Self-healing systems – What are they?

Tiina Niklander
Seminar introduction, 2007
Earlier version: AMICT, Aug 2006
Content

• Overview
• Autonomic Computing
• Elements of Self-Healing
• Architectural approach
• Examples
Overview

SELF-MANAGEMENT

SELF-CONFIGURING

SELF-OPTIMIZING

SELF-HEALING

SELF-ADAPTIVE

SELF-PROTECTING

SELF-ORGANIZING

Autonomic Computing Initiative by IBM, 2001
Self-* (selfware)

- Self-configuring
- Self-healing
- Self-optimising
- Self-protecting
- Self-aware
- Self-monitor
- Self-adjust
- Self-adaptive
- Self-governing
- Self-managed
- Self-controlling
- Self-repairing
- Self-organising
- Self-evolving
- Self-reconfiguration
- Self-maintenance
Eight Goals for a System

1. System must know itself
2. System must be able to reconfigure itself within its operational environment
3. System must pre-emptively optimise itself
4. System must detect and respond to its own faults as they develop
5. System must detect and respond to intrusions and attacks
6. System must know its context of use
7. System must live in an open world
8. System must actively shrink the gap between user/business goals and IT solutions
Autonomic Computing

• Basic model: closed control loops
  – Based on Process Control Theory

• Controller continuously compares the actual and expected behavior and makes needed adjustments

SEE: Any control-theory books
Autonomic Control Loop

**Collect**

- Collate, combine,
- Find trends, correlations

**Analyse**

- Use uncertain reasoning
- Policies, rules, …

**Decide**

- Modify behavior,
- Inform users,
- From system elements,
- Users, environment, agents, …
### Elements of Self-Healing 1/2

<table>
<thead>
<tr>
<th>Fault model</th>
<th>Fault duration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fault manifestation</td>
</tr>
<tr>
<td></td>
<td>Fault source</td>
</tr>
<tr>
<td></td>
<td>Granularity</td>
</tr>
<tr>
<td></td>
<td>Fault profile expectations</td>
</tr>
<tr>
<td>System response</td>
<td>Fault Detection</td>
</tr>
<tr>
<td></td>
<td>Degradation</td>
</tr>
<tr>
<td></td>
<td>Fault response</td>
</tr>
<tr>
<td></td>
<td>Fault recovery</td>
</tr>
<tr>
<td></td>
<td>Time constants</td>
</tr>
<tr>
<td></td>
<td>Assurance</td>
</tr>
</tbody>
</table>

Fault models

• Each aspect describes a characteristic of the fault.
  – Duration: Is the fault permanent?
  – Manifestation: What does the fault do to the system?
  – Source: Where does the fault come from?
  – Granularity: Is the fault global or local?
  – Occurrence expectation: How often will the fault occur?
System Response

- Each aspect describes a characteristic of reacting to faults.
  - Detection: How does a system detect faults?
  - Degradation: Will the system tolerate running in a degraded state?
  - Response: What does a system do when the fault occurs?
  - Recovery: Once a fault occurs, can the system return to a healthy state?
  - Time: How much time does the system have to respond to a fault?
  - Assurance: What assurances does a system have to maintain while handling a fault?
# Elements of Self-Healing 2/2

| System completeness | Architectural completeness  
|:--------------------|-----------------------------  
|                     | Designer Knowledge  
|                     | System self-knowledge  
|                     | System evolution  
| Design context      | Abstraction level  
|                     | Component homogeneity  
|                     | Behavioral predetermination  
|                     | User involvement in healing  
|                     | System linearity  
|                     | System scope  

System Completeness

• Each aspect describes how system implementation affects self-healing.
  – Architecture completeness: How does the system deal with incomplete and unknown parts?
  – Designer knowledge: How do developers deal with unavoidable abstractions?
  – System self-knowledge: What does the system need to know about its components perform self-healing?
  – System evolution: How does the system cope with changing components and environments?
Design Context

• Each aspect describes how system design affects self-healing.
  – Component homogeneity: Are the system’s distributed components homogeneous?
  – Behavioral predetermination: Is the system non-deterministic?
  – User involvement: Does a user do some of the healing?
  – System linearity: Is the system constructed out of composable components?
  – System scope: Does the size of the system affect self-healing possibilities?
Alternative taxonomy

• Maintenance of health
  – Redundancy, probing, ADL, component relation and regularities, diversity, log-analysis

• Detection of failure, discovery of non-self
  – Missing, monitoring model, notification of aliens

• System recovery back to healthy state
  – Redundancy, repair strategies, repair plan, self-assembly, recovery-oriented computing, replication, gauges, event-based action,

Size of the self-healing unit?

