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

Dr. Hausi A. Müller
and Lorena Castañeda
Department of Computer Science
University of Victoria

http://courses.seng.uvic.ca/courses/2015/summer/seng/480a
http://courses.seng.uvic.ca/courses/2015/summer/csc/485a
http://courses.seng.uvic.ca/courses/2015/summer/csc/586a

1

Announcements

- Thursday, May 21
  - Lorena Castañeda — ULS
- Monday, May 25
  - Lorena Castañeda — ULS
  - Ron Desmarais – PID Control
- Thursday, May 28
  - Hausi Müller — Feedback loops
- Friday, May 29
  - A1 due
  - Email addresses for Part III posted
    http://www.rigiresearch.com/courses/sas/assignment-1

Additional office hours:
- ~~May 20 (Wed) 11:00 – 12:00 ECS413 w/ Lorena~~
- May 26 (Tue) 11:30 – 1:00 ECS415 w/ Ron
- May 27 (Wed) 11:00 – 12:00 ECS413 w/ Lorena
- Jun 2 (Tue) 11:00 – 12:00 ECS413 w/ Lorena
- TBA ECS 415 w/ Ron

2

Reading Assignments

- ULS Book Section 1-3 on-line at
  - http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=30519
- 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

3

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

4

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

5

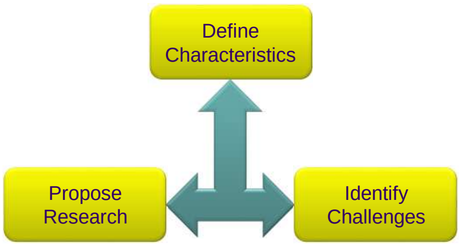
ULS Research Agenda

- 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

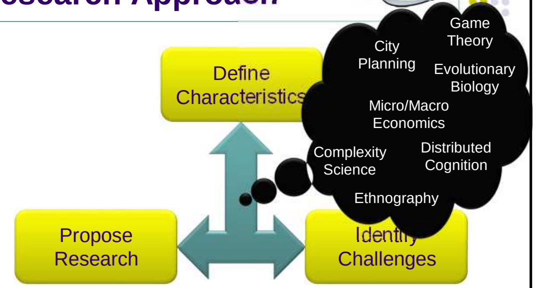
6

Research Approach



L. Northrop. Scale Changes Everything. OOPSLA 2006

Research Approach



L. Northrop. Scale Changes Everything. OOPSLA 2006

What is an ULS System

- A ULS System has unprecedented scale in some of these dimensions **(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

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

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.

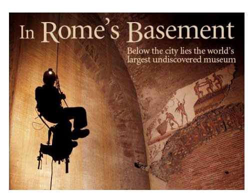
Today's Approaches

- **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
- **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.

From Buildings to Cities

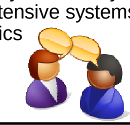
- Designing a large software system is like building a single, large building or a single infrastructure—power, water distribution



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

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



13

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.

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 ☺

15

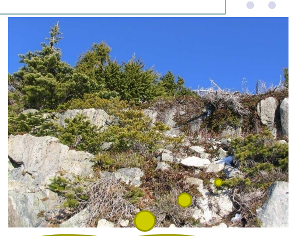
Ultra-Large-Scale (ULS) Systems

- 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

16

Evolution of Software Systems

- Legacy systems
- Systems of Systems



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

17

Definitions

- ***Ecosystem***
  - In biology, an ecosystem is a community of plants, animals, and microorganisms that are linked by energy and nutrient flows interacting with each other and with the physical environment.
  - Rain forests, deserts, coral reefs, grasslands, and a rotting log are all examples of ecosystems
- ***Socio-technical ecosystem***
  - An ecosystem whose elements are groups of people together with their computational and physical environments
  - ULS systems can be characterized as *socio-technical ecosystems*
- ***ULS system***
  - A system whose dimensions are of such a scale that constructing the system using development processes and techniques prevailing at the start of the 21st century is problematic.
  - ULS system characteristics
    - Decentralization
    - Conflicting, unknowable, and diverse requirements
    - Continuous evolution and deployment
    - Heterogeneous and changing element
    - Erosion of the people/system boundary
    - Normal failures of parts of the system

18

cf. Glossary in ULS Book

From Systems of Systems to Ecosystems

- A ULS system comprises a dynamic community of interdependent and competing organisms in a complex and changing environment
- The concept of an ecosystem connotes complexity, decentralized control, hard-to-predict reactions to disruptions, difficulty of monitoring and assessment

