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
  - Will likely change to Moodle over the next few days
  - Lecture notes posted
- Mon, Feb 4
  - Norha Villegas: Context Management and Self-Adaptivity for Situation-Aware Smart Software Systems
- Assignment 1
  - Due Feb 4 (extension) due to website challenges
  - Cite your sources
  - Part I — Useful definitions
  - Part II — Growing systems in emergent organizations
  - Part III — Ultra large scale systems (ULS)

2

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

3

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.

4

Key Questions

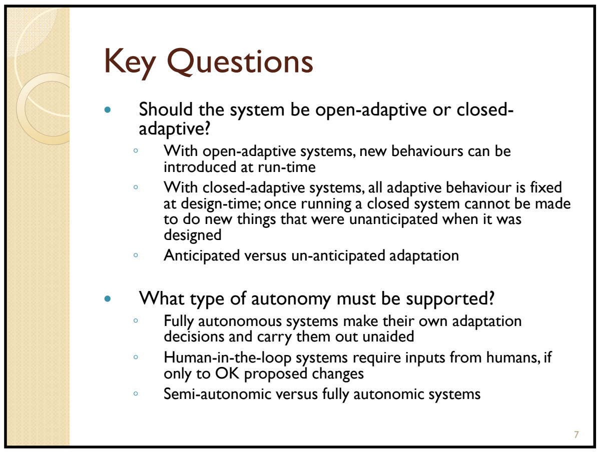
- What aspects of the environment should a self-adaptive system monitor?
  - The system cannot monitor everything in the environment
  - What aspects of the environment are truly relevant?
- How should a self-adaptive system react if it detects changes in the environment?
  - Maintain high-level goals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

5

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

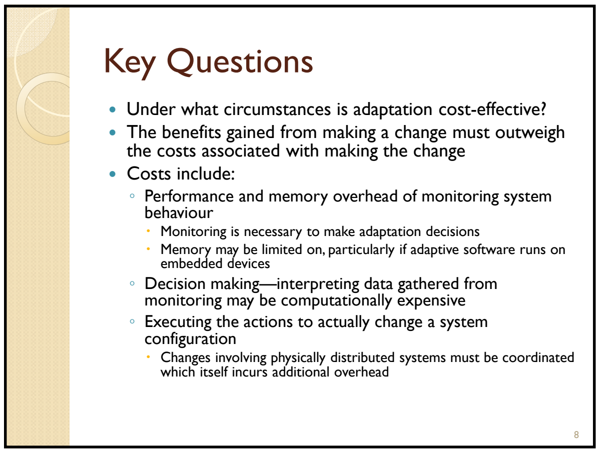
P. Oreizy, M. Gorlick, R. Taylor, D. Heimbigner, C. Johnson, N. Medvidovic, A. Quilici, D. Rosenblum, A. Wolf: An Architecture-Based Approach to Self-Adaptive Software, IEEE Intelligent Systems, pp. 54-62, 1999.



