The Autonomic Computing Architecture

Ric Telford
Director, Strategy and Technology
Autonomic Computing
April 14, 2004
Agenda

- Autonomic Computing Architecture
  - Touchpoints
  - Autonomic Managers
  - K-services

- Autonomic Computing Core Technologies
  - Problem Determination and Self-Healing
  - Solution Change Management and Self-Configuring
  - Autonomic Computing Policy Management for delivering policy-driven IT
Components of the Autonomic Computing Architecture

The autonomic computing architecture abstracts or organizes the systems into some basic elements.

- **Touchpoints**

- **Autonomic Managers**

- **K-services**
A major contributor to the complexity of managing an IT infrastructure is the diverse syntax and semantics in the mechanism used for the manageability interface.

A touchpoint is an autonomic computing system building block that implements the sensor and effector behavior for one or more of the managed resource manageability mechanism.
**Building Block: Touchpoint**

**Overview**

*A touchpoint is an autonomic computing system building block that implements the sensor/effector pattern for one or more of the manageability interface mechanisms.*

Managers obtain **details** through its **sensor**. Two Modes:
1. Manager requests **details**
2. Resource provides **details**

Managers **changes** details through its **effector**. Two Modes:
1. Manager initiates the **change**
2. Resource requests the **change**.

The **details** a manager needs to monitor and control a managed resource.

The mechanisms used to monitor and control a managed resource form its **manageability interface**.
Building Block: Touchpoint

A manageability interface for a managed resource that incorporates these four interaction styles enables most self management scenarios.

A sensor enables a client to access state using two styles:
- **Retrieve-State** is an interaction style in which a client polls for some details.
- **Receive-Notification** is an interaction style in which a resource manager sends an unsolicited message.

A effector enables a client to change state using two styles:
- **Perform-Operation** is an interaction style in which a client issues a command against a resource manager.
- **Call-Out-Request** is an interaction style in which the resource manager asks another capability for some details.
Building Block: Autonomic Manager

An autonomic manager is a configuration of automated functions that deliver “self management” capabilities.
Self-Management is an automation style that implements a control loop that is driven by the circumstances observed in the system.

An autonomic manager is an autonomic system building block that implements a control loop.

- Analyzes observed situations to determine if some change needs to be made.
- Collects details from the system and organizes them into situations that need to be analyzed.
- Shares accumulated knowledge across the elements.
- Creates or selects a plan to make a desired change.
- Makes the changes by performing the plan.
Building Block: Autonomic Manager

Self-Management is an automation style that implements a control loop that is driven by the circumstances observed in the system.
Building Block: Autonomic Managers

**“Orchestrating” AUTONOMIC MANAGER**
- Accepts higher level business goals
- Translates business policy into goals and objectives for the resource it’s managing
- Pushes Goals down onto its managed elements

**“Touchpoint” AUTONOMIC MANAGER**
- Accepts goals
- Translates goals into effectors to be pressed
- Pushes down onto effectors and measures goals via sensors

Managed Resource
- Accepts decisions
- Manages resources accordingly
Building Block: K-Service

A *k-service* is used to share knowledge between autonomic managers.

K-Types define the syntax and semantics for a type of knowledge.
K-Type is “configure” data for an AM.
When appropriate, identify/build enabling technology for *k*-types.
K-Service is a building block for sharing knowledge between AM.
K-Service existing for *k*-type/query combinations.
Building Block: K-Service

Knowledge can be passed to the autonomic manager as configuration data or the autonomic manager can request knowledge as configuration data.

