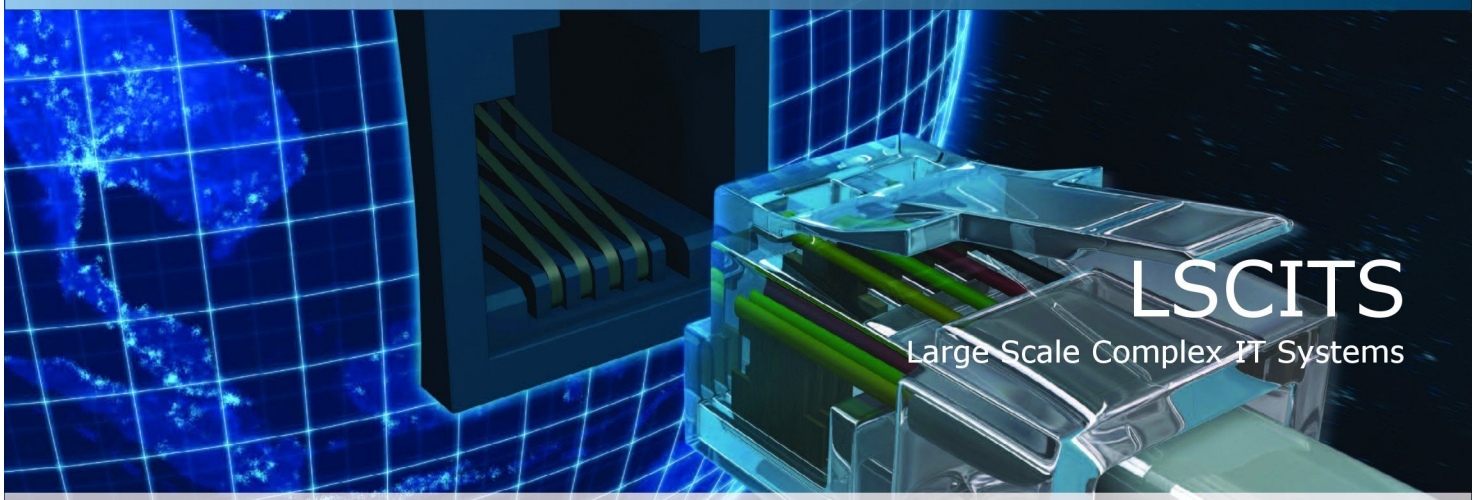


Software Engineering in the System-of-Systems Era

The functionality and flexibility underpinning today's applications in areas ranging from transportation and healthcare to aerospace and defence can no longer be provided by monolithic information systems. Instead, the required capabilities can only be achieved through employing collections of collaborative, heterogeneous and autonomously-operating systems—or *systems of systems*.

The Challenges of Systems of Systems

System of systems are characterised by unprecedented size, diversity, variability, complexity, unforeseen interactions and emergent behaviour.

The components of a system of systems (SoS) possess a level of operational autonomy that allows them to pursue their own, local objectives, independently and in addition to contributing to the global SoS objective(s). SoS components are often developed, procured and managed independently, and may belong to multiple open and evolving systems of systems that they could join and leave dynamically. An SoS subclass that typifies key information systems of the future—termed *large-scale complex IT systems*—is additionally characterised by incomplete and continually changing requirements and components, and by *normal failures*.

The challenges associated with the development of systems of systems are tremendous. They include the need to ensure the interoperability of a vastly diverse range of components, to convey global objectives to SoS components in meaningful ways, to achieve these objectives predictably and dependably in a dynamically changing environment, and to attain high levels of SoS longevity.

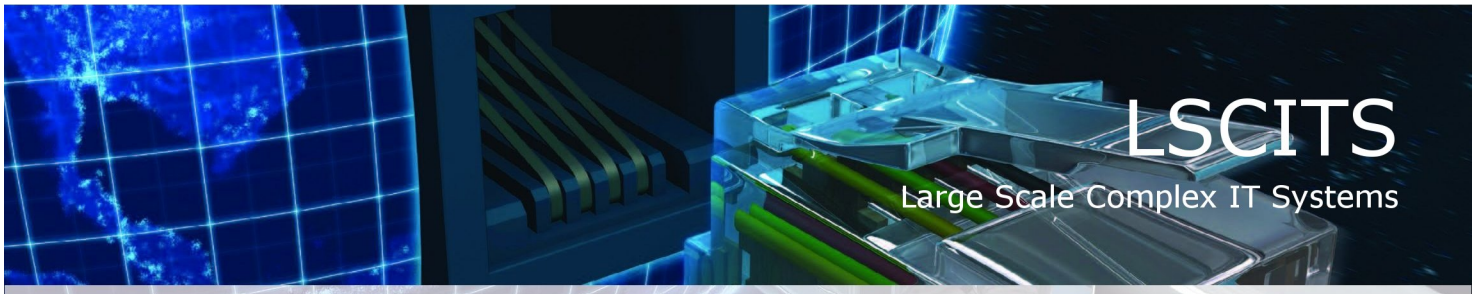
The decision to adopt an SoS solution, and hence to accept these challenges, often involves a degree of necessity. Thus, the planned application may have an inherently SoS nature, or may require the use of commercial, off-the-shelf (COTS) systems and proprietary legacy systems as part of the overall solution. The primary justification, however, is the need to leverage the *emergence* of systems of systems, i.e., their ability to provide capabilities and levels of flexibility significantly beyond those provided by the sum of their components.

