Managing Software and System Complexity
Sarah Sheard, Ph.D.

Software Solutions Conference 2015
November 16–18, 2015
Managing Software and System Complexity

Growth of Complexity

What is Complexity?

Example: Systems of Systems complexity

Contributing Factors

What Worsens Complexity?

Reducing Objective Complexity

Reducing Subjective Complexity

Conclusion
Motivation

*Any sufficiently advanced technology is indistinguishable from magic.*

- Arthur C. Clarke

*Any technology distinguishable from magic is insufficiently advanced.*

- Gehm's Corollary
Growth of Complexity

Every year, systems are more complex than last year

Capability grows…our systems do more thinking

Number of systems to interoperate with grows

Systems to be redundant, resilient, adaptable, and secure to a variety of digital threats

Results in growing complexity
What Is Complexity?

Complexity is a state or quality of being composed of many intricately interconnected parts, in a manner that makes it difficult for humans, supplemented by tools, to understand, analyze, or predict behavior.

Objective Complexity
Characteristics of technical system

Subjective Complexity
Characteristics of human experience with the system
Aspects of Objective Complexity

**Size** (number of elements, requirements, users…)

**Interconnectedness** (number of links)

**Heterogeneity** (heterogeneity of elements, and of links; multi-scale important elements)

**Change** (short term dynamics – butterfly effect, behavior – and long term dynamics – evolution)

**Sociopolitical complexity** (Stakeholder conflict, stakeholder changes)

-Sheard 2012 Dissertation, see http://seir.sei.cmu.edu/sheard/
Aspects of Subjective Complexity

- Difficulty understanding
- Difficulty determining cause and effect
- Confusion and frustration
- Unpredictability
- Inability to list all the major problems
- Difficulty teasing a problem apart into component sub-problems
## Concepts Related to Complexity

<table>
<thead>
<tr>
<th>Are These Objective or Subjective?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of code</td>
</tr>
<tr>
<td>Unmaintainable</td>
</tr>
<tr>
<td>Difficult to verify</td>
</tr>
<tr>
<td>Nonlinear behavior</td>
</tr>
<tr>
<td>Strong coupling</td>
</tr>
<tr>
<td>Open system</td>
</tr>
<tr>
<td>Unclear system boundary</td>
</tr>
<tr>
<td>Not possible to understand</td>
</tr>
<tr>
<td>Can have cascading failures</td>
</tr>
<tr>
<td>System of systems</td>
</tr>
<tr>
<td>Heterogeneous elements</td>
</tr>
<tr>
<td>User cognitive load</td>
</tr>
</tbody>
</table>
Objective → Causes → Subjective

Many pieces
Tightly coupled pieces
Nonlinear behavior
SoS with many stakeholders

Causes

Uncertain
Risky
Hard to understand
Unpredictable
Frustrating
Uncontrollable
What Can Be Complex?

Systems
- Software, computer hardware, other hardware
- Safety case

Projects
- Teams, process, constraints, laws, deadlines, …

Environments
- Technical – interfacing systems
- Sociopolitical
Is Complexity Bad?

Yes
In performance, schedule, cost: a more complex system is worse than a simpler system

No
More complex systems can have more intelligent functionality

Complexity is in all cases *difficult* to deal with.
It may be *necessary*. 
Splitting Complexity

Split complexity between technical system and operator

- Technical system can handle complexity that could confuse an operator (e.g. calculate position from sensor data)

Goals:

- System shouldn’t be so simple it leaves all complexity to the operator

- System shouldn’t be so complex it hides issues and leaves the operator completely out of the loop
Complexity Changes with Time

Complexity, however defined *objectively*, relentlessly increases

Complexity, defined *subjectively*, relentlessly decreases (for a given system)
Example: Complexity of a System of Systems

Purpose: to create a multi-purpose, multi-user technology

Organization: One enterprise including several ACAT 1 programs

• Each program has cost, schedule, performance issues and makes adjustments per own priorities

• Each program can hold up all the others

Desire top-down control; best you can get is agreements
Example: Contributing Factors to Complexity

- This system will be launched into orbit, and we can’t fix it after that
- Dealing with legacy systems
- Conflict between cheapest now and most adaptable for the future
- Stakeholders change
- New standards came out since we started designing this

- Example: Internet of Things
Reasons for Today’s Complexity Increase

Environment evolves

- Distributed systems, distributed development
- Requirements
  - Interoperability
  - Safety
  - Resilience, flexibility and adaptability
- Evolving nature of security threats and countermeasures
- Tools and environments allow it, e.g., optimizing compilers, higher-order languages.

What Makes Complexity Worse?

- Using more components
- Patching after-the-fact
- Requirement changes approved midway through the program
- Using components about which little is known (Legacy, COTS)
- Inadequate time for engineers to think (“Just do the process”)
- Multiple stakeholders, especially when they change their minds
- Changing relationships among stakeholders
- Lack of clarity in architecture
- Lack of clarity regarding which experts are the final authority

- Experience
Dealing With Complexity

Reduce objective complexity
Often: Architecture

Reduce subjective complexity
Many tools and techniques from project management and systems engineering

Treat remaining complexity as risks
Dealing With Complexity: 2

Project management and systems engineering activities

- Planning, tracking
- Identification and prioritization of requirements
- Allocation of tasks to people
- Allocation of requirements to components
- Trade studies
- Communication with customer representatives
- Risk management

Identify and address “system of systems” engineering concerns
Identify systems and software architectures that best address needed qualities
Evolve the right design by adapting proven designs
Reducing Objective Complexity

Use fewer components/pieces
Use fewer *kinds of* components or pieces
Decouple (caution: this may reduce capability)
Reduce the frequency of change (defer to “next version”)
Improve clarity of communication
Reducing Subjective Complexity

Establish capable modeling, configuration management, and design environment

Probe, proof the new technology: Prototype

Learn from others who are using it

Improve clarity of communication

Train engineers and give them time to think
Treat Remaining Complexity as Risks

- Identification
- Analysis
  - What, how, when
  - Any cascading effects
  - How likely and how bad
- Evaluation
- Mitigation
- Monitoring
Conclusion

Objective complexity relates to the system and can be measured
• Systems become more objectively complex over time

Subjective complexity varies with person and time
• A system becomes less subjectively complex with familiarity and tools

Reduce complexity with standard tools and techniques
• Architecture; planning, tracking, communicating

Treat remaining complexity as a risk: identify, analyze, mitigate
Contact Information

Presenter
Sarah A. Sheard
Senior Engineer
Telephone: +1 412.268.7612
Email: sheard@sei.cmu.edu