The behavior of the autonomic manager is controlled by policies that describe what needs to be accomplished.
Interaction between components

*The interfaces for an Autonomic Manager and a Touchpoint are defined as “services”.*
A simple example

- Autonomic elements have two management tasks
  - They manage themselves
  - They manage their relationships with other elements through negotiated agreements

Autonomic Database

“I need to allocate some additional table space”

Autonomic Storage Array

“I am reallocating storage and moving the information”
Multiple Contexts for Autonomic Behavior

Customer Relationship Management

Enterprise Resource Planning

Business Solutions
(Business Policies, Processes, Contracts)

Server Farm
Enterprise Network
Storage Pool

Groups of Elements
(Inter-element self-management)

System Elements
(Intra-element self-management)

Servers  Storage  Network Devices  Middleware  Database  Applications
Core AC Problem Determination Technology: First steps towards Self-Healing Systems

1. **Common Base Event (CBE) Model**
   - Standard to facilitate intercommunication among components supporting logging and problem determination.

2. **Generic Log Adapter**
   - Converts existing log files into CBE format

3. **Log and Trace Analyzer**
   - Organizes log and trace data into CBE format for problem determination

4. **Symptom Database**
   - File of symptoms, string match patterns, associated solutions and directives used in analysis of events and messages in a log.
Data elements in logs need to be in a consistent format to facilitate correlation of events from different infrastructure components, and to facilitate effective intercommunication among disparate applications and systems.

- Common Base Event (CBE) model is a standard describing how system activity is recorded and communicated.
- Common format for logging, management, problem determination, and autonomic computing

CBE Elements:

1. Identification of component reporting the situation
2. Identification of component affected by situation
3. The situation (REQUEST, START, REPORT, STOP, DEPENDENCY, CONFIGURE, CREATE, CONNECT, etc)
Generic Log Adapter: Overview

- An adapter for the conversion of existing log formats into CBE
- Standards based: Java plug-in on top of the Eclipse platform
- GUI: For the creation of mapping rules.
- Runtime: Takes mapping rules as input and produces CBE records as output.
- Open Source – Project Hyades: http://eclipse.org/hyades
Log and Trace Analyzer: Overview

**Customer pain point:**
Difficulty in analyzing problems in multi-component systems

- Viewing, analysis, and correlation of log files
- Consolidated environment that deals with logs and traces produced by various components
- Easier and faster for developers and support personnel to debug and resolve problems
- Link to WebSphere symptom database available today
Log and Trace Analyzer: Parsers and Correlation Engines

- Eclipse based tools
- Built in parsers: Imports existing log files and converts to CBE format on the fly.
- Built in correlation engines: Visually displays the correlation between log records using a number of factors:
  - Sequential Correlation
  - Associative Correlation
Log and Trace Analyzer: Symptom Database

- Used in the analysis of events and error messages that may occur in a log.
- XML file of symptoms, string match patterns, associated solutions, and directives.
- **Symptom Database Editor**: Edit existing symptom databases, or create custom symptom databases specifically for your environment or applications.
  - Define application specific directives and solutions
  - Augment a product’s symptom database based on actual experience
Log and Trace Analyzer: Profiling Tool

- Tool for profiling applications in real time to diagnose performance and memory leak problems
- Interactively profile applications on local and remote deployment environments
Solution Change Manager

*Customer pain point:*
Difficulty of deployment in complex systems

- A common infrastructure to ensure a simpler and more consistent deployment experience.
  - Common **tooling** to reduce the cost and complexity of building, deploying, and maintaining software solutions.
  - Common **deployment descriptors** to describe the installation capabilities and dependency requirements for a given software package.
  - Common **packaging** to which can be used for new installations, upgrades, and maintenance.
  - Common **dependency checking** technologies to validate environment (hardware, OS, software, configuration, etc.)

