Welcome to SENG 480B / CSC 485B / CSC 586B Self-Adaptive and Self-Managing Systems

Dr. Hausi A. Müller
Professor
Department of Computer Science
University of Victoria

http://courses.seng.uvic.ca/courses/2013/summer/seng/480b
http://courses.seng.uvic.ca/courses/2013/summer/csc/485b
http://courses.seng.uvic.ca/courses/2013/summer/csc/586b

1

Announcements

- Thu, May 30
  - Assignment 1 due
- Fri, May 31
  - Assignment 2 handed out
- Mon, June 3 – Tue, June 11 (inclusive
  - Congress of the Humanities and Social Sciences
  - No classes
- Fri, June 28
  - Midterm in class

http://uviccongress2013.ca/

Midterm Questions

- Describe a feedback in excruciating detail with all the components properly named and explained
- Autonomic elements and all its components and data exchange
- What are Self* capabilities?
- ULS system properties
- Wicked problems
- Feedback loops
- Autonomic manager
- MAPE-K loop
- Describe the ACRA reference architecture
- Essay type questions

3

Reading Assignments Autonomic Computing

- Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. IEEE Computer 36(1):41-50 (2003)
- IBM Corp.: An Architectural Blueprint for Autonomic Computing, Fourth Edition (2006)
  http://people.cs.kuleuven.be/~danny.weyns/csds/IBM06.pdf
- Kluth, A.: Information Technology: Make It Simple. The Economist (2004)
  http://www.economist.com/surveys/displaystory.cfm?story_id=E1_PPDSPGP&CFID=17609242&CFTOKEN=84287974

4

Reading Assignments

- ULS Book Section 1-3 on-line at
  - http://www.sei.cmu.edu/uls/the_report.html
- Murray (Ed.): Control in an Information Rich World Report of the Panel on Future Directions in Control, Dynamics, and Systems, SIAM (2003)
  - Chapters 1 & 2
  - http://www.cds.caltech.edu/~murray/cdspanel/report/cdspanel-15aug02.pdf

5

What did you learn last week?

6

The Complexity Problem

- The increasing **complexity of computing systems** is overwhelming the capabilities of software developers and system administrators to design, evaluate, integrate, and manage these systems
- Major software and system vendors have concluded that the only viable **long-term solution is to create computing systems that manage themselves**

… an elusive goal?!?

7

☺☺☺☺☺☺☺☺☺

Given this historical perspective, maybe there is hope for the information technology sector?!

8

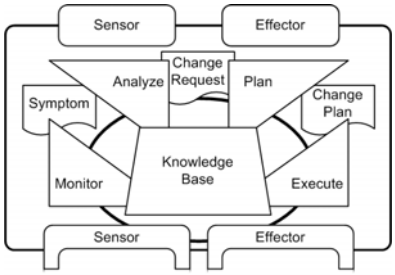
Autonomic Computing Vision

Autonomic Computing is really about making systems self-managing …

—Paul Horn, IBM Research, 2001

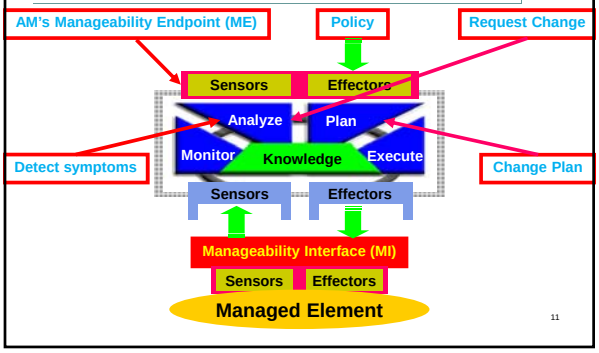
9

Autonomic Manager



10

Autonomic Manager



11

MAPE-K Loop

Monitor

- 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.

