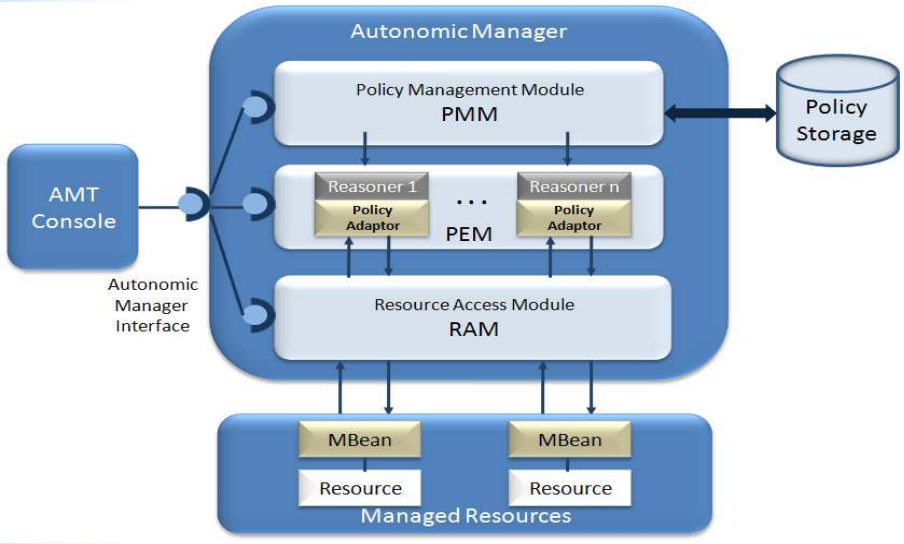
Rule Engine Based Lightweight Framework for Adaptive and Autonomic Computing

Jakub Adamczyk, Rafal Adamczyk, Marcin Jarząb,
Krzysztof Zieliński
Department of Computer Science, AGH-UST
Kraków, Poland

Agenda

- Architecture of Autonomic Management Toolkit,
- Adaptation loop activities,
- Experiments,
- Summary.

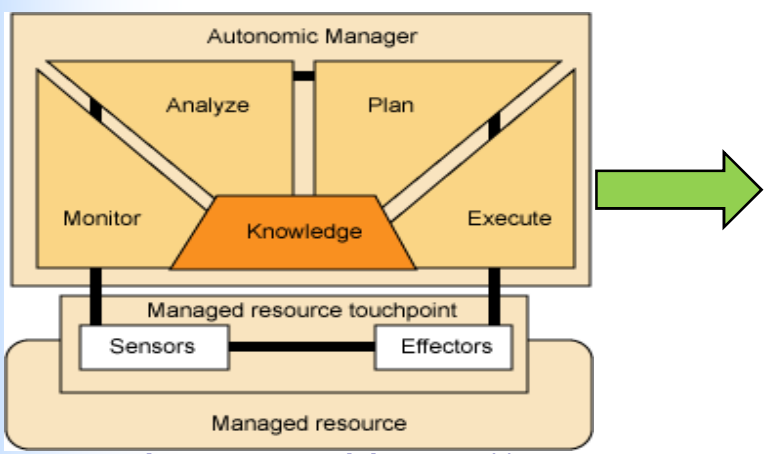
Architecture of AMT



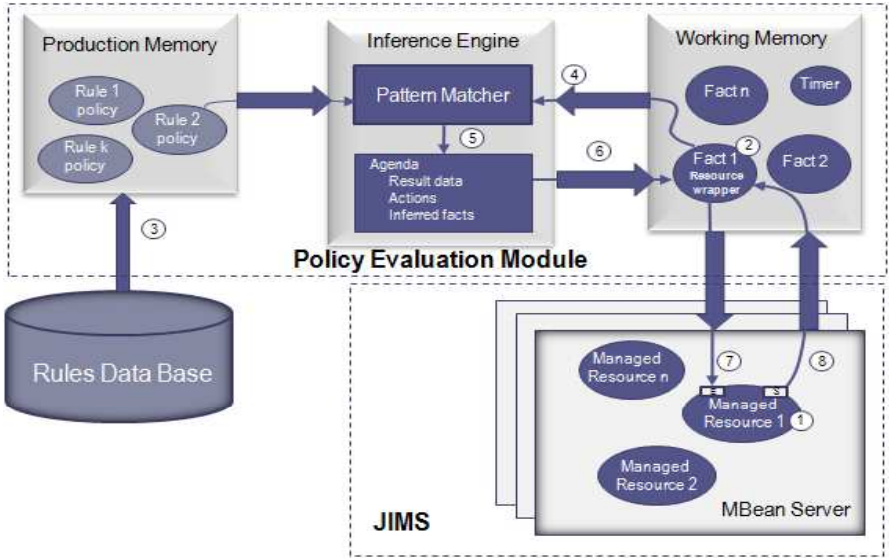
- **PMM** - policies defined by system administrator deployed to AMT,
- **PEM** - policies are obtained from storage, instantiated by a given reasoner and evaluated. The key point of this subsystem is an interface that supports interoperability with different reasoners,
- **RAM** - defines resources and manages interactions compatible with AMT specification.