- **Component**
  - Focus on connectors and component discovery
- **Service**
  - Service interfaces, Service discovery, restart
- **Node**
  - Network and interface failures, change to new connection
Architectural approach

• The healing or recovery part often requires reconfiguration and adaptation
• They change the architecture
  – Locate and use alternative component
  – Restart (or rejuvenation or resurrection) the failed component
• Self-healing can be build on reflective middleware
Experiments

• OSAD – model (On-demand Service Assembly and Delivery)
• MARKS – Middleware Adaptability for Resource discovery, Knowledge usability and Self-healing
• PAC – Autonomic Computing in Personal Computing Environment
• Using self-healing components and connectors
Life-cycle of Self-Healing

- OSAD – On-demand Service Assembly and Delivery
- Prototype in JINI environment
- Looking for alternatives only by name

MARKS

- Middleware Adaptability for Resource Discovery, Knowledge Usability and Self-healing

- Marks is targeted at embedded and pervasive, small mobile handheld devices.
- New Services: Context, Knowledge Usability and Self-Healing
- Prototype: Dell Axim 30 pocket PC & .NET
MARKS Architecture

- Services
- Core components
- ORB

Figure 1. MARKS architecture
Self-healing in MARKS

• *Healing manager* (of the network) to handle all fault types
  – To isolate faulty device (Fault containment)
  – Select surrogate device or share load among working members
• Resource manager used as repository of information for backup purposes
• *Self-healing unit* (on each device)
  – One process named *rate of change of status*
  – For monitoring the device and announcing the conditions
Self-healing components and connectors

- Healing layer
  - Monitoring and reconfiguration decisions
- Service layer
  - Normal functionality
  - Report all events to healing layer

Shin, M.E.; Jung Hoon An; Self-Reconfiguration in Self-Healing Systems
Self-healing component

- For healing:
  - Self-Healing controller
  - Component monitor
  - Reconfiguration manager
  - Repair manager
Fig. 4 Dynamic self-configuration in Component 1
Reconfiguration decision

• Anomaly detection:
  – Compare observed and expected behavior
• Isolate the ‘faulty’ object
• Repair or replace the faulty object (and return back to normal operation)
PAC – Personal Autonomic Computing

• Goal: collaboration among personal systems to take a shared responsibility for self-awareness and environment awareness

• Proof of concept: *self-healing tool* utilizing pulse monitor (heart beat)

Sterritt, R.; Bantz, D.F.;
Personal autonomic computing reflex reactions and self-healing
*Systems, Man and Cybernetics, Part C, IEEE Transactions on*
Volume 36, Issue 3, May 2006 Page(s):304 - 314
PAC

Fig. 3. Autonomic environment.
PAC

- Autonomic manager
  - Self-adjuster
  - Self-monitor
  - Internal-monitor
  - External-monitor
  - Pulse-monitor (and generator)

Fig. 2. Architecture of a PAC element.
Conclusions

• Self-healing has three roots:
  – Autonomic and self-management world
  – Distributed systems world (especially middleware)
  – Dependable and fault-tolerance world

• The failure recognition and repair decisions might be faster if autonomic

• However: effects of incorrect decisions can be large (and correct them time consuming)
References

- Philip Koopman: Elements of the Self-Healing System Problem Space. In Proceedings of ICSE WADS 03
16.1.2007 31

Additional material

1. George Heineman A Model for Designing Adaptable Software Components

2. Vikram Adve, Vinh Vi Lam, Brian Ensink Language and Compiler Support for Adaptive Distributed Applications
   ACM SIGPLAN Workshop on Optimization of Middleware and Distributed Systems (OM 2001) Snowbird, Utah, June 2001
   (in conjunction with PLDI2001)

3. Marija Rakic, Nenad Medvidovic Increasing the Confidence in Off-the-Shelf Components: A Software Connector-Based Approach
   Proceedings of SSR '01 on 2001 Symposium on Software Reusability : Putting Software Reuse in Context
   International Conference on Software Engineering, May 1999

   In proceedings of Workshop on Self-Healing, Adaptive and self-MANaged Systems (SHAMAN) 2002

6. Yennun Huang, Chandra Kintala, Nick Kolettis, N. Dudley Fulton Software Rejuvenation: Analysis, Module and Applications 

IBM director software rejuvenation. – white paper
6. **David Patterson, et.al.** Recovery Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies

7. **George Candea, James Cutler, Armando Fox, Rushabh Doshi, Priyank Garg, Rakesh Gowda** Reducing Recovery Time in a Small Recursively Restartable System
   Appears in Proceedings of the International Conference on Dependable Systems and Networks (DSN-2002), June 2002

8. **Aaron B. Brown, David A. Patterson** Rewind, Repair, Replay: Three R's to Dependability
   To appear in 10th ACM SIGOPS European Workshop, Saint-Emilion, France, September 2002

9. **Sheng Liang, Gilad Bracha** Dynamic Class Loading in the Java(TM) Virtual Machine
   Conference on Object-oriented programming, systems, languages, and applications (OOPSLA'98)
Schedule (conference simulation)

• 1. period: Writing the paper
  – 2. meeting: List of references, refinement of the topic
  – 3. meeting: Table of content
  – 4. meeting: draft (to show to Tiina)
  – 5. meeting: Paper ready for review
  – 6. meeting: Review feedback (from two members)
  – Paper ready and submitted before second period

• 2. period: Presentations
Seminar topics for Spring 2007

• Faults / Recovery / Autonomic computing
• Self-adaptive services
• Configuration-level adaptation
• Self-healing architectures
  – Agent-based
  – Components
  – Middleware
• Performance issues
  – Self-optimisation etc.
Seminar topics for Spring 2007

• Detection and monitoring
• Instrumentation
• Diagnosis (intelligent systems area)
• Repair
  – Dynamic updates
  – Hot-swap & reconfiguration (software /hardware)
  – Remote healing
• Network related
  – Survivable networks
  – Sensor networks
Software analysis / design for healing