Analyzer

- 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

12

MAPE-K Loop

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

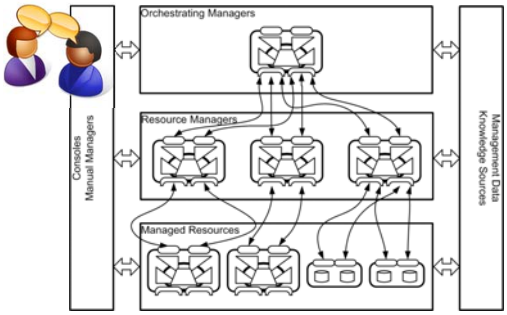
13

MAPE-K Loop Knowledge Base

- The four components of a MAPE-K loop work together by exchanging knowledge through the **knowledge base** to achieve the control objective.
- An autonomic manager
  - maintains its own knowledge
    - Information about its current state as well as past states
  - But also has access to knowledge which is shared among collaborating autonomic managers
    - Configuration database, symptoms database, business rules, provisioning policies, or problem determination expertise

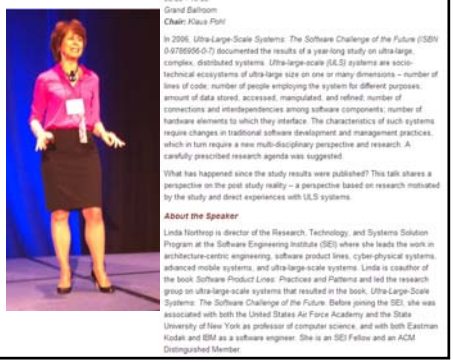
14

ACRA AC Reference Architecture



15

ICSE 2013 Keynote by Linda Northrop SEI



Linda Northrop

Software Engineering Institute - Carnegie Mellon

Does Scale Really Matter? – Ultra-Large-Scale Systems Seven Years after the Study

Friday, May 24th

08:30 - 10:30

Grand Ballroom

Chair: Klaus Pohl

In 2006, *Ultra-Large-Scale Systems: The Software Challenge of the Future* (ISBN 0-9786956-0-7) documented the results of a year-long study on ultra-large, complex, distributed systems. *Ultra-large-scale (ULS) systems* are socio-technical ecosystems of ultra-large size on one or many dimensions – number of lines of code; number of people employing the system for different purposes; amount of data stored, accessed, manipulated, and refined; number of connections and interdependencies among software components; number of hardware elements to which they interface. The characteristics of such systems require changes in traditional software development and management practices, which in turn require a new multi-disciplinary perspective and research. A carefully prescribed research agenda was suggested.

What has happened since the study results were published? This talk shares a perspective on the post study reality – a perspective based on research motivated by the study and direct experiences with ULS systems.

About the Speaker

Linda Northrop is director of the Research, Technology, and Systems Solution Program at the Software Engineering Institute (SEI) where she leads the work in architecture-centric engineering, software product lines, cyber-physical systems, advanced mobile systems, and ultra-large-scale systems. Linda is coauthor of the book *Software Product Lines: Practices and Patterns* and led the research group on ultra-large-scale systems that resulted in the book, *Ultra-Large-Scale Systems: The Software Challenge of the Future*. Before joining the SEI, she was associated with both the United States Air Force Academy and the State University of New York as professor of computer science, and with both Eastman Kodak and IBM as a software engineer. She is an SEI Fellow and an ACM Distinguished Member.

Evolution of Software Systems

- Legacy systems
- Systems of Systems

Ultra-Large-Scale (ULS) Systems
Socio-Technical Ecosystems

17

Decentralized Ecosystems

- For 40 years we have embraced the traditional centralized engineering perspective for building software
  - Central control, top-down, tradeoff analysis
- Beyond a certain complexity threshold, traditional centralized engineering perspective is no longer sufficient and cannot be the primary means by which ultra-complex systems are made real
  - **Firms** are engineered—but the structure of the **economy** is not
  - The protocols of the **Internet** were engineered—but not the **Web** as a whole
- **Ecosystems** exhibit high degrees of complexity and organization—but not necessarily through engineering

