Common Software Platforms in System-of-Systems Architectures: The State of the Practice

John Klein[1], Sholom Cohen[1], Rick Kazman[1,2]
[1]Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA, USA
{jklein, sgc, kazman}@sei.cmu.edu
[2]University of Hawaii
Honolulu, HI, USA

Abstract - *System-of-systems (SoS) architectures based on common software platforms have been commercially successful. Common platforms are a goal of several DoD initiatives (US Army Common Operating Environment, US Navy Open Architecture, multi-service Future Avionics Capability Environment), but progress on creating and adopting such platforms has been slow. We conducted a study to understand the technical issues related to SoS common platform development and adoption, and the non-technical constraints that must be satisfied. We interviewed 12 experts, collecting and analyzing mostly qualitative data. Although there were significant differences in approaches between developers of commercial SoS platforms, military SoS platforms, and command and control SoS, all reported that non-technical constraints dominate intrinsic technical issues. We recommend further research is needed to create systematic architecture design and analysis methods for SoS, to study agile development methods in the SoS context, and to develop approaches to documentation for constituent systems within a SoS.*

Keywords: System of systems, platform, architecture, analysis, documentation, empirical research.

1 Introduction

1.1 System of Systems Context

A *system* is a collection of elements that together produce some result that cannot be obtained by the elements operating individually [7]. The elements of a system may themselves be large and complex, and comprised of sub-elements. The term *system of systems* (SoS) designates the case where the constituent elements of a system are collaborating systems that exhibit two properties: (1) operational independence (each constituent system operates to achieve a useful purpose independent of its' participation in the SoS); and (2) managerial independence (each constituent system is managed and evolved, at least in part, to achieve its own goals rather than the SoS goals) [12]. There is a consensus among researchers [5][14] and practitioners [16] that these properties necessitate developing and operating a SoS in different ways than a large, complex (single) system.

In the SoS context, interoperability among the constituent systems is a primary architecture concern [10]. A SoS platform that provides services and functions to all constituent systems within a SoS is one strategy to promote interoperability [9].

1.2 Platforms, Product Platforms, and System-of-Systems Platforms

The term "platform" is used to refer to several different concepts. In the military domain, a platform is a vehicle (ship, aircraft, tank, etc.) that transports systems and provides physical services such as power and cooling [6]. In other domains, "platform" is used to refer to the common elements reused across a product line or product family. Cusumano calls this a "product platform" [3], while Madni uses the term "platform-based engineering" [11]. In both cases, the primary concern of this type of platform is reuse of hardware, software, and related assets.

A third use of the term "platform" is what Cusumano refers to as an "industry platform" and we call a "system-of-systems platform" [9]. This type of platform provides *services* to an open set of systems that interact to form a SoS. The services provided can be general-purpose, such as directory and authentication services, or domain-specific, such as geospatial information processing for a command and control SoS. The primary concerns of a SoS platform are (1) support interoperation among the systems using the platform; (2) reduce the cost and time needed to develop or modify systems for use in the SoS; and (3) enable modular substitution of constituent systems in the SoS.

These three concerns are related to each other. First, a SoS platform supports interoperation by providing common information models (semantics) and common communication mechanisms such as provided by frameworks and middleware (syntax). The platform may also prescribe patterns or sequences of interaction for certain system-of-system functions.

Second, the SoS platform also provides implementations of services needed by constituent systems. As services are relocated within the SoS architecture from constituent systems into the SoS platform, system-to-system dependencies are replaced by system-to-platform dependencies, which typically reduces the time and effort required to develop, integrate, and test systems to create the

SoS. The availability of a SoS platform also reduces the barrier to entry for an organization to create a new or replacement system, since less effort and expertise are needed, and the risk is lower.

Finally, the ability to substitute one implementation of a system for a different implementation is necessary to create an *ecosystem*, where organizations "have a strategy to open their technology to complementors and create economic incentives (such as free or low licensing fees, or financial subsidies) for other firms to join the same 'ecosystem' and adopt the platform technology as their own" [3]. The reduced barrier to entry described above contributes to the incentives to join or participate in the ecosystem.

Examples of successful commercial ecosystem-enabling SoS platforms include Facebook, Apple (iOS, OS X, iCloud, App Store, etc.), and Salesforce.com. In other domains, Future Avionics Capability Environment (FACE) and the US Army Common Operating Environment (COE) are examples of emerging SoS platforms, with their eventual success yet to be determined.

