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)

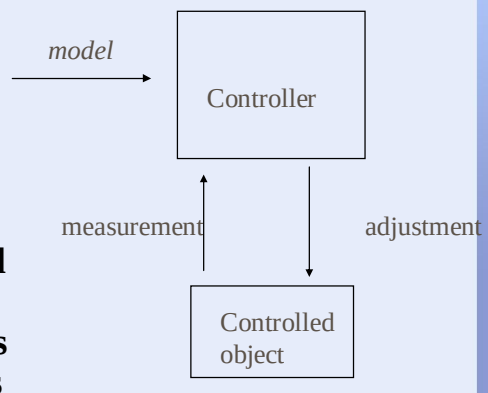
- **Self-configuring**
- **Self-healing**
- **Self-optimising**
- **Self-protecting**
- **Self-aware**
- **Self-monitor**
- **Self-adjust**
- **Self-adaptive**
- **Self-governing**
- **Self-managed**
- **Self-controlling**
- **Self-repairing**
- **Self-organising**
- **Self-evolving**
- **Self-reconfiguration**
- **Self-maintenance**

Eight Goals for a System

1. System must know itself
2. System must be able to reconfigure itseld 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

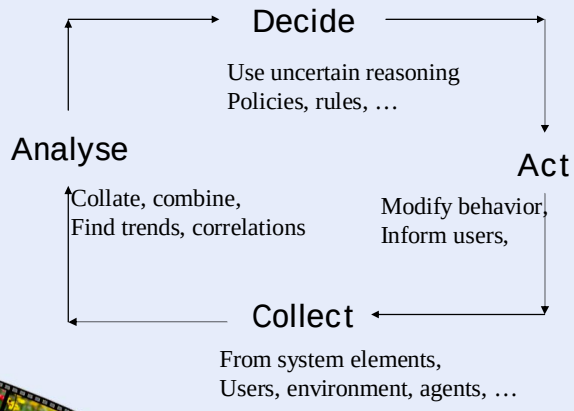
Autonomic Computing

- 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



Elements of Self-Healing 1/2

Fault model	Fault duration Fault manifestation Fault source Granularity Fault profile expectations
System response	Fault Detection Degradation Fault response Fault recovery Time constants Assurance

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

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

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**

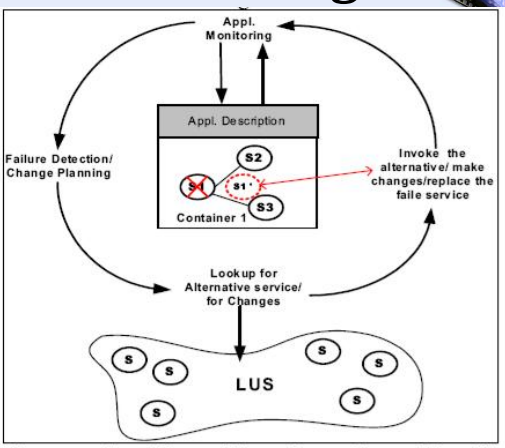


Figure 1. The lifecycle of self-healing behaviour in OSAD model.

Grishikashvili, E.; Pereira, R.; Taleb-Bendiab, A.; **Performance Evaluation for Self-Healing Distributed Services**Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on
Volume 2, 20-22 July 2005 Page(s):135 - 139

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

Sharmin, M.; Ahmed, S.; Ahamed, S.I.;**MARKS (Middleware Adaptability for Resource Discovery, Knowledge Usability and Self-healing) for Mobile Devices of Pervasive Computing Environments**
Information Technology: New Generations, 2006. ITNG 2006. Third International Conference on
10-12 April 2006 Page(s):306 - 313