- The central element of ***AMT*** is a lightweight ***Autonomic Manager*** with a ***Rule Engine*** as a decision making module:
  - Engines which implement JSR 94 (Java Rule Engine API) are supported (Drools, Jess),
  - Also compatible with IBM PMAC (Policy Management for Autonomic Computing).
- ***Resource Access Module*** is integrated with ***JIMS*** (JMX-based Infrastructure Monitoring System) framework:
  - Diversity of managed resources (Linux, Solaris, J2EE application servers, Grid Engines, networking hardware),
  - Many JMX services (notification, timer) and connectors (SOAP, RMI, SNMP).

AMT adaptation loop activities

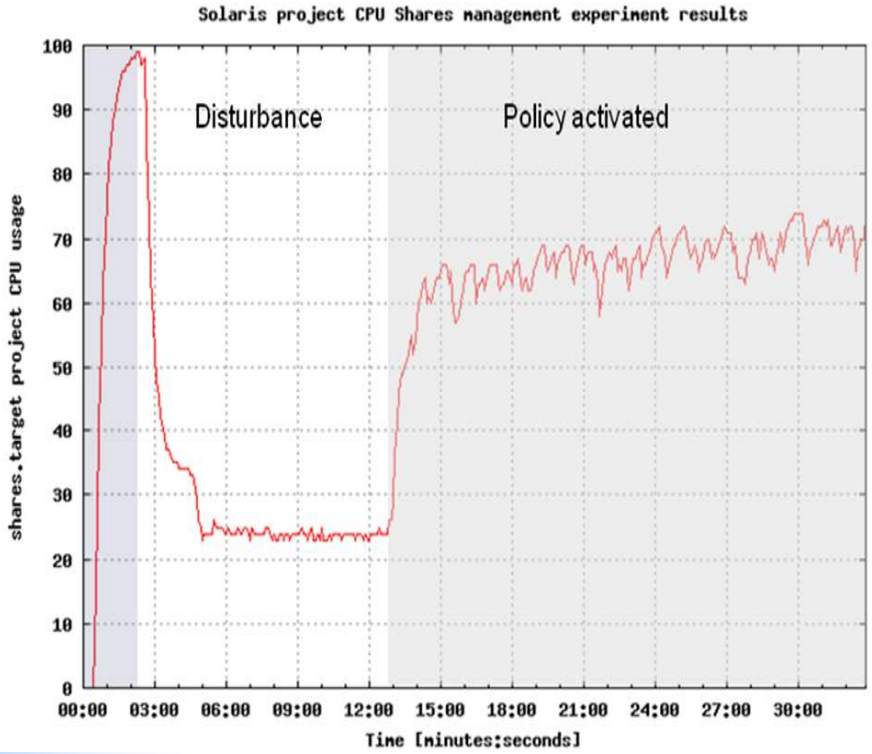


Autonomic Systems Model *proposed by Horn, P. Autonomic Computing: IBM's Perspective on the State of Information Technology, October 15, 2001*



1. **Managed Resources (MRs) are instantiated as JMX MBeans.**
2. **Resource Wrappers of MRs which play the role of facts, are constructed and inserted into the Working Memory.**
3. **Production Rules representing policies are loaded into the Production Memory. At this point, the Inference Engine is also started.**
4. **The Pattern Matching algorithm is performed on all rules in the Production Memory and facts present in the Working Memory.**
5. **All rules that are evaluated as true are added to the *Agenda* to be performed.**
6. **Action is performed on the MR representation in the Working Memory.**
7. **The action is forwarded to MR via the Resource Wrapper and enforced with effectors.**
8. **MR parameter changes accessed by sensors are communicated to the Resource Wrapper, which in turn triggers execution of step 4.**

Experiments



GOAL: *Given percentage of CPU time would always be available in conditions of constant load* for a specific Workload – for instance, a given Solaris *Project* should be guaranteed 70% of CPU time when other active workloads (disturbances) are also running.

Proportional regulator

$S_w^{t+1} = S_w^t + K_p * e(t)$,

where $e(t) = U_w^t - U_w$

Control algorithm implemented with Drools rule-based policies.

Summary

- AMT system exploits the potential of Rule Engine-based computing which is a very attractive solution for policy-driven autonomic computing systems,
- AMT enables to integrate various rule engines concurrently
  - Jess supports Fuzzy Logic – if needed,
- Rule Engine is sophisticated software module which supports scalable pattern matching algorithms (RETE),
- Thus may be used for a large number of facts and rules constituting a representation of knowledge,
- What's next?
  - Decision trees (self-healing),Queuing networks, Fuzzy logic, used for adaptive management of SOA services!