In many ways, legacy systems are already participating in socio-technical ecosystems

19

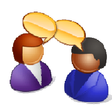
We Need to Think Socio-Technical Ecosystems

- **Socio-technical ecosystems include people, organizations, and technologies at all levels with significant and often competing interdependencies.**
- In such systems there is
  - Competition for resources
  - Organizations and participants responsible for setting policies
  - Organizations and participants responsible for producing ULS systems
  - Need for local and global indicators of health that will trigger necessary changes in policies and in element and system behavior

20

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



21

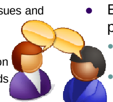
ULS Systems Solve Wicked Problems

- ***Wicked problem***
  An ill-defined design and planning problem having incomplete, contradictory, and changing requirements.
- Solutions to wicked problems are often difficult to recognize because of complex interdependencies.
- This term was suggested by H. Rittel & M. Webber in "Dilemmas in a General Theory of Planning," *Policy Sciences 4, Elsevier* (1973)
- Wicked problems are problems that are not amenable to *analytic, reductionist analysis.*

22

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.



23

An Architecture for Dealing with Wicked Problems

- A dynamic hierarchy, constellation, or arrangement of interacting system architectures
- Each dynamic arrangement has its own
  - Value propositions
  - Element types (including individuals and organizations) and associated properties (such as self-interest and private values)
  - Relations
    - For example, those found in strategic games
  - Theories
    - For example, game theory

Mark Klein, SEI, 2008

24

Why a New Perspective?

- There are fundamental assumptions that underlie today's software engineering and software development approaches that are **undermined by the characteristics of ULS systems.**
- There are challenges associated with ULS systems that today's perspectives are very unlikely to be able to address.

For the last forty years, engineering has been the dominant metaphor for software systems creation.

25

ULS Systems vs. Today's Approaches

Characteristics	Characteristics Today's assumptions undermined
Decentralized control	All conflicts must be resolved and resolved centrally and uniformly.
Inherently conflicting, unknowable, and diverse requirements	Requirements can be known in advance and change slowly. Trade-off decisions will be stable.
Continuous evolution and deployment	System improvements are introduced at discrete intervals.
Heterogeneous, inconsistent, and changing elements	Effect of a change can be predicted sufficiently well. Configuration information is accurate and can be tightly controlled. Components and users are fairly homogeneous.

26

ULS Systems vs Today's Approaches

Characteristics	Characteristics Today's assumptions undermined
Erosion of the people/system boundary	People are just users of the system. Collective behavior of people is not of interest. Social interactions are not relevant.
Failures are normal	Failures will occur infrequently. Defects can be removed.
New paradigms for acquisition and policy	A prime contractor is responsible for system development, operation, and evolution (e.g., open source, community development of data and code)

27

Next class

ULS Challenges

Chapter 3 in ULS Book

Monday, May 25
- Lorena Castañeda — ULS
- Ron Desmarais – PID Control

28

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

Dr. Hausi A. Müller
and Lorena Castañeda
Department of Computer Science
University of Victoria

http://courses.seng.uvic.ca/courses/2015/summer/seng/480a
http://courses.seng.uvic.ca/courses/2015/summer/csc/485a
http://courses.seng.uvic.ca/courses/2015/summer/csc/586a

29

Announcements

- ~~Thursday, May 21~~
  - ~~Lorena Castañeda — ULS~~
- Monday, May 25
  - Lorena Castañeda — ULS
  - Ron Desmarais – PID Control
- Thursday, May 28
  - Hausi Müller — Feedback loops
- Friday, May 29
  - A1 due
  - Email addresses for Part III posted
    http://www.rigiresearch.com/courses/sas/assignment-1

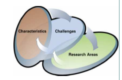
Additional office hours:
- ~~May 20 (Wed) 11:00 – 12:00 ECS413 w/ Lorena~~
- May 27 (Wed) 11:00 -12:00 ECS413 w/ Lorena
- June 1 (Mon) 1:00 – 2:00 ECS 415 w/ Ron
- June 2 (Tue) 11:30 – 1:00 ECS415 w/ Ron

30

Reading Assignments

- ULS Book Section 1-3 on-line at
  - http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=30519
- 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

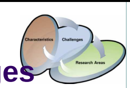
ULS Challenges



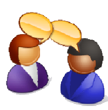
- The ULS book describes challenges in three broad areas:
  - **Design and evolution**
  - Orchestration and control
  - Monitoring and assessment

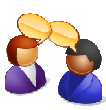
Chapter 3 in ULS Book

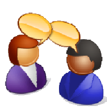
Web as Context for the Discussing ULS Challenges

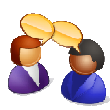


- Assume the web as a ULS system
- Given the web as context, what are the implications for each of the challenges listed on the next nine slides?
- Which challenges are difficult or easy to resolve within the web context?







