Bridging Requirements and Architecture for Systems of Systems

Charles B. Haley
Asia Pacific University College of Technology and Innovation
Technology Park Malaysia
Kuala Lumpur, Malaysia
charles [at] ucti.edu.my

Bashar Nuseibeh
Centre for Research in Computing
The Open University
Walton Hall
Milton Keynes, UK
b.nuseibeh [at] open.ac.uk

Abstract

A system of systems (SoS) is formed from existing independent component systems. Some reasons these independent systems might be combined include a merger or acquisition, a temporary partnership, because of the formation of an integrated supply chain, or if a service-oriented architecture is used. SoSs are difficult to analyze because of the scale of the integration, the components' independent existence, and the (potentially) conflicting nature of their requirements. We propose bridging between requirements analysis and architecture of an SoS by using an interdisciplinary approach. From Software Engineering we take iterating between requirements and architecture, and from Philosophy we take structured argumentation. Iterating between requirements and architecture is ideal for exposing issues with constructing a system of systems from existing artifacts. Structured argumentation is used to explore these issues, to inform the analysis, to reveal underlying assumptions in the analysis, and to help either establish system correctness to an acceptable level or provide rebuttals that invalidate the analysis.

1. Introduction

There is a growing recognition that today's complex information systems are not single entities, but instead are independent parts that function together. The independent parts are themselves complex systems, often having a lifespan and purpose of their own divorced from whatever role they play in the larger context. Easterbrook observes that the collection of parts is composed into "a complex system-of-systems that includes a broad technological infrastructure along with a wide set of human activities" [6]. Research in Systems Engineering concurs with his observation: systems of systems (SoS) are "large-scale concurrent and distributed systems the components of which are complex systems themselves" [16], and require "[an expansion to systems engineering] to consider the full range of systems engineering services increasingly needed in a modern organization [...]" [4].

Software-based information systems are usually built from existing components. Some components are small, such as networking services, and others are large, such as a commercial DBMS. The components provide services that help satisfy requirements. Note, however, that the requirements for a component must be met before it can provide its services, and the requirements and architecture of the software system must take the component's requirements and capabilities into account. In other words, the requirements/composition process is iterative. System requirements affect which components are chosen, the components' capabilities affect what the system can do, and the requirements of the components affect how the system is architected and, potentially, its requirements.

Although building an SoS is conceptually similar to building with components, the scale is different. The systems (the components) that comprise an SoS have a life and purpose independent of the SoS, with their own requirements, architecture, and lifecycle. Integrating these systems into an SoS is composition on an ultra-large scale. All the complexities of composition, such as inconsistencies, incompatibilities, and uncertainty, are also on an ultra-large scale [17]. As a result, determining unambiguous and consistent requirements for an SoS is challenging, and may not even be possible. For example, requirements for the component systems might conflict with requirements for the SoS, something easy to imagine for information security. What is possible to do in a reasonable timeframe or budget might conflict with what is required, usually resulting in changed requirements. Requirements might even be unknowable, perhaps because of stakeholders' rapidly changing needs [17].

Service-oriented applications raise similar issues. According to The Open Group, a service in a service-oriented application "is a logical representation of a repeatable business activity", and "is self-contained" [11].

Ghezzi suggests that applications built using service-oriented principles are intended to support "dynamic, goal-oriented opportunistic federations of organizations" that use "services that should be composable" [7]. Papazoglou et al define services as "autonomous, platform-independent entities that can be described, published, discovered, and loosely coupled in novel ways" [19]. In other words, a service-oriented application is an SoS where the systems are the services. We can expect them to exhibit the same difficulties as other SoSs.

These difficulties do not relieve us of the need to analyze the requirements of an SoS. What we must find is 'how'. This paper presents a proposal for using an interdisciplinary approach to analyze information system requirements for systems of systems, bridging requirements engineering and architecture. From requirements engineering we take:

- the Twin Peaks model [18] for iterating between requirements and architecture, helping us understand the impact of system requirements on system architecture and vice versa.
- the *i**/Tropos requirements engineering methodologies [3, 23], to help us understand the interplay between the components from an agent, action, intention point of view, and
- a variant of Jackson's problem diagrams [14] to represent the SOS's combined context, to help us understand how the systems are interconnected and what is interchanged between them.

From philosophy we take structured argumentation, in our case Toulmin's model [20], to help ensure that at each kind of abstraction and at each iteration, the analyst understands the ramifications of the choices and assumptions made during requirements analysis.

