StarMX: A Framework for Developing Self-Managing Java-based Systems

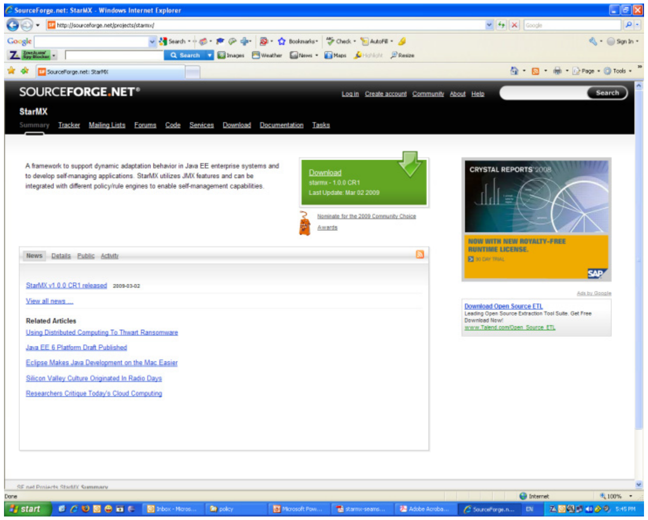
Reza Asadollahi, Mazeiar Salehie, and Ladan Tahvildari
Software Technologies Applied Research (STAR) Group
Electrical and Computer Engineering Department
University of Waterloo, Canada
{rasadoll, msalehie, ltahvild}@uwaterloo.ca

Motivation

- Realizing self-managing solutions is challenging
  - Developmental issues: architectural elements, the interaction among them…
  - Runtime concerns: performance, runtime services…
- Reusable software frameworks are helpful
  - Saving development effort
- New approach is needed:
  - Proposed solutions have limitations
  - Enabling technologies have been improved

StarMX Framework

- A generic open-source framework based on standards and well-established principles to address self-managing concerns in the Java domain
- Hosted at *sourceforge.net*
  - http://sourceforge.net/projects/starmx
  - More than 260 downloads so far



A framework to support dynamic adaptation behavior in Java EE enterprise systems and to develop self-managing applications. StarMX utilizes JMX features and can be integrated with different policy/rule engines to enable self-management capabilities.

Download
starmx - 1.0.0 CR1
Last Update: Mar 02 2009

Nominate for the 2009 Community Choice Awards

StarMX v1.0.0 CR1 released 2009-03-02

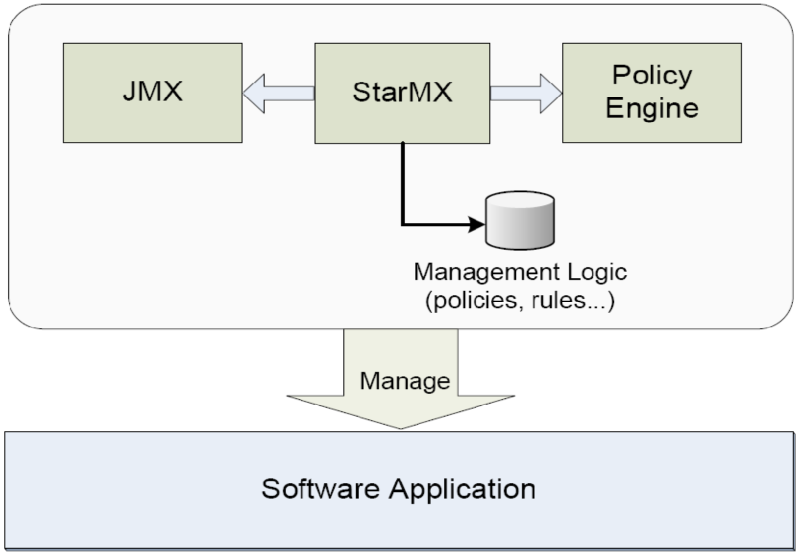
View all news ...