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

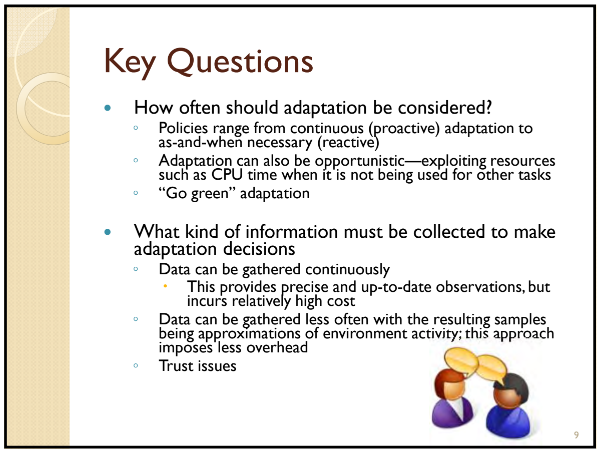
7



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

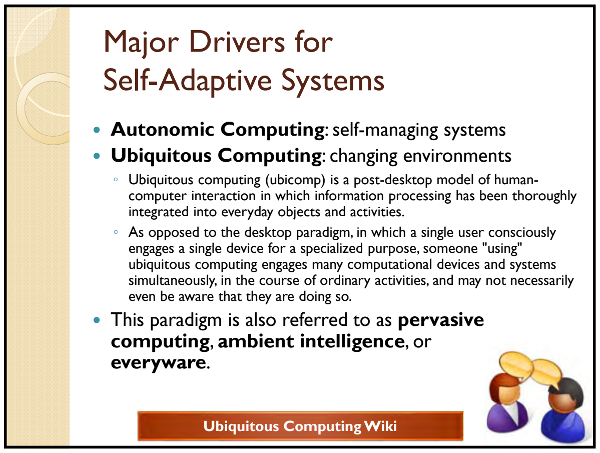
8



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

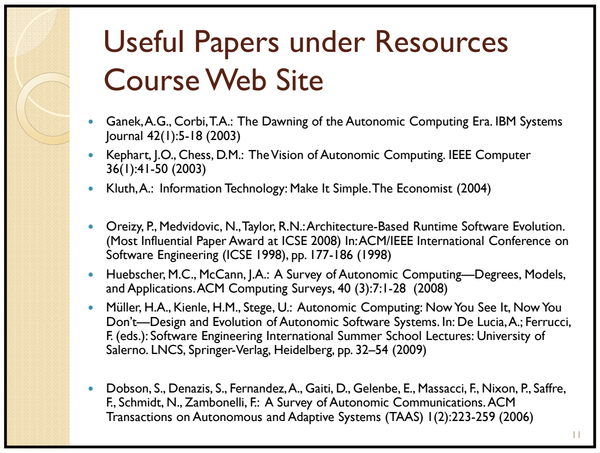
9



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**.

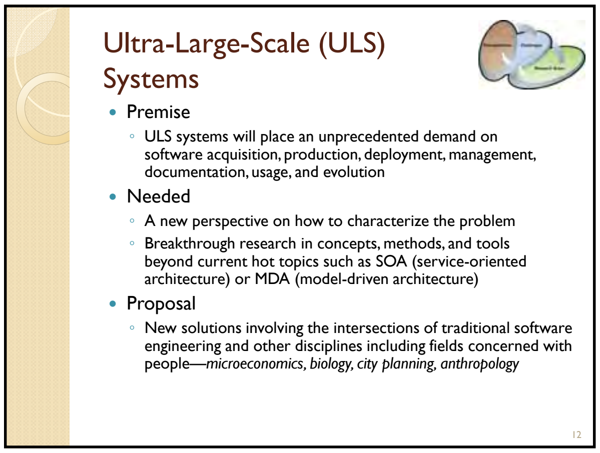
Ubiquitous Computing Wiki



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)

11



Ultra-Large-Scale (ULS) Systems

- Premise
  - ULS systems will place an unprecedented demand on software acquisition, production, deployment, management, documentation, usage, and evolution
- Needed
  - A new perspective on how to characterize the problem
  - Breakthrough research in concepts, methods, and tools beyond current hot topics such as SOA (service-oriented architecture) or MDA (model-driven architecture)
- Proposal
  - New solutions involving the intersections of traditional software engineering and other disciplines including fields concerned with people—*microeconomics, biology, city planning, anthropology*

12

ULS Sources

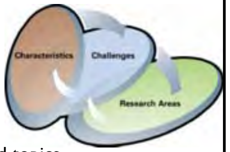
- **Scale Changes Everything**
  by Linda Northrop
  Director, Product Line Systems Program Software Engineering Institute
  OOPSLA 2006 Presentation, Oct 24, 2006

- **Ultra-Large-Scale Systems**
  The Software Challenge of the Future
  by Linda Northrop et al.
  SEI Technical Report, June 2006
  http://www.sei.cmu.edu/uls

13

ULS Research Agenda

- Describes
  - the characteristics of ULS systems
  - the associated challenges
  - promising research areas and topics

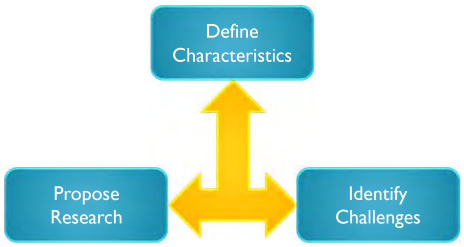


- Is based on new perspectives needed to address the problems associated with ULS systems.