Our challenge is to combine these methods into a whole that is practical to use and provides useful insights to the requirements analyst and to the stakeholders. We also want to assist in establishing 'due diligence'-style arguments that help with resource and time allocation, and possibly whether and how to go forward with construction of the system.

It is worth noting that although we claim that our proposed combination of tools and methods is both novel and useful, we do not claim that our use of an inter-disciplinary approach is novel. Argumentation has been used for many years in artificial intelligence, in particular in the legal domain; for some examples, see [1, 2, 8]. One can find several formal computing models of arguments (e.g., [5, 9, 12, 15]). The emerging discipline of Service Science, Management, and Engineering (SSME) holds that SSME is inherently interdisciplinary, covering "all types of value-creating resources, and the disciplines or competencies that study and apply them" [13]. Our own work in security requirements [10] proposes methods similar to what we propose here.

The remainder of this paper is structured as follows. Section 2 presents the notations we propose to use for describing systems. Section 3 describes how we want to use structured argumentation. Section 4 presents an illustrated example, and Section 5 concludes.

2. Requirements and Architecture

We established in the introduction that requirements for an SoS can affect the component systems, and that the requirements for the component systems can affect the SoS. Because of these cross-ways effects, we propose using the Twin Peaks model to iterate between systems requirements and systems architecture. As seen in Figure 1, Twin Peaks informs the development of an architecture by supplying the requirements as they are determined. Equally, the evolving architecture informs the requirements process when the facts on the ground argue for or against satisfaction of some requirements. We recognize that this process intermixes, and perhaps confounds, design and requirements. Our position is that producing the 'right' requirements is a laudable goal, but ultimately not useful if the requirements cannot be satisfied given the constraints or within acceptable cost and time limits.

Our proposal calls for requirements modeling to determine, at some level of abstraction, what the SoS is intended to do and what the component systems already do. Large-scale systems will require use of a higher abstraction level, at least initially. For this analysis we chose *i**/Tropos to model the systems from an agent + intention point of view. We chose *i** because it shows delegation between agents and responsibilities of agents at varying levels of detail. The agents may be computers, humans, or organizations. We postulate that by modeling at the level of agents, we can put aside the problem of system and organizational boundaries, permitting us to

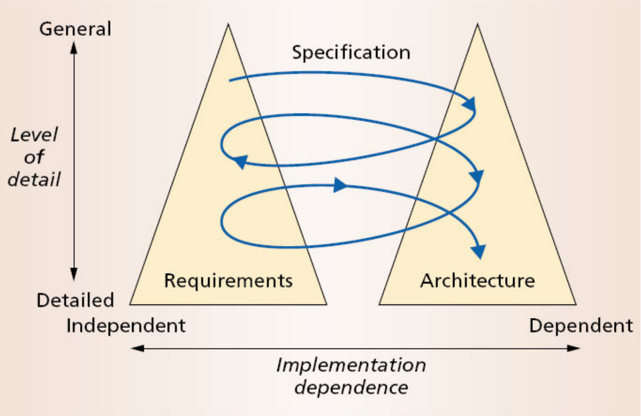

Figure 1 – Twin Peaks (from [18])

reason about the existing systems and the desired system in terms of responsibility and intention. Conflicts discovered at this level are resolved by changing the requirements, changing the existing systems, or by adding agents that act as 'impedance matchers' between the systems.

An *i**/Tropos model describes neither the physical components in a system nor the information that components share on the connections. As such, one is not able to detect problems that arise because of the topology or capability of components. For this reason our proposal calls for describing the system architecture in real-world terms using Jackson's problem diagrams [14]. Problem diagrams describe a system in terms of physical domains and the connections between them. Domains can be anything that exists in the world, including people, computers, other machines, or even naturally-occurring items like rivers. Problem diagrams are well suited to describe large-scale systems, because the domains can be at any appropriate level of abstraction, as can the description of what information is shared between domains. Finally, domains can be combined and details can be elided, both of which can help when analyzing large-scale systems.

Under our proposal, the analyst can begin in any way that makes sense to the analyst, including: 1) describe the existing systems' behavior with *i**, 2) describe the existing systems' architecture with problem diagrams, 3) describe the future system's behavior, or 4) describe the future system's architecture. The analysis makes Twin Peaks iterations between the descriptions (existing or future). When describing the future system, requirements may be at odds with reality, either conflicting with requirements of the existing system or by demanding an architecture that does not exist.

