Welcome to SENG 371 Software Evolution Spring 2013

A Core Course of the BSEng Program

Hausi A. Müller, PhD PEng
Professor, Department of Computer Science
Associate Dean Research, Faculty of Engineering
University of Victoria

Announcements

- Course website
  - http://www.engr.uvic.ca/~seng371
  - Lecture notes posted
- Labs this week
  - UML tools
  - Available in the lab
  - No need to bring your laptops or install software
- Assignment 1
  - Due Mon, Feb 4 (extension)
  - Cite your sources
  - Part I — Useful definitions
  - Part II — Growing systems in emergent organizations
  - Part III — Ultra large scale systems (ULS)

2

Job Opportunity

- http://registrar.uvic.ca/safa/documents/XMENG2.pdf

WORK STUDY PROGRAM
JOB POSTING

JOB TITLE:	MENG2 - Website Developer
DEPARTMENT NAME:	Mechanical Engineering
CONTACT NAME:	Stephanie Willerth (willerth@uvic.ca) or Brian Christie (brain64@uvic.ca)

JOB DESCRIPTION:

Will create and maintain a website containing information about regenerative medicine researchers in B.C.

QUALIFICATIONS:

Proficient in html and previous experience in working with and maintaining websites.

JOB LOCATION ON-CAMPUS:	Remote
WORK STUDY WAGE:	$11.00
HOURS AVAILABLE:	100
HOW TO APPLY:	Email C.V., links to previous work and list of references to Dr. Willerth (willerth@uvic.ca)

Applicants must be eligible for Work Study program. For details go to http://registrar.uvic.ca/safa/workstudy/workstudyindex.html

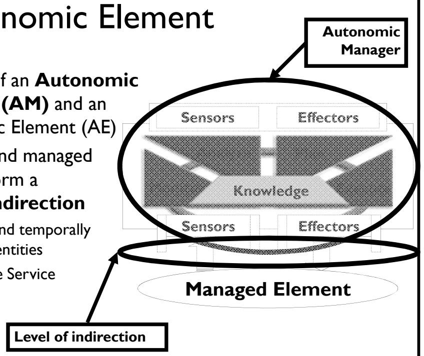
Reading assignments

- IBM Corporation: An Architectural Blueprint for Autonomic Computing, Fourth Edition (2006)
  http://people.cs.kuleuven.be/~danny.weyns/csds/IBM06.pdf
- Truex, Baskerville, Klein: Growing Systems in Emergent Organizations. Communications of the ACM, 42(8):117-123 (1999).
  http://portal.acm.org/citation.cfm?id=310930.310984&coll=GUIDE&dl=GUIDE.ACM&CFID=2240896&CFTOKEN=98671917
- Northrop, et al.: Ultra-Large-Scale Systems. The Software Challenge of the Future. Technical Report, Software Engineering Institute, Carnegie Mellon University, 134 pages ISBN 0-9786956-0-7 (2006)
  http://www.sei.cmu.edu/uls

4

Autonomic Element

- Consists of an **Autonomic Manager (AM)** and an Autonomic Element (AE)
- Manager and managed element form a **level of indirection**
  - Spatially and temporally separate entities
  - Enterprise Service Bus



5

Monitor	**Analyzer**
• Senses the managed process and its context • Collects data from the managed resource • Provides mechanisms to aggregate and filter incoming data stream • Stores relevant and critical data in the knowledge base or repository for future reference.	• Compares event data against patterns in the knowledge base to diagnose symptoms and stores the symptoms • Correlates incoming data with historical data and policies stored in repository • Analyzes symptoms • Predicts problems

MAPE-K Loop

6

Planner

- Interprets the symptoms and devises a plan
- Decides on a plan of action
- Constructs actions
  - building scripts
- Implements policies
- Often performed manually

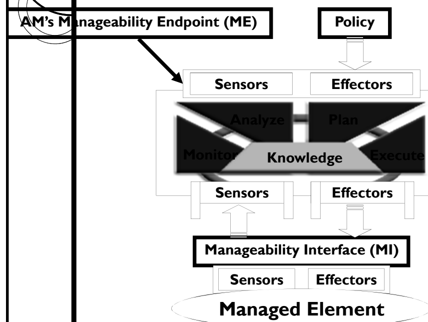
Execute Engine

- Executes the change in the managed process through the effectors
- Perform the execution plan
- Often performed manually