18

Characteristics of Wicked Problems

- You don't understand the problem until you have developed a solution
  - There is no definitive formulation of the problem.
  - The problem is ill-structured
  - An evolving set of interlocking issues and constraints
- There is no stopping rule
  - There is also no definitive Solution
  - The problem solving process ends when you run out of resources
- Every wicked problem is essentially unique and novel
  - There are so many factors and conditions, all embedded in a dynamic social context, that no two wicked problems are alike
  - No immediate or ultimate test of a solution
  - Solutions to them will always be custom designed and fitted
- Solutions are not right or wrong
  - Simply better, worse, good enough, or not good enough.
  - Solutions are not true-or-false, but good-or-bad.
- Every solution to a wicked problem is a one-shot operation.
  - You can't learn about the problem without trying solutions.
  - Every implemented solution has consequences.
  - Every solution you try is expensive and has lasting unintended consequences (e.g., spawn new wicked problems).
- Wicked problems have no given alternative solutions
  - May be no feasible solutions
  - May be a set of potential solutions that is devised, and another set that is never even thought of.

19

Web as Context for the Discussing ULS Challenges

- Assume the web as a ULS system
- Given the web as context, what are the implications for each of the challenges listed on the next nine slides?
- Which challenges are difficult or easy to resolve within the web context?

ULS Challenges

- The ULS book describes challenges in three broad areas:
  - Design and evolution
  - Orchestration and control
  - **Monitoring and assessment**

Chapter 3 in ULS Book

21

Specific Challenges in ULS System Monitoring and Assessment

- The effectiveness of ULS system design, operation, evolution, orchestration, and control has to be evaluated.
- There must be an ability to monitor and assess ULS system state, behavior, and overall health and well being.
- Challenges include
  - Defining indicators
  - Understanding why indicators change
  - Prioritizing the indicators
  - Handling change and imperfect information
  - Gauging the human elements

Design and evolution
Orchestration and control
→Monitoring and assessment

22

Specific Challenges in ULS System Monitoring and Assessment

- Defining indicators
  - What system-wide, end-to-end, and local quality-of-service indicators are relevant to meeting user needs and ensuring the long-term viability of the ULS system?
- Understanding why indicators change
  - What adjustments or changes to system elements and interconnections will improve or degrade these indicators?
- Prioritizing the indicators
  - Which indicators should be examined under what conditions?
  - Are indicators ordered by generality?
    - General overall health reading versus specialized particular diagnostics

Design and evolution
Orchestration and control
→Monitoring and assessment

23

Specific Challenges in ULS System Monitoring and Assessment

- Handling change and imperfect information
  - How do the monitoring and assessment processes handle continual changes to components, services, usage, or connectivity?
  - Note that imperfect information can be inaccurate, stale, or imprecise.
- Gauging the human elements
  - What are the indicators of the health and performance of the people, business, and organizational elements of the ULS system?

Design and evolution
Orchestration and control
→Monitoring and assessment

24

Unprecedented Levels of Monitoring

- To be able to observe and possibly orchestrate the continuous evolution of software systems in a complex and changing environment, we need to push the monitoring of evolving systems to unprecedented levels.

25

Runtime Check Monitors

- Monitor assertions and invariants
- Monitor frequency of raised exceptions
- Continually measure test coverage
- Data structure load balancing
- Buffer overflows, intrusion
- Memory leaks
- Checking liveness properties

26

Satisfaction of Requirements

- Perform critical regression tests regularly to observe satisfaction of requirements
- Perform V&V operations (transformations) regularly to ascertain V&V properties
- How to monitor functional and non-functional requirements when the environment evolves?

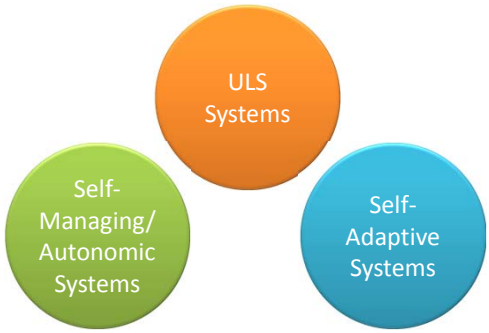
27

Monitor, Assess, and Manage System Properties

1. Govern and enforce rules and regulations
2. Monitor compliance
3. Assess whether services are used properly
4. Monitor and build user trust incrementally
5. Manage tradeoffs
6. Recognizing normal and exceptional behaviour
7. Assess and maintain quality of service (QoS)
8. Monitor service level agreements (SLAs)
9. Assess and monitor non-functional requirements