During analysis, decisions about the goals of agents will frequently be made with incomplete knowledge. The decisions will be based upon assumptions, which might be about which capabilities do exist, which can exist, what stakeholders want and do not want, the regulatory context, information and physical security, and budget and time. Although assumptions are related to goals, they are not the same. A goal is an objective, and is neither true nor false. An assumption is an assertion of truth, and its veracity can be challenged. Goals are explicit, but assumptions may be implicit. Structured argumentation, described in Section 3, is used to expose and test the veracity of assumptions and, should the assumptions not be correct, to explore the consequences.

3. Structured Argumentation

As noted above, assumptions play a significant role in the analysis. To increase confidence and to help avoid costly mistakes, the correctness of assumptions should be tested. An assumption might itself not stand up to scrutiny, or it might depend on deeper unstated assumptions that are not correct. For example, when considering information flow between the systems of two companies, an analyst might assume that user security mechanisms will be compatible because the same operating system is used in each company. One unstated assumption is that both companies are using the native security features of the operating system. Probing further might expose that one company is using a third-party biometric security product, invalidating the unstated assumption. Early exposure and testing of the underlying assumptions will help avoid costly mistakes.

We propose using Toulmin's structured argumentation [20] to test the correctness of assumptions and to expose deeper underlying assumptions. Toulmin's argument structure, shown in Figure 2, shows what the parts of an argument are and how the parts fit together. The arrows in the figure show 'movement' of the argument from grounds (left) to claims (right). Intersections show where the parts of the argument connect to support or detract from the argument.

Toulmin et al. [21] describe arguments as consisting of:

1. *Claims*, providing the end point of the argument – what one wants to establish is true.
2. *Grounds*, supplying support for the argument, e.g., evidence, facts, common knowledge, etc.
3. *Warrants*, connecting and establishing relevancy between the grounds and the claim. A warrant explains how the grounds are related to the claim, not the validity of the grounds themselves.
4. *Backing*, showing that the warrants are themselves trustworthy. These are in effect grounds for believing the warrants.
5. *Modal qualifiers*, establishing within the context of the argument the reliability or strength of the connections between the components. Modal qualifiers permit the introduction of rebuttals.
6. *Rebuttals*, describing conditions that might

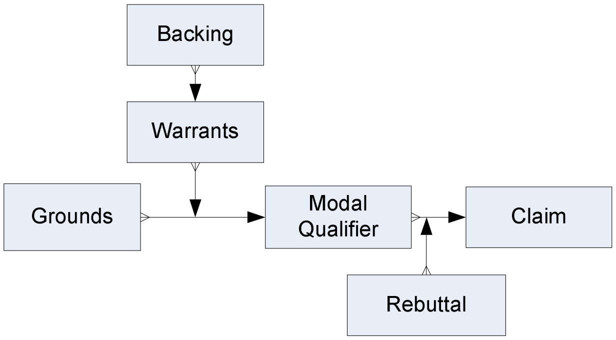

Figure 2 – Toulmin's argument structure

invalidate any of the grounds, warrants, or backing, thereby reducing the support for the claim.

Arguments are summarized by Toulmin et al. [21] as follows: “The *claims* involved in real-life arguments are, accordingly, *well founded* only if sufficient *grounds* of an appropriate and relevant kind can be offered in their support. These grounds must be connected to the claims by reliable, applicable, *warrants*, which are capable in turn of being justified by appeal to sufficient *backing* of the relevant kind. And the entire structure of argument put together out of these elements must be capable of being recognized as having this or that kind and degree of certainty or probability as being dependent for its reliability on the absence of certain particular extra-ordinary, exceptional, or otherwise *rebutting* circumstances.”

In our proposal, an initial assumption is a claim; something that is assumed to be true. Using the argument structure, we ask “why is this claim true?” The answer will produce evidence, or grounds, for the argument. Asking “what relationship does this evidence have with the claim” will generate warrants: assumptions (grounds) that connect the evidence to the claim. Asking “what could invalidate the argument” will produce rebuttals.

Considering our earlier security example, the claim ‘security mechanisms are compatible’ is supported by grounds ‘the same operating system is used’. However, asking about the relevance of grounds will expose the (unstated but assumed) warrant ‘using the same operating system means the companies are using the same security mechanisms.’ This warrant is clearly false, as there is no guaranteed causal relationship between use of an operating system and use of a security mechanism. The compatibility claim is not supported by the evidence.