MAPE-K Loop

7

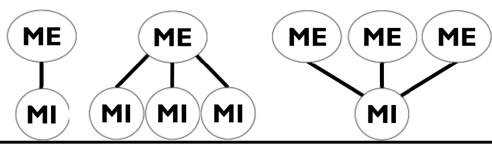
Autonomic Manager



8

Manageability Endpoints

- A Manageability Endpoint (ME) exposes the state and the management operations for a resource
- An autonomic manager communicates with a manageability endpoint through the Manageability Interface (MI)



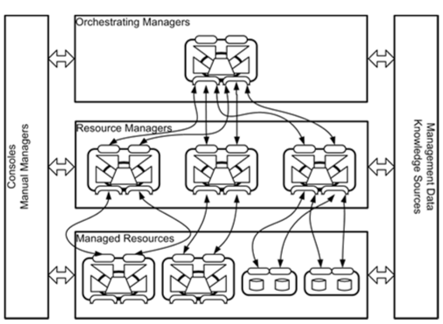
9

Manageability Interface

- An MI for monitoring and controlling a managed resource consists of sensors and effectors
- Sensors obtain data from the resource
  - read state variables in the ME
- Effectors perform operations on the resource
  - call methods in the ME
- Critical success factors for AC initiative
  - Separating AMs and MEs
  - Standardizing MIs

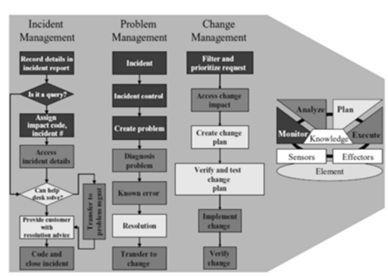
10

ACRA
Autonomic Computing Reference Architecture



11

MAPE-K Loops in IT Processes



IBM: An Architecture Blueprint for Autonomic Computing, 4th Ed. 2006

12

No Shortage of Complexity Industry Conquest Solutions

HP	Adaptive enterprise using OpenView
IBM	Autonomic computing
EDS	Agile enterprise
Hitachi	Harmonious computing
Dell	Dynamic computing
MS	Dynamic systems initiative

Industry's efforts to emulate Nature's Gold Standard of ***virtualization software*** *and* ***complexity concealment***

13

Self-Adaptive Systems Definition

- Self-adaptive software evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible
- [DARPA Broad Agency Announcement on Self-Adaptive Software (BAA-98-12) in December 1997]
- This definition is quite useful and can be extended to include other quality criteria or extra-functional/non-functional requirements (i.e., not just performance)

14

Self-Adaptive Systems Definition

- A self-adaptive software system can alter its behaviour at run-time in response to changes in its environment
- A self-adaptive system has the following abilities:
  - Accommodate dynamic change at run-time
  - Accommodate changes at run-time without shut down
  - Assess its own behaviour
  - Observe its context or environment (i.e., anything observable)

P. Oreizy, M. Gorlick, R. Taylor, D. Heimbigner, C. Johnson, N. Medvidovic, A. Quilici, D. Rosenblum, A. Wolf: An Architecture-Based Approach to Self-Adaptive Software, IEEE Intelligent Systems, pp. 54-62, 1999.

Self-Adaptive Systems My Favourite Definition

- A self-adaptive system continuously adjusts its behaviour at run-time in response to its perception of its environment and its own state in the form of fully or semiautomatic self-adaptation.
- H. Giese, Y. Brun, J. Serugendo, C. Gacek, H. Kienle, H. Müller, M. Pezzè, M. Shaw.: Engineering Self-Adaptive and Self-Managing Systems, LNCS 5527, Springer, 2009.

16

Self-adaptive Systems: Anticipated and Un-anticipated Adaptation

- Anticipated adaption
  - The different contexts to be accommodated at run-time are known at design-time
- Un-anticipated adaption
  - The variation possibilities are recognized and computed at run-time
  - The decision which variant is best is computed using self-awareness and environmental context information
- Pure un-anticipated self-adaptive system are rare
  - Most self-adaptive systems feature a combination of anticipated self-adaptation and un-anticipated self-adaptation
- Exercise: come up with a practical, technical example for each category

17

Self-Adaptive Systems Definition

- Self-adaptive software evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible
- [DARPA Broad Agency Announcement on Self-Adaptive Software (BAA-98-12) in December 1997]
- This definition is quite useful and can be extended to include other quality criteria or extra-functional/non-functional requirements (i.e., not just performance)

