Architectural Considerations for Realizing Self-Governance in Autonomic Systems and Networks

John Strassner - john.strassner@motorola.com
Fellow of the Technical Staff and Director, Autonomic Computing, Motorola, USA
Associate Professor, WIT, Waterford, Ireland
Motivation
A Traditional OSS/BSS

Key:
- Internal data flow
- External data flow
- Pending data flow
- NAME System appears twice
- Planned systems
Shortcomings - Infrastructural

- **Architectural issues**
  - Data redundancy
  - Synchronization problems
  - Application authorization issues
  - Vendor and Application “lock in”

- **Integration issues**
  - Isolated Data Silos
  - Administrative nightmare
  - Integration/customization nightmare
  - Transition from legacy systems to a new OSS
Stovepipes are Everywhere!

- Different languages
- Different semantics
- Different programming models

```sh
Router(config)# router bgp autonomous-system
Router(config-router)# neighbor { ip-address | peer-group-name} remote-as number
Router(config-router)# neighbor ip-address activate

[edit]
routing-instances {
    routing-instance-name {
        protocols {
            bgp {
                group group-name; {
                    peer-as as-number;
                    neighbor ip-address; }
            }
        }
    }
}
```
The Problem – Managing Complexity

- The complexity of system design and management keeps increasing
  - Stovepipe systems: best-of-breed functionality but integration nightmares
  - Increased technology overwhelms users and administrators
    - Different devices have different programming models and interaction models
    - Different management tasks and integration types require different skill levels

- The complexity of business is also increasing
  - People are demanding a pervasive presence
  - Many types of businesses LOSE MONEY if they can’t react fast enough
  - Varieties of threats, problems, and non-optimized behavior increasing

- Behavioral complexity is also increasing
  - Everything is interconnected, requiring different policies and functions
  - Too complex to predict, needs too high a skill level, not enough people!
Constituency Separation

- Different constituencies have different terms, grammars, and needs
  - Business “speak” vs. networking commands
    » Service Level Agreement meaning changes per constituency
  - Different representations (e.g., use of UML)

- Network services and resources are *not* currently related to business needs
  - Not reflected in device, EMS and NMS design
  - Lack of policy controlling allocation
  - Lack of ability to
    » Incorporate new knowledge
    » React in a timely manner to changes
Definition
Future Vision of Autonomic Computing?

*Machines will take over all management tasks, rendering humans superfluous.*

Hal 9000, 2001

Wrong!
Future Vision of Autonomic Computing

*Machines will free system administrators to manage system at a higher level*

Right!
Autonomic Computing Definition

- An autonomic system is a self-governing system
  - governance model is expressed using policies
  - policies are bound to business goals

- Self-governance is accomplished through the use of self-knowledge
  - model the capabilities of the system, as a function of context
  - model the constraints placed on the system, as a function of context

- Closed control loop enables the system to
  - sense changes in itself and its environment
  - analyzes changes to ensure that business goals are still being met
  - plan changes to be made if business goals and objectives are threatened
  - execute those changes, and observes the result

- Control loop augmented by self-learning and reasoning processes
Autonomic Computing Overview

- DEN-ng Models and Ontologies
- Managed Resource
- Analyze Data and Events
- Determine State
- Define New Device Configuration(s)
- Policy Manager
- Autonomic Manager
- Match?
  - YES
  - NO
- Business Rules

Control flow:
- Managed Resource to Analyze Data and Events
- Analyze Data and Events to Determine State
- Determine State to Match?
  - YES to Define New Device Configuration(s)
  - NO to Autonomic Manager
- Match? to Business Rules
- Business Rules to Policy Manager
- Policy Manager to Autonomic Manager
- Autonomic Manager to Managed Resource
Autonomic Computing and People

- People express what they want the system to achieve
  - Level could be business, or IT, or programming code

- The system strives to manage its own behavior to optimally satisfy these multiple criteria, given business constraints
  - Resources: Hardware, software, cost
  - Tradeoffs among multiple criteria must be clear

- People and self-managing systems will work together iteratively, in partnership with one another
  - People will do what they’re best at
  - Systems will gradually assume more management burden
    » As they become more competent to do so
    » As people become more comfortable with letting them
Autonomic Networking – What is it About

*Biology, Sociology, and Economics can Inspire Better Networks!*

Biology, Sociology, and Economics can Inspire Better Networks!

