

Emergent Issues in Interoperability

David A. Fisher & Dennis Smith

Standard software engineering practice assumes that the requirements for a system are knowable and that the art of software development attempts to develop a system with careful fidelity to these requirements. We refer to systems that are developed with full knowledge and that can be controlled centrally as “closed systems.”

With modern systems development and the need to develop complex systems of systems, most systems are no longer “closed”; rather they are “unbounded” because they involve an unknown number of participants or otherwise require individual participants to act and interact in the absence of needed information.

In this column we discuss the distinction between unbounded and closed systems. We then outline the impact of unbounded systems on interoperability, and suggest that unbounded systems exhibit emergent properties that cannot be fully known in advance. We conclude with an initial set of implications for the interoperability problem.

Unbounded Systems Versus Closed Systems

Unbounded systems of systems are fast becoming the norm in many of the most demanding military and commercial applications. These include command-and-control systems, air traffic control systems, the electric power grid, the Internet, individual aircraft, enterprise database systems, and modern PC operating systems. For example, in net-centric warfare as applied by U.S. troops at the beginning of the current war in Iraq, agility and rapid progress were achieved by direct interactions among ground troops, helicopters, artillery, and bombers using equipment whose designs did not anticipate such usage and the accompanying mission changes.

Most systems of systems use their component systems in ways that were neither intended nor anticipated. Assumptions that were reasonable and appropriate for individual component systems become sources of errors and malfunction within systems of systems. As a result, the individual systems – and the system of systems as a whole -- acquire vulnerabilities that can be triggered accidentally by normal actions of users and automated components, or exploited consciously by intelligent adversaries. For the complex systems of systems being constructed today and defined for the future, it is no longer possible for any human or automated component to have full knowledge of the system. Each component must depend on information received from other systems whose capabilities, intentions, and trustworthiness are unknown.

Unlike closed automated systems and traditional mathematics where neither correct nor useful results are required in the absence of complete and correct data, unbounded systems must function effectively with incomplete data and with data that cannot be fully trusted. Unfortunately, the primary mechanisms for

achieving data integrity and trust in closed, tightly coupled, and fully understood systems will not achieve the same results in unbounded systems.

On the other hand, closed systems typically rely on effective mechanisms involving centralized control, centralized data, or hierarchical structures both in the development and execution of the system in order to provide the required degree of trust. However, when users or automated components are incompletely known, are untrustworthy, can be compromised, or can make errors, these mechanisms can amplify problems and undermine success. In particular, centralized control cannot be employed effectively against participants and components that are unknown or against those for which there is no effective enforcement mechanism.

Simply put, centralized data and control create a single-point target for attacks, accidents, and other failures. They also create communications vulnerabilities by increasing communication delay, transaction time, and ultimately user response times. Any hierarchical structure in a complex system has the unfortunate property that every node and link of the hierarchy constitutes a single point of failure for the system as a whole. That is, if the success of a function or system depends on the success of each of its components and subsystems, then an error, compromise, or failure in any one component propagates to the system as a whole and undermines system-wide success.

Interoperability in Unbounded Systems

Problems that arise from the unbounded characteristics of systems of systems are normally manifest as interoperability problems. We have categorized interoperability problems as programmatic, constructive, or operational depending on whether they arise from diffusion of management responsibility, diverse technical approaches, or user interactions with the system, respectively.

Traditional approaches to interoperability have focused on strengthening coordination among the organizations involved by tightening centralized control, increasing visibility and transparency of components, imposing more and stronger standards, and imposing additional coordination mechanisms. Obvious as this approach may be to those familiar with closed fully-understood systems, in systems of systems they become less and less effective as their size, distributiveness, and unboundedness increase. The continuing advance of memory, processor, and communications technologies ensures ever-increasing demands for systems and systems of systems that are more complex and more geographically distributed with more poorly understood and unknown components. Even if complete and accurate information could be obtained, it would be outdated rapidly by continuously changing circumstances within a system of systems. Instead, effective solutions must be developed that recognize, act upon, and exploit the inherent characteristics of unbounded systems.

Often when problems of interoperability arise in complex systems, there is a tendency to try to gain greater visibility, to extend central control, and to impose stronger standards. Not only are these actions ineffective in complex systems, they also increase the likelihood of certain kinds of accidents, user errors, and other failures. What are called normal accidents are inherent and occur naturally in complex

systems [1]. The frequency of normal accidents increases with the degree of coupling in systems. Coupling is increased by central control, overly restrictive specifications, and broadly imposed interface standards. Developers of systems of systems should strive for loose coupling.

Interoperability and Emergent Algorithms

Emergent properties are those properties of a whole that are different from, and not predictable from, the cumulative properties of the entities that make up the whole. The concept of emergent properties becomes increasingly important as the number and type of “actors” in a system of systems increase. Thus, large-scale networks such as the Internet (and in the future, networks that support net-centric warfare) are likely to experience emergent properties. Such networks are composed of large numbers of widely varied components (hosts, routers, links, users, etc.) that interact in complex ways.

Of necessity, each participant in such real-world systems (both the actor in the network and the engineer who constructed it) acts primarily in his or her own best interest. As a result, perceptions of system-wide requirements are interpreted and implemented differently by various participants, and local needs often conflict with overall system goals. Although collective behavior is governed by control structures (e.g., in the case of the networks, network protocols), central control can never be fully effective in managing complex, large-scale, distributed, or networked systems.

