Metamodeling Autonomic System Management Policies
Ongoing Works

Benoît Combemale, Laurent Broto, Alain Tchana and Daniel Hagimont

Institut de Recherche en Informatique de Toulouse (UMR CNRS 5505)
2, rue Charles Camichel - BP 7122
F-31071 Toulouse Cedex 7
lastname@irit.fr

July 28, 2008
MDDAS@COMPSAC, Turku, Finland
Context

- Nowadays, software is
  - increasingly complex
  - developed with many different programming languages and paradigms
  - configured with proprietary configuration files
- Software administration
  - costs a lot (man power and money)
  - is error prone
  - lacks reactivity
- Should be automatized!
Context

Autonomic Computing

- One solution: autonomic computing
  - automatic deployment
  - self-management:
    - self configuration,
    - self optimization,
    - self healing,
    - self protection.
  - ...

...
Outline

1. Context

2. Background
   - Jade: a Component-Based Management System
   - TUNe (v1.0): a UML-Based Management System

3. TUNe (v1.1): A DSML-Based Management System
   - The Configuration Description Language
   - The Wrapping Description Language
   - The Deployment Description Language
   - The Reconfiguration Description Language

4. Use Cases

5. Conclusion and Future works
Plan

1. Context

2. Background
   - Jade: a Component-Based Management System
   - TUNe (v1.0): a UML-Based Management System

3. TUNe (v1.1): A DSML-Based Management System
   - The Configuration Description Language
   - The Wrapping Description Language
   - The Deployment Description Language
   - The Reconfiguration Description Language

4. Use Cases

5. Conclusion and Future works
Jade: a Component-Based Management System

- The **System Representation** maintains a consistent **view** of the legacy layer (software elements, attributes and relationships)
- Relies on the Fractal component model
  - Architecture Description Languages (ADL) for deployment
  - component model’s API for management

⇒ Jade system [Hagimont-ClusterComputing2006]
Jade: a Component-Based Management System

Lessons Learned

- A component model is still too low level:
  - administrators have to learn the component model’s ADL (in extension!)
  - integrators have to learn the component model’s API

- Consequences:
  - loss of time and money
  - error prone
  - self-return to the initial state! :D
TUNe (v1.0): a UML-Based Management System

- TUNe (Toulouse University Network)
- To address complexity of component based management
  ⇒ Management policies with a universal language: UML
    - deployment and component diagrams
    - state and activity diagrams
- The System Representation is
  - initially generated at deployment time
    - using deployment diagram
  - potentially modified by reconfigurations
    - using reconfiguration diagrams
⇒ TUNe (v1.0) system [Broto-SAC2008]
TUNe (v1.0): a UML-Based Management System

Overall Architecture
TUNe (v1.0): a UML-Based Management System

Lessons Learned

- TUNe (v1.0) allows a UML-based software administration
  - WDL to describe legacy software encapsulation,
  - Diagram to specify layout and deployment policies,
  - Diagrams to define start and reconfiguration charts.

- However, the use of an unified language (e.g. UML) led us to
  - adapt/twist the original semantics of UML (defined by the TUNe runtime),
  - use the UML formalism that may not be the intuitive one,
  - use UML editors which do not enforce TUNe semantics. (no enforcement).
Plan

1. Context

2. Background
   - Jade: a Component-Based Management System
   - TUNe (v1.0): a UML-Based Management System

3. TUNe (v1.1): A DSML-Based Management System
   - The Configuration Description Language
   - The Wrapping Description Language
   - The Deployment Description Language
   - The Reconfiguration Description Language

4. Use Cases

5. Conclusion and Future works
Each point of view relies on a Domain Specific Modeling Language (DSML),
Each language is supported by constrained tools.
The **Configuration Description Language**

- A simple intentional ADL,
- Allows to define the overall architecture (not in extension!).

```
Attribute
name: String
type: DataType
default: Literal
```

```
Software Element
name: String
filename: String
```

```
Interface
name: String
```

```
Intentional Link
name: String
lower: Int
upper: Int
```

```
serverName: String = "webServer"
user: String = "combemale"
group: String = "users"
serverRoot: String = "/www"
listen: Int = 8002
```

```
httpPort: Int = 8080
ajpPort: Int = 8009
lbFactor: Int = 100
```

```
port: Int = 8001
username: String = "combemale"
password: String = "password"
user: String = "root"
```

```
softwareElement
name: String
filename: String
Attribute
name: String
type: DataType
default: Literal ...
```
The Wrapping Description Language

- Allows to define Wrapper (i.e. legacy software encapsulation) and its relations with Software Element,
- OCL constraints for expressing consistency constraints within the architectural view.
The Deployment Description Language

- Allows to specify deployment policies,
- OCL constraints for expressing consistency constraints within the architectural view.

```
Policy
  body: String
  policy: Int

Software Element
  filename: String

AbstractNode
  name: String
  node: Int

Deployment
  initial: Int
  deployments: 0..*

Cluster1
  policy = "policy.Cluster1"

Cluster2
  policy = "policy.Cluster2"
```

- LB
- Apache
- Tomcat
- MySQL

- Cluster1
- Cluster2
The Reconfiguration Description Language

- Description of the life cycle of each *Software Element* with schemas of automatic (re)configuration,
- Relies on State and Activity diagrams of UML2.0
  - UML profile with 2 kinds of activity:
    - WDL method calls,
    - potentially other methods to add or remove components.
- Clear description of links between events and reconfiguration actions (similarly to ECA rules),
- OCL constraints for expressing consistency constraints within the architectural view.
Plan

1. Context

2. Background
   - Jade: a Component-Based Management System
   - TUNe (v1.0): a UML-Based Management System

3. TUNe (v1.1): A DSML-Based Management System
   - The Configuration Description Language
   - The Wrapping Description Language
   - The Deployment Description Language
   - The Reconfiguration Description Language

4. Use Cases

5. Conclusion and Future works
Use Cases

- J2EE
  - Apache, Tomcat, MySQL
  - self-repair, self-sizing
- Diet (distributed load-balancer)
  - hierarchy of distributed servers
  - deployed on 400 nodes in 2.16 minutes
  - self-repair
- Virtualized OS
  - OS migration with Xen
  - energy management
Plan

1 Context

2 Background
   - Jade: a Component-Based Management System
   - TUNe (v1.0): a UML-Based Management System

3 TUNe (v1.1): A DSML-Based Management System
   - The Configuration Description Language
   - The Wrapping Description Language
   - The Deployment Description Language
   - The Reconfiguration Description Language

4 Use Cases

5 Conclusion and Future works
Conclusion

- **Addressed issues:**
  - administration is complex and costly
  - component based autonomic management still requires the use of low level paradigms
  - UML-based autonomic management still too general (i.e. unified!)
  - TUNe (v1.1) allows DSML-based software administration.

- **Ongoing works** for the version 1.1:
  - To finalize the TUNe metamodel (e.g. extension of the reconfiguration capabilities, improvement of ADL expressiveness...),
  - To provide editing tools by generative approach (e.g. GMF) for the different points of view
Future works: towards a v2.0 of TUNe

- Many different points of view should be considered
  - crosscutting autonomic administration policies,
  - platform description,
  - hardware infrastructure,
  - TUNe self deployment...

- Modularity for different families of languages (⇒ megamodeling!)

- Models@runtime: use of models for the system representation
  - automatic generation of context-aware monitoring and administration tools
  - static and dynamic conformity of the system representation model
  ⇒ a first step towards a model-driven system administration
Conclusion and Future works

Thank you for your attention...

Questions?

Try it!
Send me an email to have access to our SVN
combemale@irit.fr