- **Technical complexity**: human body ↔ technology, devices
- **Business complexity**: macro-economics ↔ e- and m-Commerce
- **Behavioral complexity**: social interaction ↔ service composition
- **Operational complexity**: healing ↔ anti-virus, configuration management

Complexity abounds!
Autonomic Computing Architecture
Behavioral Orchestration

DEN-ng Models and Ontologies

Gather Vendor-Specific Data

Compare Actual State to Desired State

Define New Device Configuration(s)

Loop 1: Maintenance

Match?

YES

NO

Loop 2: Adjustment

Managed Resource

(models and ontologies)
Developing an AAMC

DEN-ng Models and Ontologies

Able to Control Managed Resource

Autonomic Manager

XML

Model-Based Translation Layer

Vendor-Specific Data

Vendor-Specific Commands

Managed Resource

Translates to Autonomic Language

Not Autonomic!
Revising the Control Loop

DEN-ng Models and Ontologies

Gather Vendor-Specific Data

Compare Actual State to Desired State

Define New Device Configuration(s)

Match?

YES

NO

Loop 1: Maintenance

Loop 2: Adjustment

Managed Resource

Model-Based Translation Layer

Normalize

Mediation Layer

Define New Device Configuration(s)
Add in Policy Management

DEN-ng Models and Ontologies

Policy Manager

Business Rules

Managed Resource

Gather Vendor-Specific Data

Model-Based Translation Layer

Compare Actual State to Desired State

Match?

Define New Device Configuration(s)

Loop 1: Maintenance

Loop 2: Adjustment

Control

Policies control application of intelligence

Autonomic Manager

Control

Mediation Layer

Control

Policy Management
Architectural Principles in More Detail:

- Modeling
- Ontology
- Learning
- Reasoning
- Policy
Unique Design Points of DEN-ng

DEN-ng views the world in terms of
- **Capabilities** - *normalized functionality*
- **Constraints** - *restrictions on what you can use*
- **Context** - *environment in which different objects operate*

DEN-ng is built using patterns and roles
- Patterns are used to capture common relationships and occurrences of physical connections and structures
- Patterns make the model inherently extensible

DEN-ng uses a finite state machine
- Other models represent the current state of an object
- DEN-ng builds different models of different states, and binds them together using a finite state machine
But There’s a Problem…

Data Model A

Cognitive Dissonance!

Data Model B
Knowledge Management
*Adding Semantics*

- **Known knowledge**
  - Issue a command, expect a response
  - Model enables validation of command and response

- **Discovered knowledge**
  - New information that is not modeled in the system
  - Need to identify its relevance and its effect on the system

- **Autonomic systems need both!**
  - Modeling known knowledge enables system behavior orchestration
  - Incorporating relevant unknown knowledge
    - Makes the model more relevant to the current application
    - Verifies that currently used policies and processes are correct
    - Allows *context* to drive behavior
Purpose of Ontologies

- Defines a *shared vocabulary*
- It enables data to be *understood*, so that different data can be related to each other, even if those data are in different formats using different languages
- It enables the efficient *reuse of knowledge*
- Ontologies offer a formal mechanism for defining a better understanding of facts
- Shared ontologies ensure that different components and applications communicate about different aspects of the same entity in a standard way
Semantic Equivalence

- How to Equate Vendor-Specific Data
- Formalize ontology representation
- Calculate semantic similarity between matched concepts by recursively analyzing each aspect of each concept
  - Define a similarity function $S$
  - For each concept $C_1$ in ontology $O_1$, find a similar concept $C_2$ in ontology $O_2$ such that $S(C_1, C_2)$ is maximized
  - Define a weighting to measure contribution of each aspect towards entire semantic similarity rating
  - Results: Exact, superset, subset, overlap

- Find highest similarity match between concepts to perform ontology mapping, then report
Semantics to the Rescue!

