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
- Mon, Feb 4
  - Norha Villegas: Context Management and Self-Adaptivity for Situation-Aware Smart Software Systems
- Assignment 1
  - Due Feb 4 (extension) due to submission challenges
  - Assignment 1 instructions have been updated
  - Submit by e-mail to seng371@uvic.ca — ideally one .pdf file
  - 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

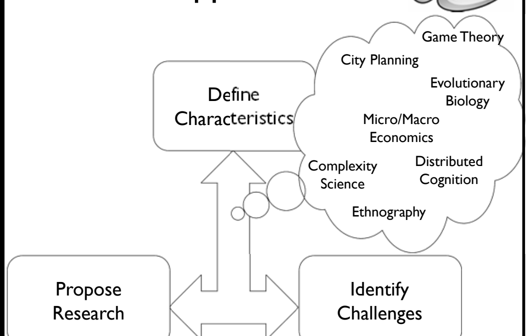
3

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

Research Approach



5

Scale Changes Everything

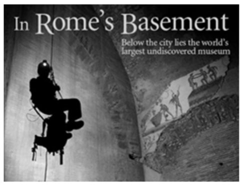
- Characteristics of ULS systems arise because of their scale
  - Decentralization
  - Inherently conflicting, unknowable, and diverse requirements
  - Continuous evolution and deployment
  - Heterogeneous, inconsistent, and changing elements
  - Erosion of the people/system boundary
  - Normal failures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

6

From Buildings to Cities

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



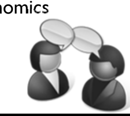
- Rome was not built in a day.
- It takes a long time to do a job properly.
- You should not expect to do it quickly.

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

7

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



8

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.

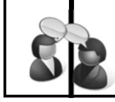
9

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

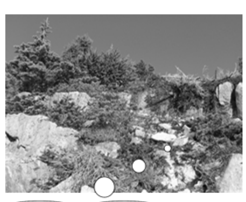


How? With adaptive systems and feedback loops ☺

10

Evolution of Software Systems

- Legacy systems
- Systems of Systems



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

11

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

cf. Glossary in ULS Book

12

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

13

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

14

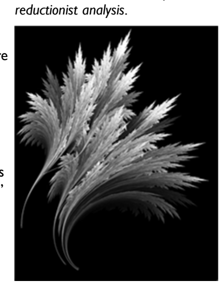
Decentralized Ecosystems

- 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

15

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*.



16

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.

17

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

18