Related Articles

Using Distributed Computing To Thwart Ransomware
Java EE 6 Platform Draft Published
Eclipse Makes Java Development on the Mac Easier
Silicon Valley Culture Originated In Radio Days
Researchers Critique Today's Cloud Computing

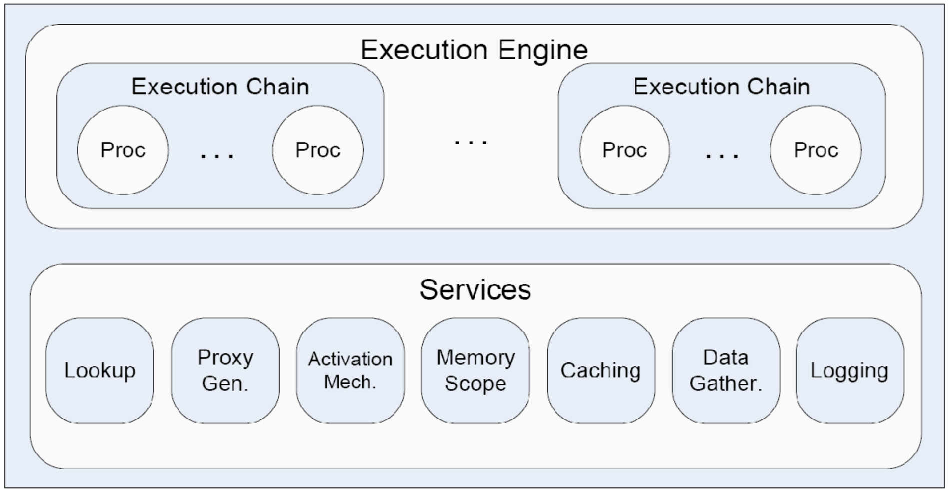
Enabling Technologies

- Java Management Extensions (JMX)
- Policy engines (e.g. IBM ABLE, Apache Imperius…)
  - Simple integration with different policy/rule engines



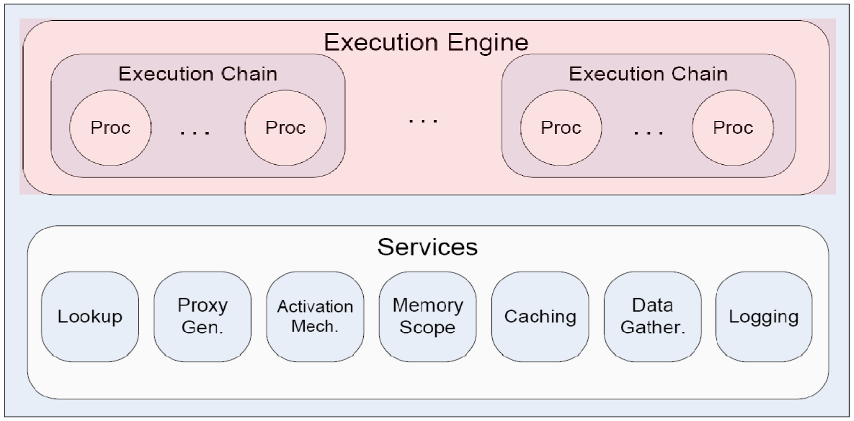
StarMX Architecture Overview

- Architectural characteristics:
  - Capturing common tasks in the development
  - Providing required features and services
  - Providing enough flexibility in designing self-managing requirements
- High-level view of the architecture:

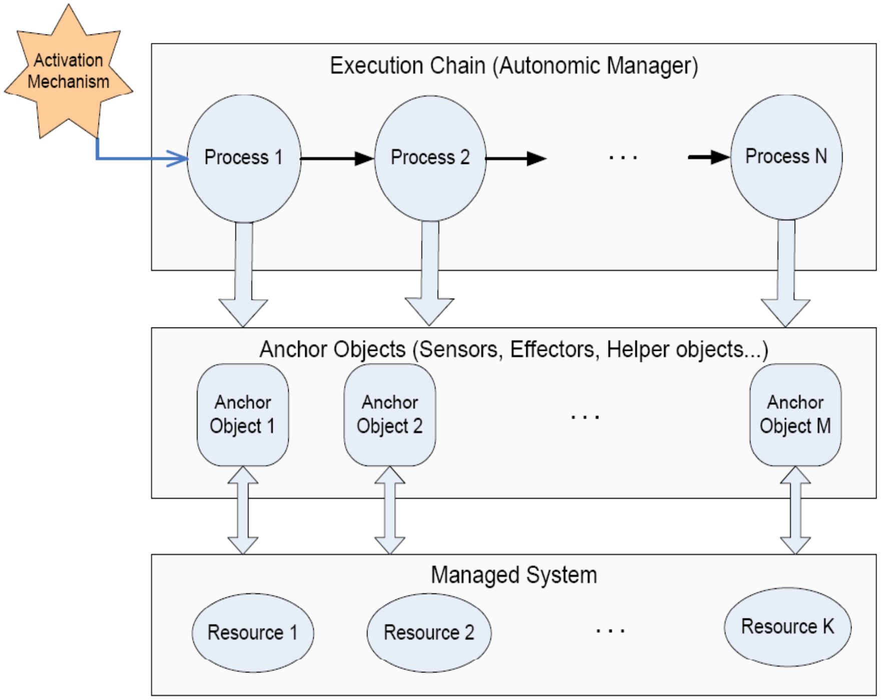


Execution Engine

- Automates management operations
- Uses the services provided in the *Service Layer*
- Key components: *Execution Chain* and *Process*
- *Process* as the building block of *Execution Chain*