How Can Software Engineering Help

Existing techniques and tools are unable to address the whole spectrum of challenges associated with the development of systems of systems. Notwithstanding the disparity between what can be achieved using current approaches and these challenges, the SoS development frameworks of the future are bound to incorporate some of today's system development techniques or adapted variants of them. Software engineering paradigms that are promising candidates for this role are described below, and summarised in Table 1.

Service-oriented architectures (SOA) SoS development involves the integration and secure interoperation of vastly diverse technical systems. Thanks to their platform independence, loose coupling and support for security, SOA solutions represent strong candidates for implementing new computer systems or front-ends to legacy systems that need to be integrated into an SoS.

Policy-based autonomic computing Ecosystems, cities and economies are often pointed out as examples of effective systems of systems. A common characteristic of all these systems of systems is the way in which their global objectives are specified through high-level incentives, rewards and penalties rather than by setting concrete, precise targets. Thus, the behaviour of ecosystems is governed by laws of nature. The development and everyday life of cities are subject to common or civil laws and regulations. The evolution of economies is guided by taxation policies. If these successful real-world examples are to be followed, techniques will be required that can



Software Engineering in the System-of-Systems Era

convey the global objectives of systems of systems as *high-level policies* to their autonomous components. (Policy-based) autonomic computing addresses the development of systems that can manage themselves based on a set of high-level policies, and therefore represents an ideal paradigm for developing the computer-system components of an SoS.

Formal verification A major concern of systems of systems is their ability to achieve an overall objective in predictable and dependable ways, through the collaboration of component systems with different (and potentially conflicting) local goals. Formal verification, and in particular model checking and probabilistic model checking, comprise a range of techniques that could be used or adapted for use in the verification of SoS policies, and ultimately for SoS dependability management and assurance.

Model-driven development and code generation The open, evolving nature of systems of systems allows their components to join and leave dynamically. Having SoS components collaborate with peer systems whose characteristics are often unknown until runtime is a major challenge. A combination of model-driven development and runtime code generation in which a dynamically acquired model of a peer system is used to generate the necessary interfaces and logic for collaborating with this peer represents a promising approach to addressing this challenge.

Component-based development SoS engineering requires the integration of existing and future commercial, open-source and proprietary systems, and component-based development provides techniques that can help achieve this goal.

Dynamic reconfiguration Systems of systems are required to adapt continually to changes in their environment, structure and objectives. Recent advances in the study of dynamically reconfigurable software and hardware provide promising approaches for the development of the computer systems to be incorporated into the systems of systems of the future.

Online machine learning The levels of self-management that SoS components must achieve in impossible to anticipate circumstances are significantly beyond what can be pre-programmed into a computer system. The online use of techniques specific to machine learning is therefore likely to play a major role in the development of computer-based SoS components.

Resource discovery In the era of mobile computing, SoS components are expected to actively seek partner systems and establish collaborations with them, thus joining (and leaving) loosely-coupled federations of systems on a regular basis. The rich spectrum of resource discovery techniques employed by today's distributed (e.g., grid- and web-based) computer systems can be used as a basis for the development of techniques to support these capabilities.

Table 1. Software engineering techniques for SoS development

Techniques and paradigms	interoperability, security	dependability (assurance)	collaboration	global-objective specific.	predictability	adaptability	longevity	flexibility
	SoS challenges							
Service-oriented architectures	✓							
Policy-based autonomic computing				✓				✓
Formal verification		✓			✓			
Model-driven development/code gen.						✓		
Component-based development			✓					
Dynamic reconfiguration						✓	✓	✓
Online machine learning						✓	✓	
Resource discovery			✓					✓

Further Information

1. Large-Scale Complex IT Systems Initiative web site, http://www.lscits.org.
2. R. Calinescu and M. Kwiatkowska. Software engineering techniques for the development of systems of systems. In *Foundations of Computer Software: Future Trends and Techniques for Development, Proc. 15th Monterey Workshop*, 2008 (to appear).
3. Carnegie Mellon Software Engineering Institute. *Ultra-Large-Scale Systems. The Software Challenge of the Future*, 2006.
4. Greg Goth. Ultralarge systems: Redefining software engineering? In *IEEE Software*, 25(3):91-94, May/June 2008.

EPSRC Engineering and Physical Sciences Research Council