1.3 Goals of this Study

As noted above, there are commercially successful SoS platforms, but success in military systems and other domains has been elusive. As part of a multi-year research project, we are developing systematic approaches to support SoS platform development. Our research began with this exploratory study of the state of the practice of SoS architecture development, with a focus on architectures that include SoS platforms. Our goals for this study included answering the following questions:

1. What processes are used to develop SoS architectures, and how are software elements of the architecture treated in the processes used?

2. What challenges do SoS programs face in developing architectures; performing test, integration, and assurance; managing runtime configuration and operation; and evolving the SoS? What approaches have been used in successful programs to overcome these challenges?

3. What are the constraints on new approaches to developing, using, and evolving these SoS architectures?

4. What are the important differences between practices used to create commercial SoS architectures and military SoS architectures?

The following section presents our research approach, including our interview protocol and participant demographics. We next present the results of our interviews, followed by analysis and discussion of the results. We conclude by identifying additional research needed to address the issues raised in this study.

2 Research Method

Our first attempt to answer the questions outlined above was to convene a workshop, bringing invited participants together to answer these questions in a group setting. Participants were recruited from the professional networks of the research team members. Our inclusion criterion for participation was direct experience as an architect or systems engineering leader on the development of at least one SoS. Each invitee was also requested to forward the invitation to other appropriately qualified members of his network.

Only one invitee agreed to participate in the workshop/focus group, and several responses implied a reluctance to share relevant experience in a group setting. This led us to develop an interview protocol that reported responses anonymously. The recruitment process was repeated, yielding 14 qualified participants. Two of these participants later withdrew from the study, leaving the 12 participant interviews that are reported here.

Study participants had between 10 and 25 years of professional experience. Two of the participants had experience working on one SoS project, and eight participants had experience working on four or more SoS projects. Table 1 describes the types of organizations represented by the participants.

Table 1 - Organization Types Represented

Organization Type	Number of Participants
Commercial software development (non-military)	4
Military system development – Industry	5
Military system development – Government	3

For each interview, one researcher acted as the lead interviewer and at least one other researcher participated. All interviewers recorded responses that were later combined into a single interview record. We prepared a script to guide the lead interviewer and act as a checklist to ensure that all topics were covered, as an interviewee's response to one question often covered several of our topics. Nine of the interviews were conducted with the lead interviewer meeting the interviewee in-person and other interviewers participating in the interview by telephone, and the other three interviews were conducted solely by telephone. All of the researchers have experience conducting interviews as part of exploratory research.

3 Interview Questions

Each interview began with the lead interviewer reading a prepared statement that stated that all reported results would be anonymized to protect the privacy of the interviewee and their organization, and that the interviewee should not disclose any protected proprietary information. We then collected the demographic information presented above.

Interview questions were divided into three sections, corresponding to the study goals outlined above. The complete interview instrument is available online at http://www.andrew.cmu.edu/user/jklein2/SoS-Study-Interview-Questions.pdf.

The first set of questions focused on the processes used to develop SoS architectures. The questions in this set allowed us to understand how the participant defined the term "system of systems", and their general approach to the architecture process. Maier identified conflict between the classical systems engineering "is part of" decomposition hierarchy and the layered software approach based on the "is used by" relation [13], and so we asked questions to understand how the participant addressed this conflict. Finally, constituent systems in a SoS are independently developed and evolved, and so we asked questions to understand how architecture trade offs are framed and how decisions are made, balancing the concerns of the SoS with the concerns of each constituent system. The objective of this set of questions was to understand the scope of activities and concerns of SoS architecture, and to understand how software concerns interact with other SoS concerns.

The second set focused on challenges in various system lifecycle phases, and on how successful projects addressed those challenges. We focused on lifecycle activities that are related to architecture: development of constituent systems for the SoS; test, integration, and assurance of the SoS; runtime configuration and management of the SoS; and sustainment and evolution of the SoS. For each of these activities, we asked participants to discuss technical and non-technical challenges, and to provide examples of projects that successfully addressed the challenges. The objective of this set of questions was to identify specific gaps in current practice where new methods would have the greatest impact, and to identify specific solutions employed by the interviewees that would be candidates for generalization.

The final set of questions focused on the constraints that a solution (e.g., a new method for design or analysis) must satisfy. We focused on the same four activity areas used in the previous question set. The objective of this set of questions was to identify factors necessary for any new approach to be successfully translated from research into practice.

4 Results and Discussion

Participant's responses broadly separated into three groups, based on the their group context, experience, and responsibilities. These groups are (1) commercial SoS platforms, (2) command and control SoS, and (3) military SoS platforms. In the discussion that follows, we organize our findings using these groups.