L. Northrop. Scale Changes Everything. OOPSLA 2006

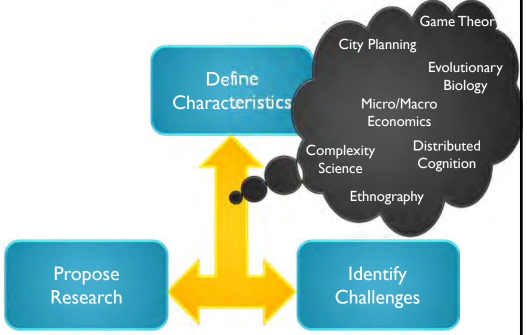
14

Research Approach



15

Research Approach



16

Characteristics of ULS Systems

- Ultra-large size in terms of
  - Lines of code
  - Amount of data stored, accessed, manipulated, and refined
  - Number of connections and interdependencies
  - Number of hardware elements
  - Number of computational elements
  - Number of system purposes and user perception of these purposes
  - Number of routine processes, interactions, and "emergent behaviours"
  - Number of (overlapping) policy domains and enforceable mechanisms
  - Number of people involved in some way
  - ……

17

What is an ULS System

- A ULS System has unprecedented scale in some of these dimensions
  - Lines of code
  - Amount of data stored, accessed, manipulated, and refined
  - Number of connections and interdependencies
  - Number of hardware elements
  - Number of computational elements
  - Number of system purposes and user perception of these purposes
  - Number of routine processes, interactions, and "emergent behaviours"
  - Number of (overlapping) policy domains and enforceable mechanisms
  - Number of people involved in some way

ULS systems are interdependent webs of software-intensive systems, people, policies, cultures, and economics.

18

Scale Changes Everything

- Characteristics of ULS systems arise because of their scale
  - Decentralization
  - Inherently conflicting, unknowable, and diverse requirements
  - Continuous evolution and deployment
  - Heterogeneous, inconsistent, and changing elements
  - Erosion of the people/system boundary
  - Normal failures
  - New paradigms for acquisition and policy

These characteristics may appear in today's systems,
but in ULS systems they dominate.
These characteristics undermine the assumptions
that underlie today's software engineering approaches.

19

Today's Approaches

- **The Engineering Perspective**—for large scale software-intensive systems
  - largely top-down and plan-driven
  - requirements/design/build cycle with standard well-defined processes
  - centrally controlled implementation and deployment
  - inherent validation and verification—at design time
- **The Agile Perspective**—proven for smaller software projects
  - fast cycle/frequent delivery/test driven
  - simple designs embracing future change and refactoring
  - small teams and retrospective to enable team learning
  - tacit knowledge

Today's approaches are based on perspectives that fundamentally do not cope with the new characteristics arising from ultra-large scale.

20

From Buildings to Cities

- Designing a large software system is like building a single, large building or a single infrastructure—power, water distribution

- Rome was not built in a day.
- It takes a long time to do a job properly.
- You should not expect to do it quickly.

Ruins under Rome: In Rome's Basement, National Geographic, 2006

21

ULS Systems Operate More Like Cities

- Built or conceived by many individuals over long periods of time (Rome)
- The form of the city is not specified by requirements, but loosely coordinated and regulated—zoning laws, building codes, economic incentives (change over time)
- Every day in every city construction is going on, repairs are taking place, modifications are being made—yet, the cities continue to function
- ULS systems will not simply be bigger systems: they will be interdependent webs of software-intensive systems, people, policies, cultures, and economics

22

New Perspectives Are Needed

"The older is not always a reliable model for the newer, the smaller for the larger, or the simpler for the more complex…Making something greater than any existing thing necessarily involves going beyond experience."

Henry Petroski
Pushing the Limits: New Adventures in Engineering

The mentality of looking backward doesn't scale.

23

Change of Perspective

- From satisfaction of requirements through traditional, top-down engineering

The system shall do this
… but it may do this …
as long as it does this …

- To satisfaction of requirements by regulation of complex, decentralized systems

How? With adaptive systems and feedback loops ☺

24