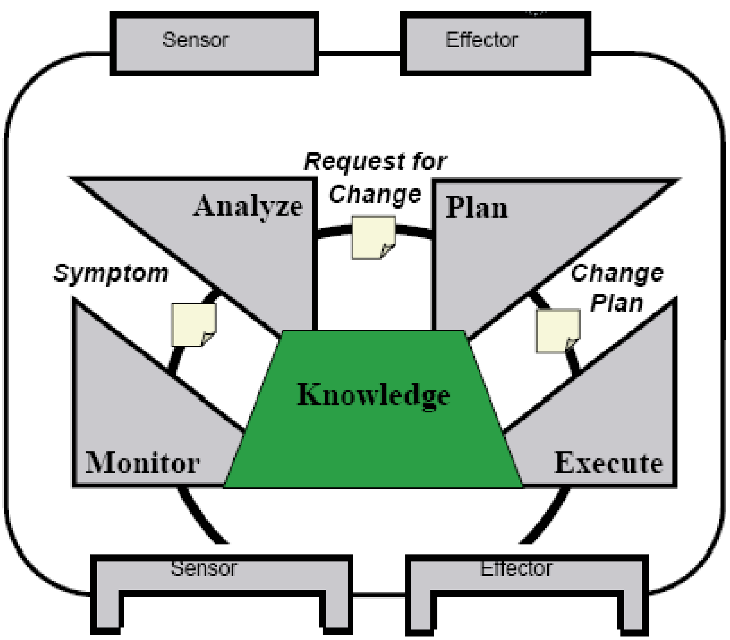
Execution Chain

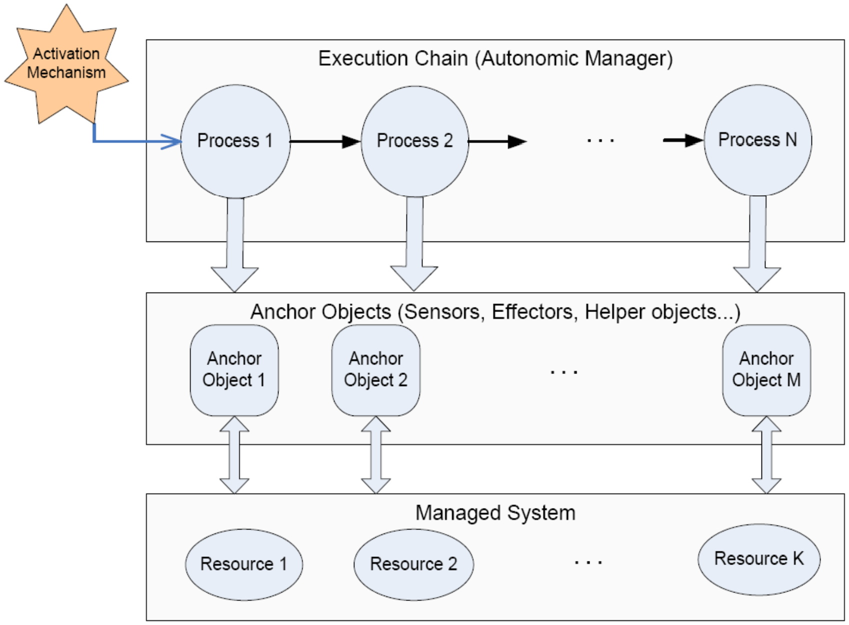


- □ Process implementation:
  - Using a policy language
  - Using the Java language
- □ Anchor objects:
  - JMX MBeans or MXBeans
  - Simple Java objects
- □ Activation mechanisms:
  - Timer-based
  - Event-based

Execution Chain Cont.