4.1 Architecture Framing and Processes

Participants framed and defined the SoS platform in markedly different ways. Command and control SoS architects and military SoS platform architects described the platform in terms of "what it is". They focused on technology characteristics, such as APIs and programming language bindings, and on the services provided by the platform. Commercial platform developers, on the other hand, framed the platform in terms of "what it does". They focused on the platform's ability to create network effects that support an ecosystem. Military SoS platform architects also recognized the importance of an ecosystem to the success of the platform, but they were less focused on the role of the platform in enabling the ecosystem.

None of the participants reported the use of particular methods or approaches for SoS architecture development or analysis or for SoS platform definition.

Most participants identified two development scenarios: Creation of a new SoS comprised primarily of new constituent systems and integration of existing systems to create a SoS. The first scenario applies primarily to directed systems of systems [12], where constituent systems goals and governance are aligned well with those of the SoS. In this scenario, architecture design can begin either top-down, based on requirements with a platform emerging as the design matures, or bottom-up, creating a platform first and then defining systems that use the platform. In the second scenario, architecture is much more constrained, and consistency or conceptual integrity across the SoS may not be achievable without substantial rework (and hence cost).

When making architecture decisions, no participant reported performing economic modeling of design alternatives. For commercial platform developers, time-to-market was a primary decision driver and after a viable solution was identified, they did little additional solution space exploration. On the other hand, architects involved with military systems reported that extensive trade studies were performed, with architecture decisions frequently driven by development constraints. Software was not an early concern for them – it was initially treated like any other element of the SoS architecture, but as the architecture design matured, concerns such as maximizing software development efficiency, minimizing development cost, and meeting development schedules were high priorities that were balanced against overall SoS measures of performance.

Command and control SoS architects reported that downstream lifecycle costs and sustainment costs were less important than SoS operational performance. In contrast, both commercial and military platform architects reported that success depended on proper consideration of future needs. In the commercial organizations, these future needs were defined through market analysis, and in development of military platforms, future needs were informed by science and technology investment roadmaps.

All participants noted that deep domain knowledge was necessary to design a successful architecture. Domain knowledge enabled architects to identify the most important tradeoffs, eliminate ineffective parts of the solution space, and make timely decisions.

The command and control SoS architects reported that the way software architecture concerns are framed has changed over time. Earlier projects framed decisions only in terms of functional requirements, while more recent projects are framing decisions in terms of both functional and quality attribute requirements. Commercial system-of-system platform architects framed decisions in a context that included both functional and quality attribute requirements, and they did not explicitly distinguish between the two types of requirements.

4.2 Challenges and Patterns of Success

All participants reported that the primary challenges in developing and evolving SoS architectures are not rooted in technology, but are due to non-technical factors. These non-technical factors include misalignment of development organization and authority with the architecture, misalignment of system and SoS goals, reluctance to introduce dependence on the SoS platform into the constituent system architectures, and regulatory and policy constraints (for systems acquired by the United States government) that diminish the potential value of a SoS platform approach.

Another challenge reported across all projects is a challenge in migrating existing constituent systems in the SoS to use the platform: New platform features frequently duplicated existing features in the constituent systems. Modifying a constituent system to use the platform version of a feature incurs a short-term cost, but produces long-term value from reduced integration and sustainment costs. Commercial platforms use the value of modification as an incentive, whereas military organizations relied on top-down mandates.

These reported challenges are similar to challenges of developing, adopting, and sustaining software product lines [15]. Experience from software product lines and platform-based engineering provide insight into some of the challenges of platform-based systems of systems. Practices that are successful for single products need to change to achieve success in a product line context. Similarly, practices focused on developing single systems must change to be successful in the context of a platform-based SoS. Practices used for software product lines consider the relationships among development, organizational, and management concerns, and recognize that architecture and technology is just one contributor to overall product line success.

Software product line and platform-based engineering practices also promote the reuse of assets other than software, such as tools, plans, templates, test equipment, test cases, and personnel training and skills. Architects of military SoS platforms included assets such as documentation, training materials, and user community collaboration repositories as part of their SoS platform.

When developing or evolving systems to use the SoS platform, many participants reported challenges related to documentation of the constituent systems within the SoS. Although extensive architecture and design documentation may exist for a constituent system, it is often focused on the independent operation of the constituent system, and does not adequately address concerns related to the constituent system's operation in the SoS. Examples included resource scheduling approaches and handling of interface errors or exceptions.

