management phases:
  - Design phase:
    - System to be managed is modeled
    - Abstract high-level polices are defined manually
    - The refinement into technical low-level policies is performed automatically (supported by a tool)
  - Runtime phase:
    - The refined low-level policies are efficiently enforced



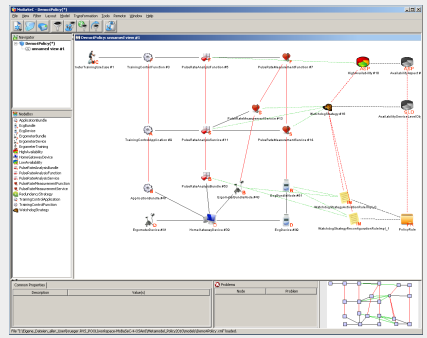
Layered System Model

- System is modeled on *3 layers of different abstraction*
- Each layer is *self-contained & independent* → 3 models of the same system, differing in the degree of abstraction
- Layers:
  - **Use Cases & Aspects** (U&A): barely technical details, instead e.g. non-functional requirements & quality criteria are modeled
  - **Services & Domains** (S&D):service-oriented point of view; clients, services and dependency relations are modeled and assigned to domains
  - **Components & Devices** (C&D): elements existing at runtime (e.g., software components and hardware devices) are modeled
- Refinement relations to connect adjacent layers of the model

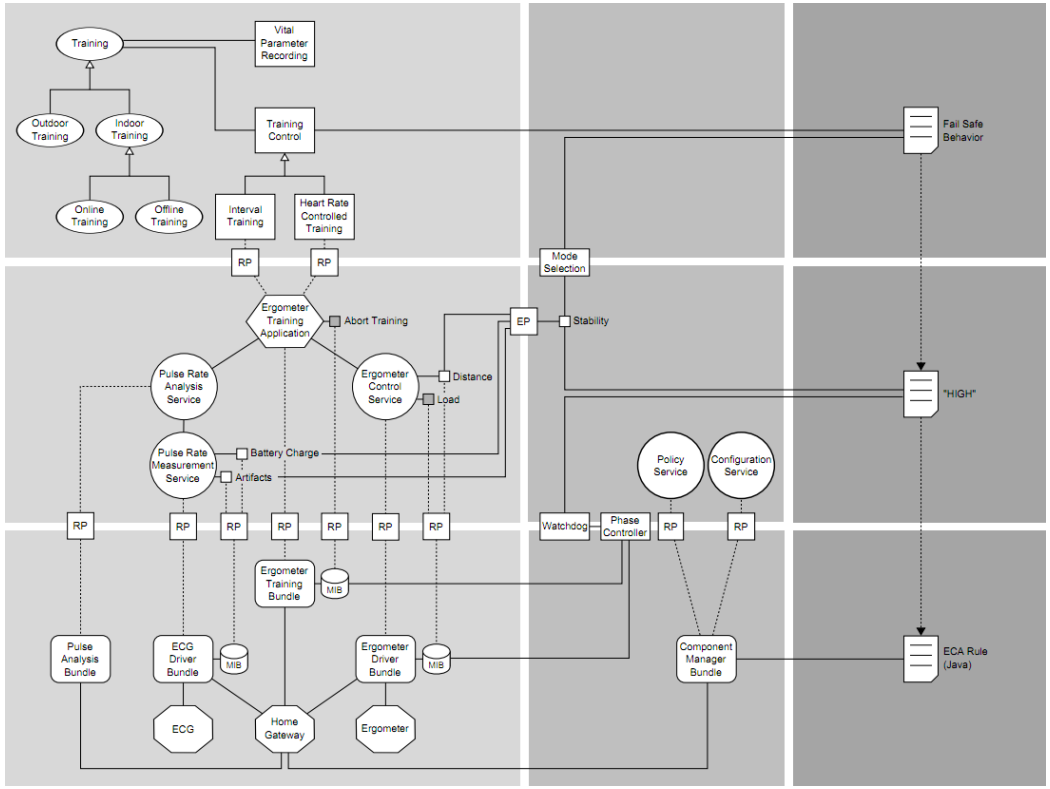


Tool „MoBaSeC“

- “Model Based Service Configuration”
- Java-based; Metamodels also programmable by means of Java
- Supports user in the design phase by
  - providing a graphical frontend which allows to model a system (by easy-to-use drag’n’drop functions)
  - providing backend functions to automatically traverse & evaluate a model, and to refine the low-level policies based on this
- Generated low-level policies are
  - automatically compiled into efficient executable Java bytecode
  - automatically distributed to the policy storages where they are required



Policy tool „**MoBaSeC**“

Example

- ToDo
- Not sure if an example does make sense in this presentation?