18

Self-Adaptive Systems Definition

- A self-adaptive software system can alter its behaviour at run-time in response to changes in its environment
- A self-adaptive system has the following abilities:
  - Accommodate dynamic change at run-time
  - Accommodate changes at run-time without shut down
  - Assess its own behaviour
  - Observe its context or environment (i.e., anything observable)

P. Oreizy, M. Gorlick, R. Taylor, D. Heimbigner, C. Johnson, N. Medvidovic, A. Quilici, D. Rosenblum, A. Wolf: An Architecture-Based Approach to Self-Adaptive Software, IEEE Intelligent Systems, pp. 54-62, 1999.

Self-Adaptive Systems My Favourite Definition

- A self-adaptive system continuously adjusts its behaviour at run-time in response to its perception of its environment and its own state in the form of fully or semiautomatic self-adaptation.
- H. Giese, Y. Brun, J. Serugendo, C. Gacek, H. Kienle, H. Müller, M. Pezzè, M. Shaw.: Engineering Self-Adaptive and Self-Managing Systems, LNCS 5527, Springer, 2009.

20

Self-adaptive Systems: Anticipated and Un-anticipated Adaptation

- Anticipated adaption
  - The different contexts to be accommodated at run-time are known at design-time
- Un-anticipated adaption
  - The variation possibilities are recognized and computed at run-time
  - The decision which variant is best is computed using self-awareness and environmental context information
- Pure un-anticipated self-adaptive system are rare
  - Most self-adaptive systems feature a combination of anticipated self-adaptation and un-anticipated self-adaptation
- Exercise: come up with a practical, technical example for each category

21

Key Questions

- What aspects of the environment should a self-adaptive system monitor?
  - The system cannot monitor everything in the environment
  - What aspects of the environment are truly relevant?
- How should a self-adaptive system react if it detects changes in the environment?
  - Maintain high-level goals
  - Relax non-critical goals to allow the system a degree of flexibility
  - Goal trade-off analysis

22

Key Questions

- What are the conditions that trigger adaptation?
- Response time
  - To address poor response times, a system might adapt itself by optimising resource utilisation
- Fault-tolerance
  - To recover from a subsystem or device failure
- Extension
  - To accommodate new functionality at run-time

P. Oreizy, M. Gorlick, R. Taylor, D. Heimbigner, C. Johnson, N. Medvidovic, A. Quilici, D. Rosenblum, A. Wolf: An Architecture-Based Approach to Self-Adaptive Software, IEEE Intelligent Systems, pp. 54-62, 1999.

Key Questions

- Should the system be open-adaptive or closed-adaptive?
  - With open-adaptive systems, new behaviours can be introduced at run-time
  - With closed-adaptive systems, all adaptive behaviour is fixed at design-time; once running a closed system cannot be made to do new things that were unanticipated when it was designed
  - Anticipated versus un-anticipated adaptation
- What type of autonomy must be supported?
  - Fully autonomous systems make their own adaptation decisions and carry them out unaided
  - Human-in-the-loop systems require inputs from humans, if only to OK proposed changes
  - Semi-autonomic versus fully autonomic systems

24

Key Questions

- Under what circumstances is adaptation cost-effective?
- The benefits gained from making a change must outweigh the costs associated with making the change
- Costs include:
  - Performance and memory overhead of monitoring system behaviour
    - Monitoring is necessary to make adaptation decisions
    - Memory may be limited on, particularly if adaptive software runs on embedded devices
  - Decision making—interpreting data gathered from monitoring may be computationally expensive
  - Executing the actions to actually change a system configuration
    - Changes involving physically distributed systems must be coordinated which itself incurs additional overhead

25

Key Questions

- How often should adaptation be considered?
  - Policies range from continuous (proactive) adaptation to as-and-when necessary (reactive)
  - Adaptation can also be opportunistic—exploiting resources such as CPU time when it is not being used for other tasks
  - "Go green" adaptation
- What kind of information must be collected to make adaptation decisions
  - Data can be gathered continuously
    - This provides precise and up-to-date observations, but incurs relatively high cost
  - Data can be gathered less often with the resulting samples being approximations of environment activity; this approach imposes less overhead
  - Trust issues

26

Major Drivers for Self-Adaptive Systems

- **Autonomic Computing**: self-managing systems
- **Ubiquitous Computing**: changing environments
  - Ubiquitous computing (ubicomp) is a post-desktop model of human-computer interaction in which information processing has been thoroughly integrated into everyday objects and activities.
  - As opposed to the desktop paradigm, in which a single user consciously engages a single device for a specialized purpose, someone "using" ubiquitous computing engages many computational devices and systems simultaneously, in the course of ordinary activities, and may not necessarily even be aware that they are doing so.
- This paradigm is also referred to as **pervasive computing**, **ambient intelligence**, or **everyware**.