The large scale and complexity of the SoS architecture context created several challenges in creating the initial instantiation of the SoS platform. Architects of the command and control SoS used the "V-model" [4] to develop their SoS. Significantly, the relatively long time between architecture definition and system integration allowed some architecture errors to remain undiscovered until late in the development process. On one project, they addressed this challenge by shifting to an iterative agile approach during later phases of the development cycle. This enabled faster feedback on the correctness of design decisions, but there were unresolved questions that remained: Is it practical to use an iterative approach from the beginning of the project, or is there an initial base of functionality that should be in place before starting an iterative approach? What is the best way to plan iteration contents and duration?

The commercial SoS platform architects reported several approaches to creating and delivering the initial instantiation of the platform. From these, we have identified two "proto patterns" [17]. The first proto pattern is a sequence for evolving the architecture of a new platform. This began by first defining and implementing atomic message types and message schemas, with no concept of workflow (i.e. sequences of messages related to a business task or process). Initially, all workflow is organically built into the systems and applications using the platform. Later, workflow orchestration was added to the platform, with the platform providing versioned workflow definitions that include endpoint roles (endpoint cardinality, supported message sets, and other workflows that the endpoint can participate in), workflow sequence definitions, and transaction support. This proto pattern allowed an initial version of the platform to be deployed quickly, and allowed incremental definition of workflows based on actual platform use.

A second proto pattern is related to the evolution proto pattern described above. Workflow execution scalability and availability is achieved by maintaining workflow state only in the participating endpoints, not in the platform infrastructure. This "stateless platform" approach is a refinement of stateless services in service-oriented architectures.

Maintaining backward compatibility for systems using the SoS platform was reported as a challenge in architecture evolution. The commercial platform architects addressed this challenge through extensive test automation. Nearly all testing was automated, with one organization reporting that they have "tens of thousands" of automated tests, which allow them to maintain full compatibility back to systems developed for the first versions of the platform (the platform is now almost 10 years old and is updated three times per year). Commercial SoS platform architects also reported a proto pattern for deploying new platform features. This three-step pattern begins by piloting a new feature with selected customers, and special IT operations processes are used to carefully monitor usage and quality attributes such as performance. In a second release, the feature is stabilized with those customers, and IT operations processes are similar to standard production processes. Finally, in a third release, the feature is generally available to all customers in production. (In the organization using this proto pattern, the time from pilot to general availability of a feature was 4-8 months.)

Command and control SoS architects reported similar issues maintaining compatibility as the architecture underwent evolution throughout the initial SoS development iterations. They also used a test automation strategy. This strategy used tests that covered both syntax and semantics of interfaces, and incorporated modeling and simulation systems into the test environment to extend test coverage beyond just platform interfaces.

4.3 Solution constraints

Participants identified two general constraints that must be satisfied by any new approaches or methods to address the challenges discussed in the previous section.

The first constraint is that any new approach or method should be integrated with existing tools, including tools used for architecture modeling, analysis, and documentation, and tools used for project/program management. The need for integration with architecture tools was expected – adoption of new approaches and methods is facilitated if there is no need for acquiring or learning new tools. The need for integration with project/program management tools is indicative of the strategic importance of architecture decisions, and the necessity of efficiently translating decisions about technical approaches into cost, schedule, and other metrics relevant to program executives.

The second constraint was that any new approach or method must align with assurance and certification processes. Much of the potential value of a SoS platform is reducing the cost and time to perform assurance and certification of the SoS, but this value can be accrued only if the features included in the platform, the analysis of the platform architecture, and documentation provided for the platform are aligned with the assurance and certification requirements of the SoS.

Finally, several participants involved in military SoS discussed specific constraints that their environment imposes on creating an ecosystem based on a SoS platform. These participants expect that such an ecosystem could reduce SoS acquisition costs, since the modular substitution of SoS elements should promote competition among suppliers of the elements. The ecosystem is also expected to increase innovation by enabling a broader community of contributors to new and improved SoS capabilities. Creation of such a SoS platform-based ecosystem in this environment is currently constrained by US government acquisition policies and by a limited ability to create effective incentives for both acquirers and suppliers to join or participate in the ecosystem.

5 Conclusions

This study interviewed 12 experts to characterize the state of the state of the practice of system-of-system architecture development, with a focus on architectures that include SoS platforms. Our goal was to inform further research in systematic approaches to support the development and evolution of architectures for SoS platforms. This study identified several areas where additional research is needed.