- Consistent methodology for creating software packages
- Install, update, fix, uninstall, repair, rollback, commit the package
- Verifying the deployment so the software is ready to use

**Architecture and Standards**
- Data model of an installation package and installable units
- Interfaces of components to process this data
Solution Change Manager Highlights

Tooling

- Create Application Components and descriptors
- Create Solutions/Packages
- Create Updates

Installer

- Product Install
- Update Install

Dependency Checker

- Analyze Dependencies
- Deploy pre-reqs (as part of Package)
- Deploy Application Components

Enablement in the middleware and OS
Solutions Administration Today

- Industry Solutions include...
  - eServer, WebSphere, Tivoli, DB2, Lotus, Rational...
  - Business Partner Applications
  - Customer Applications
- Different Admin consoles
  - No look & feel consistency
  - No administration integration
- Multiple - costly learning curves
  - Delayed deployment of solution
  - Increased admin training costs
- Different technologies
  - Java, C, C++, HTML, XML
  - Installed Client
  - Web based
Integrated Solutions Console Technology

- Standards-based architecture
  - J2EE, Java, XML
  - JSR 168 - Portlet API's
- Portlets allow administration functions to be developed in a solution-oriented manner
- Packaged and deployed like J2EE Web Applications
Console Components built on ISC
Goal: Admin Console Convergence

Integrated Solutions Console (ISC)
Example Functions

- **System Health**
  - Group Status and Properties
  - System Status and Properties
  - Resource Status and Properties

- **Problem Identification**
  - Consolidated Monitoring
  - Access to logs and message queues

- **Corrective Management**
  - System Control (e.g. shutdown, restart, etc.)
  - Job/Process Management and Control (e.g. kill a process)
  - Resource Management and Control (e.g. delete an event,
  - Task Execution
Summary of Autonomic Computing Policy Goals

- Develop the AC Policy Language (4-tuple) specification
  - XML grammar that provides a unified view of policy content across a heterogeneous enterprise

- Develop technology – Policy Management for Autonomic Computing – which delivers a policy-driven autonomic manager for resource management
  - Used to configure and manage resources

- Provide design to guidance for developing system-level Autonomic Managers
  - Goal-based Autonomic Managers, like eWLM
  - Joint work w/ ODDC
Types of Policies

- **Business Policy**
  - Typically encoded into applications, or associated with rules
  - E.g., Gold customers get better airline seats, faster service

- **IT Policy**
  - Typically encoded into the IT application, or occasionally with policy-based management
  - E.g., Gold transactions get 500ms average response time

- Seldom intersect, but should
  - Gold customer gets preferential application treatment (gold seating), and preferential IT treatment (workload)

Our aim is an integrated, policy-based system: Easier to manage, better customer experience
Manager / managed is a common IT paradigm

Policies guide the behavior of the manager. “If the employee is not a mobile user, then their cell phone bill is not reimbursable”.

This is what I’m doing

This is what you should do.
Background: Autonomic Managers and Policy

Autonomic Managers are simply Managers conforming to AC interfaces and data formats

Autonomic managers are guided by ("configured by") policies (AC Policy Language, aka 4-tuple).

This is what I’m doing (Common Base Event)
This is what you should do.

Sensor and effector interfaces, event, policy, etc.
Anatomy of the AC Policy Language (“4-tuple”)

- Four common concepts identified:
  - **Scope** • Specifications to identify what is or is not subject to the intent.
  - **Precondition** • Specifications to express when a policy is to be applied or is active.
  - **Business Value** • Specifications to express utility functions to make economic trade offs
  - **Decision** (Goal/Action/Result) • Specifications to describe observable behavior or objective.

- Designed by adopting concepts from the industry policy languages
  - Workload Mgmt, Provisioning, IETF/DMTF standard, Storage policies
A Simple Policy Example

Scope: server type X
PC:
  event: application stopped
condition:
  service X=running
BV: 10
Decision: Restart app

Monitors for “application stopped” CBE.

Retrieves “service X state”.