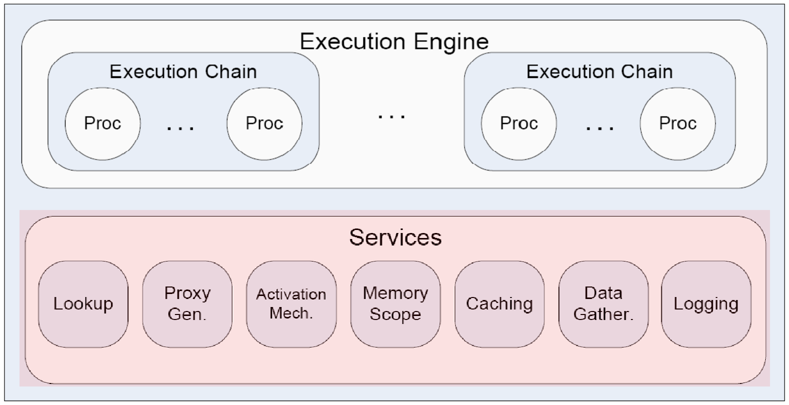
- Execution Chain as Autonomic Manager
- Process may represent one or more MAPE functions
- High flexibility in designing autonomic managers





Service Layer

- Lookup
  - provides access mechanisms to the anchor objects
- Proxy Generation
  - creates a proxy object dynamically if the anchor object is an MBean



- Activation Mechanism
  - defines the techniques for triggering execution chains
  - timer-based and event-based
- Caching
  - improves performance of the *lookup* service by holding references to previously accessed anchor objects

Services Layer Cont.

- Memory Scopes
  - repositories to store or exchange data among processes:
    - *StarMXScope*: shared among all processes
    - *PolicyScope*: private scope to a process
    - *ExecutionScope*: shared among all processes in one chain
- Data Gathering
  - collects statistical data about the execution of each process or execution chain
- Logging
  - a facility to keep records of the events in the framework

Development Steps Using StarMX

1. Define self-managing requirements
   - Based on non-functional requirements and SLA
2. Design and instrument sensors and effectors
3. Develop management logic
   - Implementing policy-based or Java-based processes
4. Configure StarMX in its xml file
   - Anchor objects and their lookup information
   - Closed loops via *execution chains* and *processes*
5. Deploy and run
   - Local or remote

Evaluation

□ To assess the suitability and fitness of the framework in a relatively complex situation

□ To evaluate the completeness of its feature set

□ To analyze its performance overhead

Case Study

- CallController2 (CC2), a VoIP system
- CC2 is deployed on Mobicents, an open source platform certified for JSLEE
- CC2 services include:
  - *Regular VoIP calls*
  - *Call forwarding*
  - *Call blocking*
  - *Voice mail*
- Lines of Code: 171K

Case Study Cont.

- Goal: to satisfy quality requirements at varying load conditions via enabling/disabling services
- 3 different user levels: bronze, silver, and gold
- 3 MBeans as sensors and effectors
- 30 policies to define self-managing requirements
- Load scenarios: Low and High
- Server: Win Server 2003 x64 SP2, Intel Core 2 Quad Q6700 2.66 GHz, 8 GB RAM, Ethernet 100 Mbps

Autonomic Capabilities

Criteria	**StarMX capability**
Degree of Autonomy	Closed loop
Control Scope	Multiple resources
Self-* properties support	All self-* properties
Management logic expression	Declarative and programmatic
MAPE loop construction	Flexible architecture
Monitoring technique	Timer-based and event-based
Data communication facility	Memory scopes
Remote management	Supported
Applicable environment	Any Java-based system
Managing non-Java systems	Via WebServices or JNI -based anchor objects
Runtime updating management logic	*Under construction*