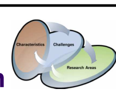


Specific Challenges in ULS System Design and Evolution

- Social activity for constructing computational environments
  - How do we model interaction with a social context in a way that offers guidance for how to design and support ULS systems?
- Legal issues
  - How do we deal with licensing, intellectual property, or liability concerns that arise due to the size, complexity or geographical distribution of a ULS system developed under multiple authorities? How will legal policies adapt?
- Enforcement mechanisms and processes
  - How do we create enforcement mechanisms (i.e., governance) for a set of (legal, design, process) rules that support and maintain the integrity of the system? How do we negotiate exceptions (e.g., for SOA governance)?
- Definition of common services supporting the ULS system
  - How do we define an infrastructure (a set of technological, legal and social services) that will be common to many elements of a ULS system?

→Design and evolution
Orchestration and control
Monitoring and assessment

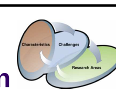
Specific Challenges in ULS System Design and Evolution



- Rules and regulations
  - How will whole industries come together to agree on rules and regulations to ensure overall coherence and quality while still being sufficiently flexible to compete?
- Agility
  - How can the groups responsible for ULS development, maintenance, and evolution be kept sufficiently agile to respond effectively to changes in requirements, system configuration, or system environment?
- Handling of change
  - How can the processes for developing, maintaining, and evolving a ULS system be adapted to handle in situ design change and evolution rather than relying on static requirements preceding design and implementation?
- Integration
  - How can we minimize the effort needed to integrate components built independently by different teams, with different goals, and at different times to create the current system?

→Design and evolution
Orchestration and control
Monitoring and assessment

Specific Challenges in ULS System Design and Evolution



- User-controlled evolution
  - How do we provide components and composition rules that give users the ability to create new, unplanned capabilities?
- Computer-supported evolution
  - How do we provide automated methods to evolve ULS systems?
- Adaptable structure
  - How do we create designs that are effective and robust even as requirements and the ULS environment change continually?
- Emergent quality
  - How do we organize processes for producing ULS systems so that they converge on high-quality designs? How do we recognize emergent quality?

→Design and evolution
Orchestration and control
Monitoring and assessment

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

37

Specific Challenges in ULS System Orchestration and Control

- Refers to the set of activities needed to make the elements of a ULS system work together in reasonable harmony to ensure continuous satisfaction of mission objectives
- Orchestration is needed at all levels of ULS systems and challenges us to create new ways for
  - Online modification
  - Maintenance of quality of service while providing necessary flexibility
  - Creation and execution of policies and rules
  - Adaptation to users and contexts
  - Enabling of user-controlled orchestration

Design and evolution
→Orchestration and control
Monitoring and assessment

38

Specific Challenges in ULS System Orchestration and Control

- Online modification
  - How can necessary adjustments to a system be made while the system is running, with minimal disturbance to user services?
  - How can the changes be propagated throughout the system if necessary?
- Maintenance of quality of service while providing necessary flexibility
  - How can the overall quality of service be maintained while enabling the flexibility to provide different levels of service to different groups?
- Creation and execution of policies and rules
  - What policies and rules lead to effective solutions despite divergent viewpoints of stakeholders?
  - How are such rules and policies created?
  - How are they executed?

Design and evolution
→Orchestration and control
Monitoring and assessment

39

Specific Challenges in ULS System Orchestration and Control

- Adaptation to users and contexts
  - How can the needs of users and stakeholders be discovered and understood?
  - How can those needs be translated into execution-time modifications and adaptations?
  - How can the context—both the user's context and the physical context—be sensed, captured, and translated into adaptations?
- Enabling of user-controlled orchestration
  - How do we provide components and composition rules that give users the ability to adapt and customize portions of the system in the field?

Design and evolution
→Orchestration and control
Monitoring and assessment

40

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

41

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

42

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

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?

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.