Ontology A → Cognitive Similarity! → Ontology B

Data Model A → Cognitive Dissonance! → Data Model B

Study ID: DEMO1

Report ID: NELOAD

TK_SPEED: 1544000 bits/sec

Peakness % Util

A_END: T E S T N J A T M 01

Z_END: T E S T N J A T M 03

© Motorola 2006
Reinforcement Learning

- A learning agent interacts with the environment
  - Observes current (partial) state \( s \) of the environment
  - Takes an action \( a \)
  - Receives an (immediate) scalar reward \( r \)

- Agent learns a long-range value function \( V(s,a) \)

  estimating cumulative future reward: \[ R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \]

  - A standard RL algorithm “Sarsa”: learns state-action value function
    \[ \Delta V(s,a) = \alpha \left( \{ r + \gamma V(s',a') \} - V(s,a) \right) \]

- By design RL does “trial-and-error” learning without model of environment
- Naturally handles long-range dynamic consequences of actions (e.g., transients, switching delays)
- Solid theoretical grounding for MDPs; recent practical success stories
Reasoning Definitions

➢ Deductive
  - Reasoning from the general to the specific
    \[(p \rightarrow q) \land p \Rightarrow q\]

➢ Abductive
  - *Reasoning from effects backwards to causes.*
  - *Reasoning from observed facts to the best possible explanation.*
    \[(p \rightarrow q) \land q \Rightarrow p\]
Purpose of Reasoning

- Form a hypothesis or hypotheses for why something occurred in order to
  - Gain better understanding of the problem and what caused it
  - Hopefully solve the problem! 😊

- Monitor set of known hypotheses in the system
  - Based on *experience*, if a hypothesis is continually proved true, elevate it to a theory
    » Performance increase and knowledge refinement
    » Add/remove affected knowledge to/from the knowledge base
  - Find things that are true which we cannot necessarily prove (*axioms*) and add to the knowledge base
Adding in Semantics and AI

Gather Vendor-Specific Data

Compare Actual State to Desired State

Define New Device Configuration(s)

Match?

YES

NO

Loop 1: Maintenance

Control

Loop 2: Adjustment

Control

Policies control application of intelligence

Business Rules

Policy Manager

Autonomic Manager

Managed Resource

Ontological Comparison

Reasoning and Learning

Model-Based Translation Layer

Normalize

Define New Device Configuration(s)

Ontological Comparison

Reasoning and Learning

Policy Manager

Managed Resource

Ontological Comparison

Reasoning and Learning

Model-Based Translation Layer

Normalize

Define New Device Configuration(s)
Types of Policy

➤ Action Policy
- Specifies action a that should be taken in current state S
  - ON (Event) IF(Condition) THEN (Action)
  - Susceptible to conflicts
- Rationality is compiled into the policy

➤ Goal Policy
- Specifies desired resulting state ρ or criteria for set of states
  - System must compute action $a: S \rightarrow \rho$
  - Any member of desired states acceptable
- Rational behavior is generated by optimizer/planner

➤ Utility Function Policy
- Function assigns a single real value to each resulting state $U(\rho) \rightarrow \rho$
  - Tradeoffs directly encoded, thus no conflicts
  - System must compute optimal action
- Rational behavior is generated by optimizer/planner
Policy Tradeoffs

➤ Action Policies
- Guide an agent in performing a sequence of events
- Aim is to transform system state to a new (desired) state
- May not be able to completely specify actions required to reach new state

➤ Goal Policies
- What we want, rather than what to do
- Agents must be able to deduce what to do
- Requires sophisticated models and planning algorithms

➤ Utility Policies
- The “goal” of Goal Policies
- Map each possible state into a vector (rather than binary goals)
- System uses appropriate optimization technique to determine most valuable state and means for achieving it
- Utility elicitation can be hard!
Action Policy Set for Decision Making

Gold: IF (RT_G > 100 msec) THEN (Increase CPU_G by 5%)

Silver: IF (RT_S > 200 msec) THEN (Increase CPU_S by 5%)

Overlapping Action Policies (conflict depends on CPU utilization)

G: IF (RT_G > 100 msec) THEN (Increase CPU_G by 5%) : Priority = 10
S: IF (RT_S > 200 msec) THEN (Increase CPU_S by 5%) : Priority = 5

No explosion of state space
Goal Policies

\[ G: \text{RT}_G < 100 \text{ msec} \]
\[ S: \text{RT}_S < 200 \text{ msec} \]

Performance model: \[ \overline{T}(\bar{\lambda}, \bar{C}) \]

Gives feasible response time curves

Conflict: Gold/Silver Tradeoff
What to do?

