

Welcome to SENG 371 Software Evolution Spring 2013 A Core Course of the BEng 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
 - Lab slides and activities are posted
- Mon, Feb 4
 - Norha Villegas: Context Management and Self-Adaptivity for Situation-Aware Smart Software Systems
- Assignment I
 - Due Feb 4 (extension) due to submission challenges
 - Assignment I 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

Scale Changes Everything



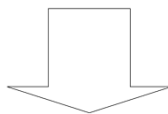
- Characteristics of ULS systems arise because of their scale
 - Decentralization
 - Inherently conflicting, unknowable, and diverse requirements
 - Continuous evolution and deployment
 - Heterogeneous, inconsistent, and changing elements
 - Erosion of the people/system boundary
 - Normal failures
 -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

4

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 😊

5

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

6

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

7

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.

8

An Architecture for Dealing with Wicked Problems

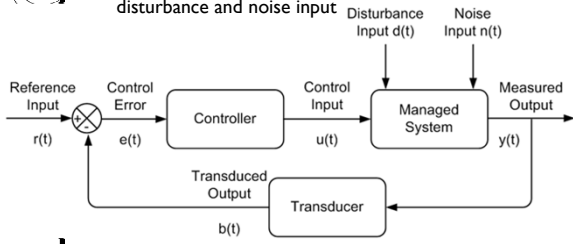
- A dynamic hierarchy, constellation, or arrangement of interacting system architectures
- Each dynamic arrangement has its own
 - Value propositions
 - Element types (including individuals and organizations) and associated properties (such as self-interest and private values)
 - Relations
 - For example,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

Mark Klein, SEI, 2008

9

Realization of a Dynamic Architecture

- Feedback control system with disturbance and noise input



Hellerstein, Diao, Parekh, Tilbury: Feedback Control of Computing Systems. John Wiley & Sons (2004)

10

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.

11

ULS Systems vs. Today's Approaches

ULS Characteristics	Today's assumptions
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.

12

ULS Systems vs. Today's Approaches

ULS Characteristics	Today's assumptions
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)

13

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

14

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?

15

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

16

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

17


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

18

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

19

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

20

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

21

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

22