MARKS Architecture

- Services
- Core components
- ORB

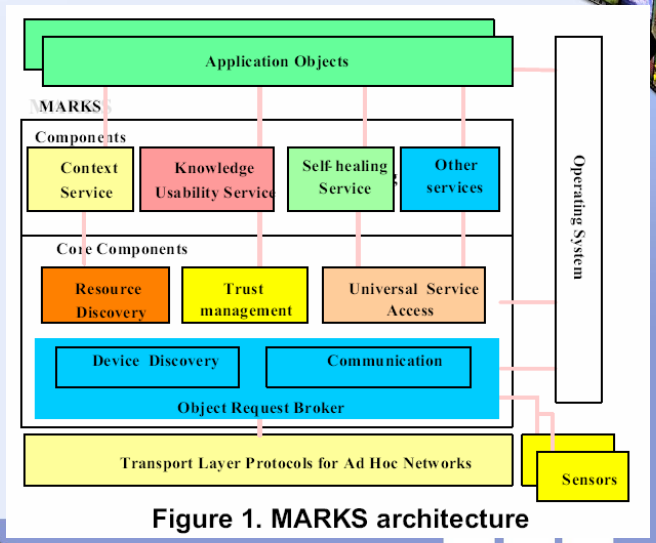


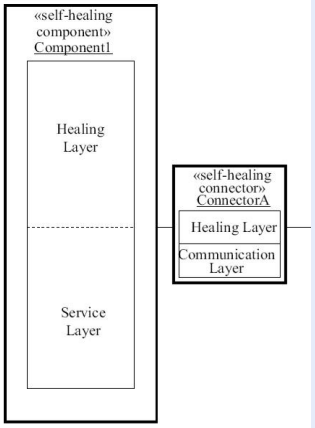
Figure 1. MARKS architecture

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

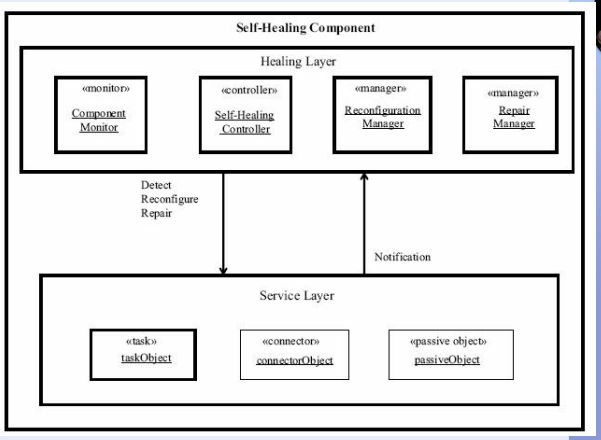
- **Healing layer**
  - **Monitoring and reconfiguration decisions**
- **Service layer**
  - **Normal functionality**
  - **Report all events to healing layer**



Shin, M.E.; Jung Hoon An; **Self-Reconfiguration in Self-Healing Systems** Engineering of Autonomic and Autonomous Systems, 2006. EASe 2006. Proceedings of the Third IEEE International Workshop on 27-30 March 2006 Page(s):89 - 98

Self-healing component

- **For healing:**
  - **Self-Healing controller**
  - **Component monitor**
  - **Reconfiguration manager**
  - **Repair manager**