4. An Illustrated Example

This section presents a simplified example illustrating our proposed approach. We recognize that the scenario is overly simple, that the example does not show whether our ideas will scale up, and that the issues raised are rather obvious. This is intentional; our goal at this point is to present our concepts while minimizing complexity in the scenario.

Industrial Weights & Measures Ltd (IWM) has developed a product they believe will sweep the world, a customizable kitchen weighing scale. Called the OffScale, it can be delivered in any combination of 6,000 colors, with or without decoration composed of precious metal or stones. The scale speaks six languages or dialects; the customer chooses which six from 218 available. It provides nutrition information based on what is being weighed; the nutrition tables are specific to the customer’s region. In addition, the scale can be equipped with a wireless networking facility appropriate for the customer’s region, permitting the scale to retrieve nutrition information for food products not in its database.

IWM is capable of manufacturing the scales, but it does not have an appropriate sales channel for such a luxury item, support capacity in the target regions, or the capability to provide the wireless networking. Both to solve these problems and to give the desired ‘exclusive’ impression, IWM forms a virtual enterprise, OffScale Ltd. The partners are For You Only (FYO), a company specializing in sale of exclusive impulse-purchase products; Global Support Inc (GSI), a provider of service and technical support; and Worldwide Roaming (WR), a provider of mobile international roaming data networks.

FYO insists that because an OffScale will frequently be purchased on impulse, the sales person must be able to produce price quotes while the person is in the store. Unfortunately, the complexity of customization is such that sales people must use a customizer provided by IWM. The customizer will provide a list of legal options at every point, along with the price and delivery time. If the wireless option is chosen, then WR must be consulted to find what kind of network hardware is required and to price the connection. Differing warranty service levels are available, so GSI must price based on the service level and the region. A purchased configuration is passed to IWM to be built, to WR to provision the network, to GSI to arrange the support, and to FYO to invoice the customer and prepare the selected delivery method.

All of the companies agree to present an OffScale Ltd face to the world. However, to preserve their individual competitive advantages, each company wishes to keep its operational and cost structures private. Each organization has its own information system. These systems will be used to support OffScale, but will also be used to support other parts of their businesses. The companies believe that their existing applications will be sufficient to support OffScale.

In this paper, we will limit our requirements examination to the sales process. Figure 3 shows the *i** intentional model for this process. The circles are actors in the system, and the ovals are goals that one actor delegates to another. We see that a prospect delegates the goal Quote Scale to the sales person; the buyer does not care how the goal is achieved. Similarly, the salesperson delegates the goal Configure Product to an agent at IWM, and so on. The diagram tells us that for a sales person to make a quote while face-to-face with a prospect, all four partners are involved and must provide the needed information (satisfy their delegated goals) in real time. As such, the companies must be networked together and all the quoting processes automated.

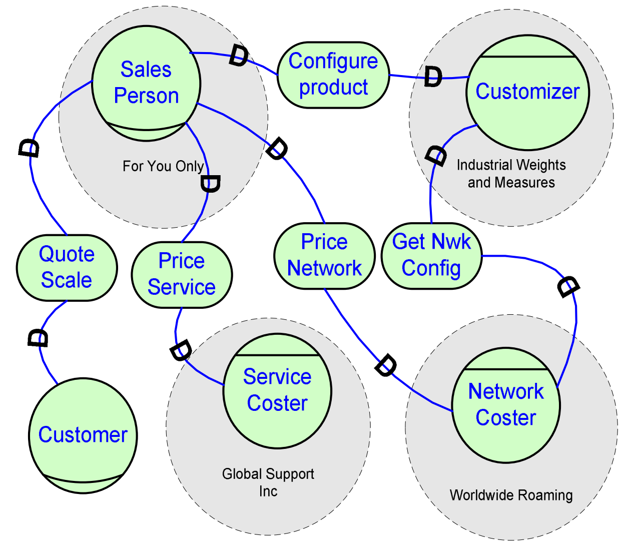

Figure 3 – *i diagram of OffScale's sales process**

We next produce the context diagram for the system, using a variant of Jackson's problem diagrams. Figure 4 shows the proposed context diagram for the SoS, based on the stated premise that the existing applications will support the sales processes. The solid boxes represent physical items (domains in Jackson's terms), in this case computer systems running an application; the application is noted in the box. The lines between boxes indicate that information (phenomena in Jackson's terms) is shared between the computers, in our case by a network of some kind. The dashed boxes represent corporate boundaries, and are shown for information only. The domains in the diagram are summaries; they do not show the details of the computing systems. In Jackson's terms, the domains are *projections*, showing the system at a reduced level of detail to facilitate the analysis.

