Towards…
A Model Driven Autonomic Management System

Laurent Broto, Estella Annoni, Daniel Hagimont,
Benoît Combemale and Jean-Paul Bahsoun

IRIT CNRS
2, rue Charles Camichel - BP 7122
F-31071 Toulouse Cedex 7
lastname@irit.fr

April 7, 2008
ITNG, Las Vegas, USA
Outline

1 Introduction

2 TUNe
- Our approach
- The System Representation
- The Wrapping Description Language
- Component & Deployment Diagrams
- Reconfiguration Diagrams

3 Use Cases

4 Conclusion and Future works
Plan

1. Introduction

2. TUNe
   - Our approach
   - The System Representation
   - The Wrapping Description Language
   - Component & Deployment Diagrams
   - Reconfiguration Diagrams

3. Use Cases

4. Conclusion and Future works
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!
Autonomic Computing

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

Combemale et al. (IRIT)
Many proposals rely on a component model
- Architecture Description Languages (ADL) for deployment
- component model’s API for management
  ⇒ Jade system [Hagimont-ClusterComputing2006]

However
- few of them addressed legacy software management
- a component model is still too low level
  - administrators have to learn the component model’s ADL (in extension !)
  - developpers have to learn the component model’s API

Consequences
- loss of time and money
- error prone
- self-return to the initial state ! :D
Plan

1. Introduction

2. TUNe
   - Our approach
   - The System Representation
   - The Wrapping Description Language
   - Component & Deployment Diagrams
   - Reconfiguration Diagrams

3. Use Cases

4. Conclusion and Future works
Our approach

- TUNe (Toulouse University Network)
- Based on a component model
  - Fractal
- To address legacy software management
  - Wrapping legacy software in components
  - Description in WDL files
- The legacy level is reified in components
  - System Representation
- To address component model’s low level API
  - Management policies with a universal language: UML
    - Deployment and component diagrams
    - State and activity diagrams
Overall Architecture

System representation

Management/control interface

Management layer

Legacy layer

Wrapper

Wrapper

Wrapper

(re) Configurations

Deployment

UML diagrams

TUNe

Code Generator

TUNe

Our approach

Combemale et al. (IRIT) A Model Driven Autonomic Management System ITNG, April 7-9 2008 9 / 21
The System Representation

- Maintain a consistent *view* of the legacy layer
  - legacy software attributes
  - legacy software relationships
- Initially generated at deployment time
  - using deployment diagram
- Potentially modified by reconfigurations
  - using reconfiguration diagrams
The Wrapping Description Language

- Wrapping components are difficult to implement
- A language to describe legacy software encapsulation
  - method definitions
    - start()
    - stop()
    - configure()
    - ...
  - define configuration actions implemented in these methods
    - actions are programmed in Java
    - can access any attribute of the component architecture
Component & Deployment Diagrams

- A UML profile
- Specify deployment policies
- Define the overall structure, not in extension
- Ensure consistency
  - this component diagram constrains the component architecture that may be issued by deployment or reconfiguration
Component & Deployment Diagrams

Combemale et al. (IRIT)

A Model Driven Autonomic Management System

ITNG, April 7-9 2008
Reconfiguration Diagrams

- A UML profile
- kinds of activity:
  - WDL method calls
  - potentially another methods to add or remove components
- startchart: a diagram for application launching
- reconfiguration charts: for self-repair and self-optimizing
Reconfiguration Diagrams

[Diagram showing reconfiguration processes with nodes labeled as `this.stop`, `arg.start`, `arg.SeD.stop`, `this.start`, `MA.configure`, `LA.configure`, `SeD.configure`, `MA.start`, `LA.start`, `SeD.start`.]
Plan

1. Introduction

2. TUNe
   - Our approach
   - The System Representation
   - The Wrapping Description Language
   - Component & Deployment Diagrams
   - Reconfiguration Diagrams

3. Use Cases

4. Conclusion and Future works
Use Cases

- Diet (distributed load-balancer)
  - hierarchy of distributed servers
  - deployed on 400 nodes in 2.16 minutes
- J2EE
  - Apache, Tomcat, MySQL
- SNMP (Simple Network Management Protocol)
  - deploying snmp agents
Plan

1 Introduction

2 TUNe
   - Our approach
   - The System Representation
   - The Wrapping Description Language
   - Component & Deployment Diagrams
   - Reconfiguration Diagrams

3 Use Cases

4 Conclusion and Future works
Conclusion

- Addressed issues
  - administration is complex and costly,
  - component based autonomic computing still requires the use of low level paradigms.
- TUNe allows a UML-based software administration
  - System representation by the use of Fractal,
  - WDL to describe legacy software encapsulation,
  - Diagram to specify layout and deployment policies,
  - Diagrams to define start and reconfiguration charts.
Conclusion and Future works

Future works

- Many different diagrams should be considered
  - hardware infrastructure
  - TUNe self deployment!
- Use a metamodel to provide:
  - a constraint and domain-specific languages,
  - a user-friendly tools.
  ⇒ potentially a first step towards a model-driven systems administration
Thank you for your attention...

Questions?

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