The first area is selection of features for a SoS platform. The study identified a number of critical stakeholder concerns, including time-to-market, ease of adoption, support for future capabilities, and alignment with SoS assurance and certification processes. Feature selection requires consideration of both the problem space, to identify candidate platform features and assess their value, and the solution space, to assess costs to implement and maintain each feature. Systematic approaches to analyze the problem space might combine techniques such as mission thread analysis [8] with domain analysis [15]. Solution space analysis might include economic models and models that consider alignment of the architecture with constraints such as acquisition strategy, organizational structures, and other socio-technical factors. These analyses would be facilitated by catalogs of architecture knowledge, such as pattern handbooks, to provide a repertoire of solutions that exhibit particular functional and quality attribute properties. Finally, a systematic approach, such as economic modeling, is needed to prioritize and select features for inclusion in the platform from a set of candidates.

The second area for additional research is in agile development methods for platform-based systems of systems. Approaches for architecture-led incremental development have primarily focused on the software and system level [1][2]. Further work is needed to model the more complicated dependencies in a SoS architecture, and to develop iteration planning strategies that accommodate the managerial independence of the constituent systems.

A final area for additional research is to create approaches to characterize and document constituent systems to support their use in systems of systems. Systematic approaches are needed to identify the relevant concerns, and collect and present the information to efficiently satisfy those concerns. An approach such as the creation of an ISO 42010-style architecture description viewpoint may be appropriate.

6 Acknowledgements

Copyright 2013 Carnegie Mellon University and IEEE. This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

This material has been approved for public release and unlimited distribution. DM-0000234.

7 References

[1] F. Bachmann, R. L. Nord, and I. Ozkaya, “Architectural Tactics to Support Rapid and Agile Stability,” *CrossTalk,* May/June 2012.

[2] S. Bellomo, R. Nord, and I. Ozkaya, “A Study of Enabling Factors for Rapid Fielding (Combined Practices to Balance Speed and Stability)”, in *Proc. of the International Conference on Software Engineering (ICSE)*, 2013.

[3] M. Cusumano, “The Evolution of Platform Thinking,” *Communications of the ACM,* vol. 53, no. 1, pp. 32-34, January 2010.

[4] K. Forsberg and H. Mooz, “The Relationship of System Engineering to the Project Cycle,” in *Proc. First Annual Symposium of National Council on System Engineering*, 1991, pp. 57-65.

[5] A. Gorod, B. Sauser, and J. Boardman, “System-of-Systems Engineering Management: A Review of Modern History and a Path Forward,” *IEEE Systems Journal,* vol. 2, no. 4, pp. 484 -499, December 2008.

[6] J. W. Greenert, “Payloads over Platforms: Charting a New Course,” *US Naval Institute Proceedings Magazine,* vol. 138, no. 7, July 2012.

[7] INCOSE. *A Consensus of the INCOSE Fellows* [Online]. http://www.incose.org/practice/fellowsconsensus.aspx (Accessed 18 Aug 2012).

[8] R. Kazman, M. Gagliardi, and W. Wood, “Scaling up software architecture analysis,” *Journal of Systems and Software,* vol. 85, no. 7, pp. 1511-1519, July 2012.

[9] J. Klein, G. Chastek, S. Cohen, et al., “An Early Look at Defining Variability Requirements for System of Systems Platforms,” in *Proc. Workshop on Requirements Engineering for Systems and Systems-of-Systems (RESS)*, 2012.

[10] J. Klein and H. van Vliet, “A systematic review of system-of-systems architecture research,” *Submitted for publication,* 2013.

[11] A. M. Madni, “Adaptable platform-based engineering: Key enablers and outlook for the future,” *Systems Engineering,* vol. 15, no. 1, pp. 95--107, 2012.

[12] M. W. Maier, “Architecting principles for systems-of-systems,” *Systems Engineering,* vol. 1, no. 4, pp. 267-284, 1998.

[13] M. W. Maier, “System and software architecture reconciliation,” *Systems Engineering,* vol. 9, no. 2, pp. 146--159, 2006.

[14] M. W. Maier, “Research Challenges for Systems-of-Systems,” in *Proc. IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2005, pp. 3149 - 3154. doi: 10.1109/ICSMC.2005.1571630

[15] L. M. Northrop and P. C. Clements, “A Framework for Software Product Line Practice, Version 5.0.” Software Engineering Institute, Online 2012, http://www.sei.cmu.edu/productlines/frame_report/index.html (accessed 28 Feb 2013).

[16] ODUSD(A&T), “Systems Engineering Guide for Systems of Systems, Version 1.0.” ODUSD(A&T), 2008.

[17] *Proto Pattern.* [Online] http://c2.com/cgi/wiki?ProtoPattern. 17 Feb 2009. (accessed 18 Dec 2012).