Evaluates the policy conditions and business value

Restart the app.
Policy Management for Autonomic Computing: High-level Solution Architecture

Policy Editing Tools

Policy Definition (Policy Grammar)

Editor

Editor Storage

Autonomic Manager

A

P

M

K

E

Retrieve State

Command

Notify

Call out

Managed Resource

© 2004 IBM Corporation
Autonomic Manager vocabulary: Touchpoint Autonomic Managers and Orchestrating Autonomic Managers

Some AMs manage resources directly, others manage AMs.
Assembling storage components: Business Continuity and Provisioning

Storage Manager
Scope: gold transactions
Precondition: 8am to 8pm, data size < 1MB
Decision: 95% reads < 10 ms
Biz Value: very high

Storage Provisioner
Scope: gold transactions
Precondition: 8am to 8pm, data size < 1MB
Decision: 95% reads < 10 ms
Biz Value: very high

Storage Recovery Manager
Scope: storage devices
Precondition: 8am to 8pm,
Decision: Recovery within 15 mins
Biz Value: very high

Storage Backup Managers
Scope: storage devices
Precondition: 8am to 8pm,
Decision: log file synch every 5 mins
Biz Value: very high

Storage
Storage

Autonomic Manager
## Toward an Autonomic, Policy-driven System

<table>
<thead>
<tr>
<th>Level 1</th>
<th>Level 2</th>
<th>Level 3</th>
<th>Level 4</th>
<th>Level 5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic</strong></td>
<td><strong>Managed</strong></td>
<td><strong>Predictive</strong></td>
<td><strong>Adaptive</strong></td>
<td><strong>Autonomic</strong></td>
</tr>
<tr>
<td>Manual analysis and problem solving</td>
<td>Centralized tools, manual actions</td>
<td>Cross-resource correlation and guidance</td>
<td>System monitors, correlates and takes action</td>
<td>Dynamic business policy based management</td>
</tr>
</tbody>
</table>

### Static Code
- Unable to tailor the behavior of the resource

### Parameterized Code
- Tailoring possible, but requires manual effort and monitoring

### Scripts
- Current state of most customers
- Automate set of actions
- Programming skill required

### Resource Policies
- Declarative version of monitor and react scripts
- No programming

### Cross-resource System Policies
- Declarative, cross-resource specification of intentions
Toward an Autonomic, Policy-driven Systems

0. Static Code
1. Parameters to configure a resource
2. Scripts manage a resource
3. Action policies manage a resource
4. Goal policies manage a resource
5. Goal policies manage resources

Most customers (and implementations) are at levels 1 and 2.
Summary

- Autonomic Computing represents the future of managing complexity in IT
- Autonomic Computing needs to be implemented in a consistent way to ensure interoperability across components, hence the need for an architecture and standards
- Autonomic Computing can be accelerated by having a common set of core technologies – common problem determination, install and policy are critical

“IBM’s autonomic computing initiative will become its most important cross-product initiative—Thomas Bittman, Gartner Group”
The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM’s sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

IBM, the IBM logo, the e-business logo and other IBM products and services are trademarks or registered trademarks of the International Business Machines Corporation, in the United States, other countries or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries or both.

Microsoft, Windows, Windows NT and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries or both.

All other trademarks, company, products or service names may be trademarks, registered trademarks or service marks of others.

Disclaimer: NOTICE – BUSINESS VALUE INFORMATION IS PROVIDED TO YOU ‘AS IS’ WITH THE UNDERSTANDING THAT THERE ARE NO REPRESENTATIONS OR WARRANTIES OF ANY KIND EITHER EXPRESS OR IMPLIED. IBM DISCLAIMS ALL WARRANTIES INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IBM DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING THE USE, VALIDITY, ACCURACY OR RELIABILITY OF THE BUSINESS BENEFITS SHOWN. IN NO EVENT SHALL IBM BE LIABLE FOR ANY DAMAGES, INCLUDING THOSE ARISING AS A RESULT OF IBM’S NEGLIGENCE, WHETHER THOSE DAMAGES ARE DIRECT, CONSEQUENTIAL, INCIDENTAL, OR SPECIAL, FLOWING FROM YOUR USE OF OR INABILITY TO USE THE INFORMATION PROVIDED HEREWITH OR RESULTS EVEN IF IBM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE ULTIMATE RESPONSIBILITY FOR ACHIEVING THE CALCULATED RESULTS REMAINS WITH YOU.