The net effect is that the global properties, capabilities, and services of the system as a whole emerge from the cumulative effects of the actions and interactions of the individual participants propagated throughout the system. The resulting collective behavior of the complex network shows emergent properties that arise out of the interactions among the participants.

The effect of emergent properties can be profound. In the best cases, the properties can provide unanticipated benefits to users. In the worst cases, emergent properties can detract from overall capability. In all cases, emergent properties make predictions about behavior such as reliability, performance, and security suspect. This is potentially the greatest risk to wide-scale networked systems-of-systems. The SEI recognizes that any long-term solution must involve better understanding and managing of emergent properties.

Recent research in the area of emergent algorithms [2, 3] has begun to identify, develop, and refine the methods first developed for other sorts of systems to solve problems of constructive interoperability. These methods and techniques are derived by analogy from approaches that have been effective in social, biological, and economic systems, but are applicable to the design, implementation, and evolution of software in a systems-of-systems context.

The methods of emergent algorithms as they apply to interoperability include cooperation without coordination, dynamic adaptation, continuous trust validation, dynamic capability assessment, opportunistic actions, anticipatory neighbor assistance, encouragement and influence, perturbation, and survivable architectures. At the same time, emergent approaches demonstrate an aversion to the risks imposed by tight

coupling of systems, dependency on centralized control and data, and the vulnerabilities of hierarchical structures.

At their most fundamental level, emergent algorithms exploit cascading effects of loosely coupled, dynamically changing, and partially trusted neighbors to achieve a common purpose shared by a subset of the participants. Only a limited repertoire of emergent methods has been identified, and they are only partially understood. The complete range of effects, whether positive or ill, resulting from cascading interactions with dynamically changing neighbors, is unknown. Phase shifts are a particularly difficult class of emergent effects that can occur in any physical system. They are poorly understood in most physical domains, but offer the potential for both dramatic benefits and catastrophic failures. Well-known examples of phase shifts include the transition of an airplane wing angle from one that provides lift to one initiating a stall, an overload in a power system that initiates a blackout, or the action of a fuse in breaking a circuit.

Conclusions

This article offers some starting points and future directions for interoperability. Systems of systems now being constructed are characterized by vast complexity, many unknown aspects, and limited control with multiple organizations involved in their management, implementation, and use. The environment is increasingly unbounded. In unbounded systems, traditional software engineering methods become more and more constrained in their ability to effectively achieve interoperability. Emergent algorithms, applying methods analogous to those used in natural systems, may offer viable alternatives to traditional software engineering approaches.

References

- [1] Perrow, Charles. *Normal Accidents – Living with High-Risk Technologies*. New York: Basic Books, 1984.
- [2] Fisher, D. A. and Lipson, H. F. “Emergent Algorithms -- A New Method for Enhancing Survivability in Unbounded Systems,” *Proceedings of 32nd Annual Hawaii International Conference on System Sciences (HICSS-32)*, Maui, HI, Jan. 5-8, 1999. Los Alamitos, CA: IEEE CS Press, 1999.
- [3] Fisher, David A. “An Emergent Approach to Interoperability in COTS-Based Systems,” submitted to *International Conference on COTS-Based Software Systems 2005 (ICCBSS-2005)*, Bilbao, Spain, Feb. 7-11, 2005.

About the Authors

David A. Fisher is a Senior Member of the Technical Staff at the Software Engineering Institute (SEI) at Carnegie Mellon University, Pittsburgh PA. He also holds faculty appointments in the H. John Heinz III School of Public Policy and Management and in the Department of Engineering and Public Policy (EPP) and supervises theses in the Information Network Institute (INI) program of the Electrical and Computer Engineering (ECE) Department, all at Carnegie Mellon University.

He currently conducts research in interoperability in systems-of-systems with emphasis on emergent approaches and survivable architectures. Earlier he led the design and implementation of the Easel modeling and simulation system, developed algorithms in the areas infrastructure assurance, cooperative unmanned autonomous vehicles, mobile ad hoc networks, propagation of epidemics, and assessment of communications protocols. Before coming to the SEI, he managed program in component-based software and learning technologies at the Advanced Technology Program of the U.S. Department of Commerce, developed a theory of property based types, lead research projects in design of compilers, operating systems, and instruction set architectures, was Vice President for Advanced Development at Western Digital Corporation, proposed and orchestrated the development of the Ada language as a joint effort of the US DoD and the EU, and served as Staff Specialist for C4 in the Office of the Secretary of Defense.

Fisher holds a Ph.D. in Computer Science from Carnegie Mellon University, an M.S.E. from Moore School of Electrical Engineering, University of Pennsylvania, and a B.S. in Mathematics from Carnegie Institute of Technology.

Dennis Smith is the leader of the SEI initiative on the integration of software-intensive Systems. This initiative focuses on interoperability and integration in large-scale systems and systems of systems. Earlier, he was the technical lead in the effort for migrating legacy systems to product lines. In this role he developed the method “Options Analysis for Reengineering” (OAR) to support reuse decision making. He has also been the project leader for the computer-aided software engineering (CASE) environments project. Smith is a co-author of the book, Principles of CASE Tool Integration. He has an M.A. and PhD from Princeton University, and a B.A from Columbia University.

The views expressed in this article are the author’s only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.