Ubiquitous Computing Wiki

Useful Papers under Resources Course Web Site

- Ganek, A.G., Corbi, T.A.: The Dawning of the Autonomic Computing Era. IBM Systems Journal 42(1):5-18 (2003)
- Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. IEEE Computer 36(1):41-50 (2003)
- Kluth, A.: Information Technology: Make It Simple. The Economist (2004)
- Oreizy, P., Medvidovic, N., Taylor, R.N.: Architecture-Based Runtime Software Evolution. (Most Influential Paper Award at ICSE 2008) In: ACM/IEEE International Conference on Software Engineering (ICSE 1998), pp. 177-186 (1998)
- Huebscher, M.C., McCann, J.A.: A Survey of Autonomic Computing—Degrees, Models, and Applications. ACM Computing Surveys, 40 (3):7:1-28 (2008)
- Müller, H.A., Kienle, H.M., Stege, U.: Autonomic Computing: Now You See It, Now You Don't—Design and Evolution of Autonomic Software Systems. In: De Lucia, A.; Ferrucci, F. (eds.): Software Engineering International Summer School Lectures: University of Salerno. LNCS, Springer-Verlag, Heidelberg, pp. 32–54 (2009)
- Dobson, S., Denazis, S., Fernandez, A., Gaiti, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N., Zambonelli, F.: A Survey of Autonomic Communications. ACM Transactions on Autonomous and Adaptive Systems (TAAS) 1(2):223-259 (2006)

28

Class Participation Assignment

- Pick a self-managing scenario
- Define managed resources
- Define managing goals
- Define trade-off choices

29

Presentation notes

- Rob, Anita, Jordan, George
  - S — Wireless Network
  - R — Hardware, routers, bandwidth, connection
  - G — Availability, security, connections speed, maintenance
  - T — Availability vs user base, availability vs. maintenance, availability vs. connections speed
- Brandon, Amanda, Romil
  - S — Backup over a network
  - R — Storage, network, CPU
  - G — Back ups on time, back up is verified, security, CPU available for other tasks
  - T — Time management/size, CPU management, CPU load, network load

30

Presentation notes

- Justin, Allen, Mikko
  - S — Library
  - R — Books, Money, Computers, Staff
  - G — New books, more computers, More staff
  - T — Limited funds, limited space
- Y, Sam, Mack
  - S — Hospital
  - R — Staff, drugs, equipment, space
  - G — Maximize space efficiency, maximize staff, lower cost, raise standards
  - T — Money vs Staff, Time vs Cost

31

Presentation notes

- David, Daniel, Brad, Ian
  - S — Cloud based webservice
  - R — Servers, Database, Load balancer
  - G — Maximize speed and availability, lower cost
  - T — Speed vs Availability (limit bandwidth of users), Speed vs Cost, Availability vs Cost
- Wes, Curtis, Jeremy, Kai
  - S — Send people to space
  - R — Oxygen, Pressure, Temperature, Lights, Time
  - G — O2 > 20%, …
  - T — Temperature vs Pressure, basic chemistry

32

Presentation notes

- Paul, Scott, Marc
  - S — Grocery Store
  - R — Product stock, Back stock, Employees, Space
  - G — Keep produce fresh, conserve space, staff numbers
  - T — Order vs Back stock, More staff vs Less staff, costs and trends
- Mike, Geoff, Adam
  - S — AI for a video game (FPS)
  - R — Position, health, ammo, objectives
  - G — Survive, Aggression, Conservative, Orders
  - T — Sacrifice position and ammo to recover health, sacrifice health to capture an objective

33

Presentation notes

- Richard, Nick, Vish
  - S — Server farm
  - R — Main server, VM, ressource servers
  - G — Cost, save energy, servers
  - T — Smaller VM's, centralized server vs many servers, security

34