Run-Time Check Monitors

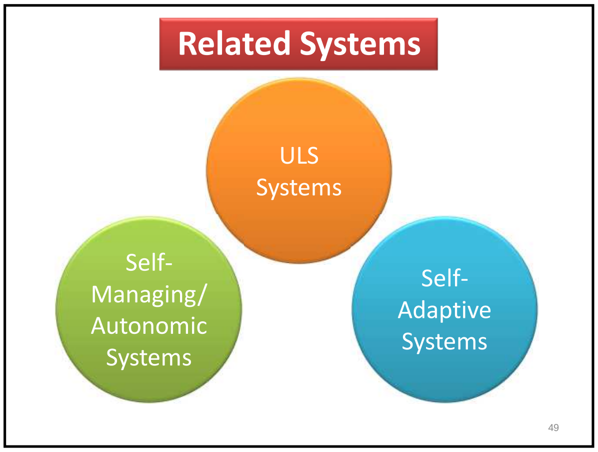
- Monitor assertions and invariants
- Monitor frequency of raised exceptions
- Continually measure test coverage
- Data structure load balancing
- Buffer overflows, intrusion
- Memory leaks
- Checking liveness properties

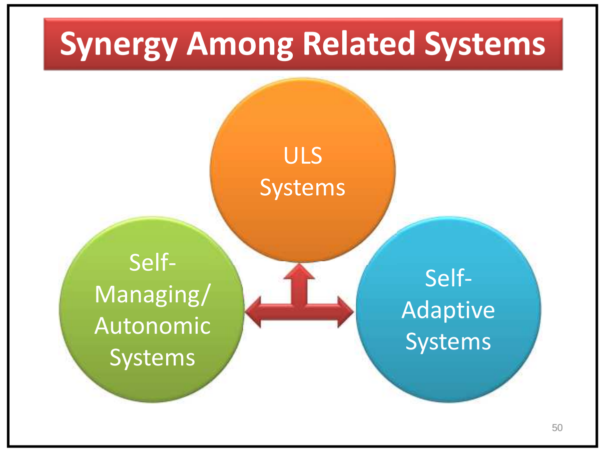
Satisfaction of Requirements

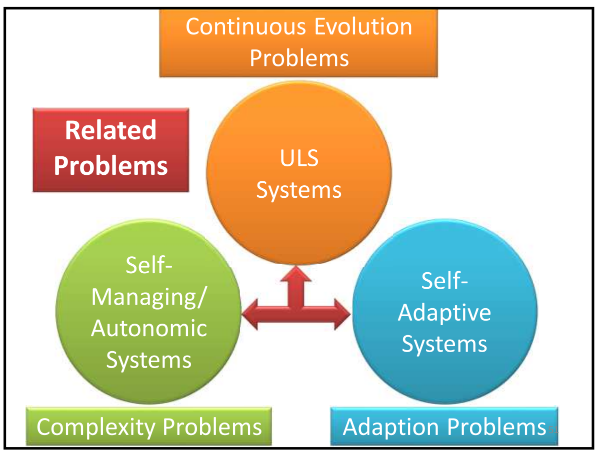
- Perform critical regression tests regularly to observe satisfaction of requirements
- Perform V&V operations (transformations) regularly to ascertain V&V properties
- How to monitor functional and non-functional requirements when the environment evolves?

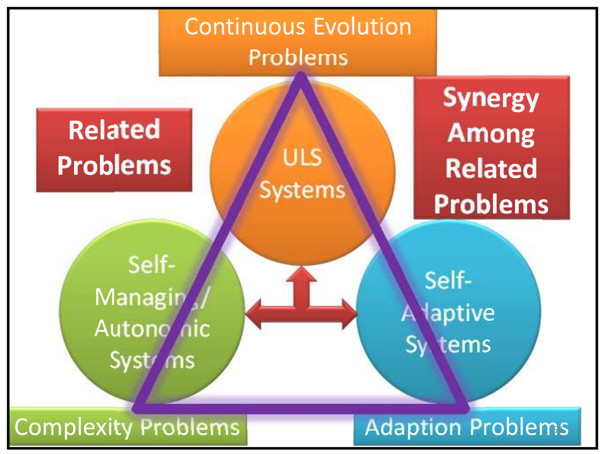
Monitor, Assess, and Manage System Properties

- Govern and enforce rules and regulations
- Monitor compliance
- Assess whether services are used properly
- Monitor and build user trust incrementally
- Manage tradeoffs
- Recognizing normal and exceptional behaviour
- Assess and maintain quality of service (QoS)
- Monitor service level agreements (SLAs)
- Assess and monitor non-functional requirements









The Continuous Evolution Problem

Devices, environments, infrastructure, web, services, business goals, user expectations, …

all evolve over time

53

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

54