It’s all good! What is best?

It’s all bad! What to do?
Utility Function Policies

\[ U(RT_G, RT_S) = U_G(RT_G) + U_S(RT_S) \]

- States have real value, rather than binary good/bad classification
- Map all states of interest in to single unique value
- Tradeoffs directly encoded
- *A means to better use action and goal policies*
Concluding Example
Service Maintenance Scenario

What’s UML?
Is Harder Than It Looks…

Where is Jack?

HELP!

XJ4_7721
Business to System Interactions

- Business Rules
- Other Rules
- Environmental Constraints
- Intelligent Devices

Business Process Model

Finite State Machine

Shared Information Data Model

Code Generation
Model-Driven Code Generation

DEN-ng UML Model → Schema Preparation Process → Parsed Output

Model Mapping Rules → Schema Generator Process → Documentation and Help Files

Errors and Warnings

Directory and JavaSpace Mappings for Persistence

Java Mapping for Session Computation
Code Generation On Steroids

➢ Conversion of UML-isms to Java
➢ Class and attribute name translation driven by configurable property files
➢ Extraction of enumerations from UML source files
➢ Auto-insertion of import statements to handle class dependencies
➢ Association maintenance
➢ Produce a programmer’s reference to
  - Facilitate developer’s usage and understanding of the model
  - Translate complex relationships, especially recursion, into English
    » Define what the legal containers of a given element are
    » Serves as a QA check of the model
Process Flow

1. Instrument Switch to Generate SysLog Events
2. Detect New SysLog Event
3. Match Event?
4. Analyze Event
5. Map Event Data to DEN-ng Model
6. Determine Capabilities of New Hardware
7. Match Capabilities to Work Order
8. Configure New Hardware
9. Test
10. Notify Other OSS & BSS Components
Summary
Conclusions

➢ Autonomic computing is about managing complexity
  - Machine intelligence for the service of people
  - Ensures consistency of human decisions
  - Architecture built to enable business to drive network services

➢ Autonomic Computing is a grand challenge, requiring advances in several fields of science and technology
  - Architecture, Systems, Software Engineering
  - Modeling, Optimization
  - Artificial Intelligence: planning, learning, knowledge representation, multi-agent systems, negotiation, emergent behavior
  - Human-system interfaces and Policy

➢ Integrating these technologies to support self-management in complex, realistic environments is a research challenge in itself
  - What are the best architectures and design patterns?
  - Building system prototypes is key to developing and validating AC technology and architecture
Questions?

“Create like a god. Command like a king. Work like a slave”
- Constantin Brancusi
References
Conferences and Web Sites

- **International Conference on Autonomic Computing (ICAC 2006)**
  - June 13-16, 2006 in Dublin
  - [www.autonomic-conference.org](http://www.autonomic-conference.org)
    » Can find programs and presentations from ICAC ’04 and ‘05
  - Several AC workshops held in conjunction with ICAC 2006
    » Hot Topics in Autonomic Computing
    » Self-Managing Systems, Networks and Services
    » Engineering Emergence for Autonomic Systems
    » Smart Grid Technologies

- **Web sites**
  - General: [www.autonomiccomputing.org](http://www.autonomiccomputing.org)
Autonomic Computing References


  - Contains citations for over 50 Autonomic Computing research papers


Foundational References (1)


Foundational References (2)

- [http://www.loa-cnr.it/guarino.html](http://www.loa-cnr.it/guarino.html)
TMF References


MDA, Patterns, and Roles


[3] In particular, variations of the role object pattern are used - please see http://www.riehle.org/computer-science-research/1997/plop-1997-role-object.pdf
Framework References

[1] Please see: www.omg.org/mda
Applications

➢ DB2 Utilities Throttling (DB2 v8.1)

➢ Self-tuning memory management