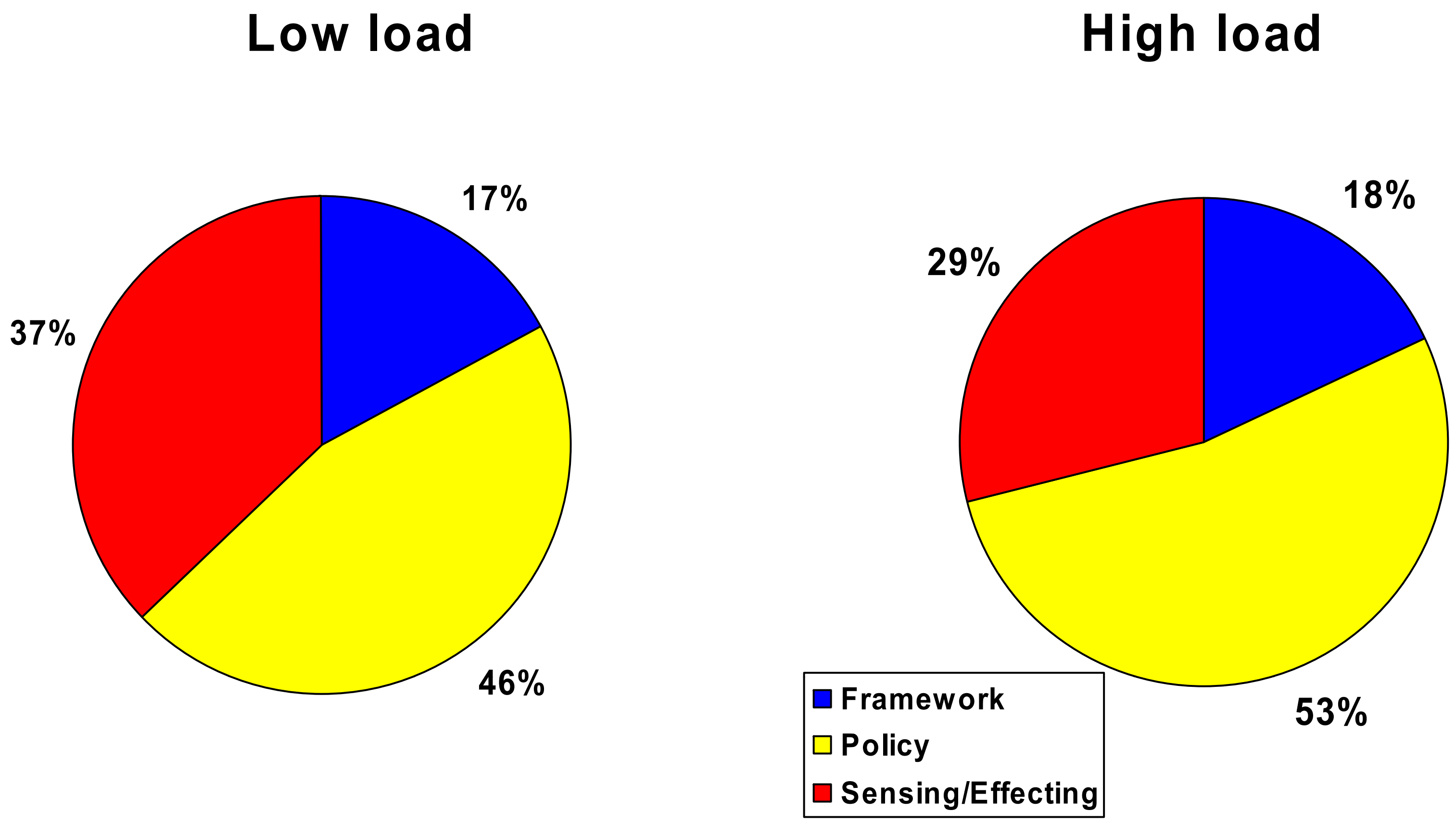
Performance Analysis

- StarMX performance is good if its impact on the system performance is negligible
- Total adaptation cost is comprised of:
  - Anchor object execution cost
  - MBean proxy execution cost
  - Process execution cost
  - **StarMX Framework execution cost**
- Local and remote deployment models have different impacts on performance

Performance Analysis Results

Performance criteria	**Low load**	**High load**
Load test time	732 (sec)	850 (sec)
No. of executed policies	1191	1472
Total adaptation time	2.8183	4.7152
Total policy execution time	2.3299	3.8645
Total sensing/effecting time	1.0515	1.3617
Avg policy execution time	2.3663 E-3	3.2032 E-3
Avg framework cost per policy	**0.41 E-3**	**0.578 E-3**
Adaptation proportion to total time	0.38 %	0.55 %

Adaptation Cost Proportions



Related Work

- Accord {H. Liu *et al*}
  - A framework to define autonomic elements and their composition
- Adaptive Server Framework {I. Gorton *et al*}
  - A framework for J2EE systems based on separation of concerns principle
- Autonomic Management Toolkit {J. Adamczyk *et al*}
  - A rule-engine based approach for adaptation in Java systems
- J3 Process {J. White *et al*}
  - A model-driven approach for fine-grained adaptation in EJB components
- Rainbow {D. Garlan *et al*}
  - An architecture-based self-adaptation framework

Conclusion and Future Works

- StarMX facilitates addressing self-managing requirements in different computing systems
- It performs well in running environments and provides the required runtime services
- It will be possible to manage the framework and its properties dynamically soon
- Supporting Web Services Distributed Management (WSDM) standard is a planned work for future