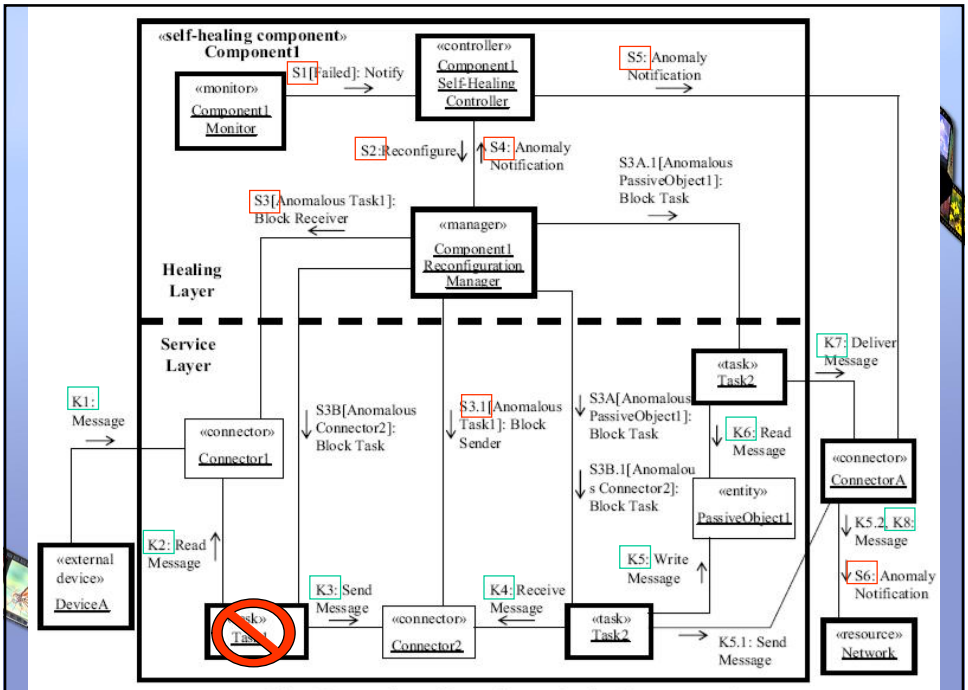


Fig.4 Dynamic self-configuration in Component1

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)**

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
- Jeffrey O. Kephart and David M. Chess: The Vision of Autonomic Computing. IEEE Computer, January 2003, pp. 41-50

Additional material

1. A Model for Designing Adaptable Software Components
   George Heineman
   In 22nd Annual International Computer Software and Applications Conference, pages 121--127, Vienna, Austria, August 1998. In 22nd Annual International Computer Software and Applications Conference, pages 121--127, Vienna, Austria, August 1998
2. Language and Compiler Support for Adaptive Distributed Applications
   Vikram Adve, Vinh Vi Lam, Brian Ensink
   ACM SIGPLAN Workshop on Optimization of Middleware and Distributed Systems (OM 2001) Snowbird, Utah, June 2001 (in conjunction with PLDI2001)
3. Increasing the Confidence in Off-the-Shelf Components: A Software Connector-Based Approach
   Marija Rakic, Nenad Medvidovic
   Proceedings of SSR '01 on 2001 Symposium on Software Reusability : Putting Software Reuse in Context
4. A Cooperative Approach to Support Software Deployment Using the Software Dock
   Richard S. Hall, Dennis Heimbigner, Alexander L. Wolf
   International Conference on Software Enginering, May 1999
5. The Illinois GRACE Project: Global Resource Adaptation through CoopEration
   Sarita V. Adve, Albert F. Harris, Christopher J. Hughes, Douglas L. Jones, Robin H. Kravets, Klara Nahrstedt, Daniel Grobe Sachs, Ruchira Sasanka, Jayanth Srinivisan, Wanghong Yuan
   In proceedings of Workshop on Self-Healing, Adaptive and self-MANaged Systems (SHAMAN) 2002

6. Autonomic Computing
   Paul Horn, IBM Research
7. Software Rejuventation: Analysis, Module and Applications
   Yennun Huang, Chandra Kintala, Nick Kolettis, N. Dudley Fulton
   Proceedings of the 25th International Symposium on Fault-Tolerant Computing (FTCS-25), Pasadena, CA, pp. June 1995, pp. 381-390
8. IBM director software rejuvenation.
   White paper
9. Recovery Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies
   David Patterson, Aaron Brown, Pete Broadwell, George Candea, Mike Chen, James Cutler, Patricia Enriquez, Armando Fox, Emre Kiciman, Matthew Merzbacher, David Oppenheimer, Naveen Sastry, William Tetzlaff, Jonathan Traupmann, Noah Treuhaft
   UC Berkeley Computer Science Technical Report UCB//CSD-02-1175, March 15, 2002
10. Reducing Recovery Time in a Small Recursively Restartable System
    George Candea, James Cutler, Armando Fox, Rushabh Doshi, Priyank Garg, Rakesh Gowda
    Appears in Proceedings of the International Conference on Dependable Systems and Networks (DSN-2002), June 2002
11. Rewind, Repair, Replay: Three R's to Dependability
    Aaron B. Brown, David A. Patterson
    To appear in 10th ACM SIGOPS European Workshop, Saint-Emilion, France, September 2002
12. Dynamic Class Loading in the Java(TM) Virtual Machine
    Sheng Liang, Gilad Bracha
    Conference on Object-oriented programming, systems, languages, and applications (OOPSLA'98)