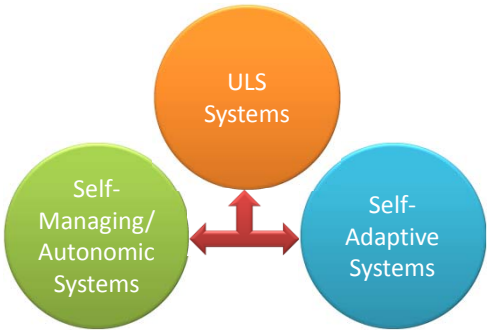
28

Related Systems

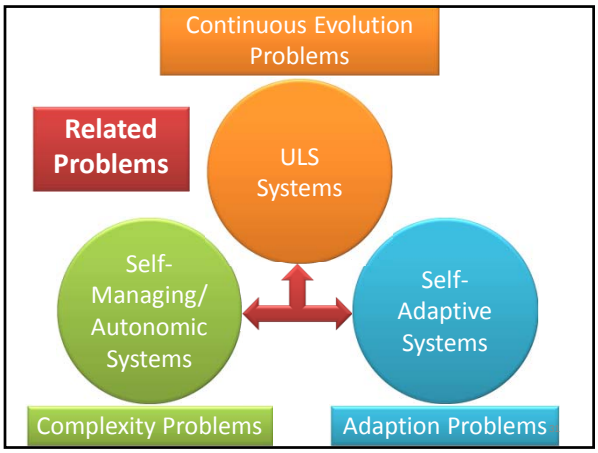


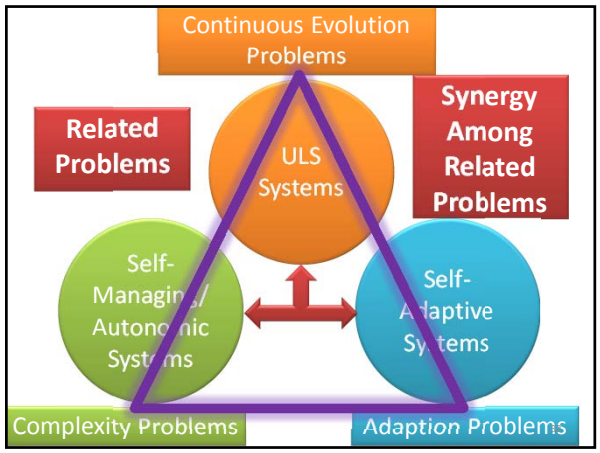
29

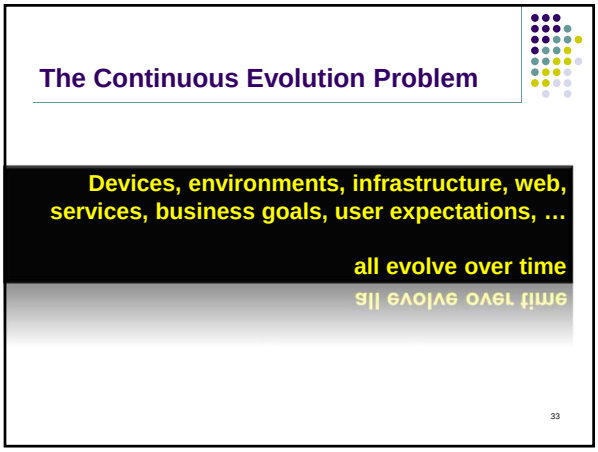
Synergy Among Related Systems



30


Continuous Evolution Problems
Related Problems
ULS Systems
Self-Managing/ Autonomic Systems
Self-Adaptive Systems
Complexity Problems
Adaption Problems


Continuous Evolution Problems
Related Problems
ULS Systems
Synergy Among Related Problems
Self-Managing/ Autonomic Systems
Self-Adaptive Systems
Complexity Problems
Adaption Problems


The Continuous Evolution Problem
Devices, environments, infrastructure, web, services, business goals, user expectations, …
all evolve over time
33