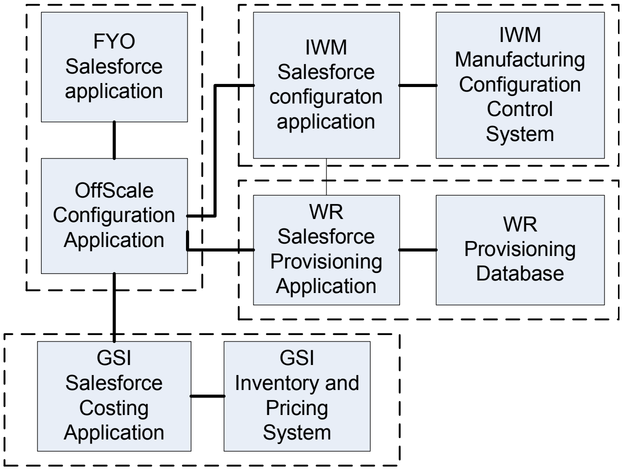
FYO Salesforce application
OffScale Configuration Application
IWM Salesforce configuraton application
IWM Manufacturing Configuration Control System
WR Salesforce Provisioning Application
WR Provisioning Database
GSI Salesforce Costing Application
GSI Inventory and Pricing System

Figure 4 – First context diagram

Our next step is to construct an argument diagram for the requirement that accurate price quotes are available in real-time (seconds). Figure 5 shows an initial argument for the requirement, constructed from information provided to the analyst by the system architects, and derived from the *i** diagram. The diagram says that assuming that the two grounds (all pricing steps are automated, and systems are networked) are true, and assuming that the warrants (networks permit application integration and automation permits rapid response) are true, then the claim quotes will be available quickly is also true. However, after further reflection the analyst thinks of three rebuttals: circumstances that if true invalidate the assumptions and therefore the argument. If any of these rebuttals can be shown to be true, the argument is false.

The first rebuttal raises the possibility that either a company's systems are not networked at all, or that they are networked in a way that does not permit application integration. For example, if one of the companies does not have an extranet facility (a way of permitting outside access to some parts of the internal network) of some kind or another, then the network is in effect not available. The second rebuttal challenges the warrant automation permits rapid response by asking if the existing applications in fact contain the application programming interfaces (APIs) required to support OffScale. The third rebuttal challenges the assertion that all the functions necessary to quote a price are already automated.

The analyst would take these rebuttals back to the system architects and stakeholders for further consideration. In our scenario, this questioning surfaces the following facts:

- When precious stone decorations are to be quoted, to avoid what was thought to be large inventory costs the IWM customizer requires that stones be priced manually, which introduces significant delay. FYI asserts that delaying a quotation is unacceptable. To resolve this problem, either the requirement must

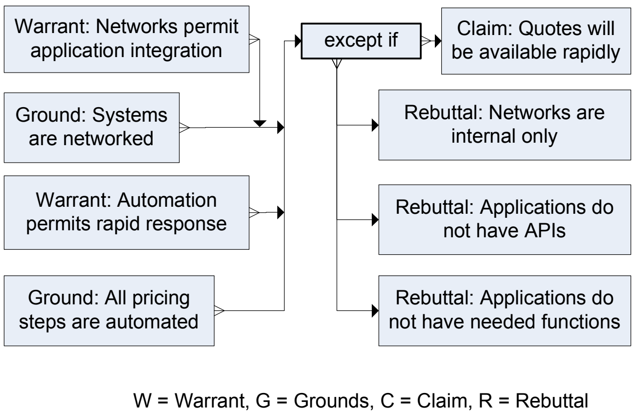
Warrant: Networks permit application integration
except if
Claim: Quotes will be available rapidly
Ground: Systems are networked
Rebuttal: Networks are internal only
Warrant: Automation permits rapid response
Rebuttal: Applications do not have APIs
Ground: All pricing steps are automated
Rebuttal: Applications do not have needed functions
W = Warrant, G = Grounds, C = Claim, R = Rebuttal

Figure 5 – Argument that prices are available

change or the precious stones configuration option must be removed or automated somehow.

- GSI has never needed an extranet, because all of its employees operate in GSI offices. The company has no facility in place for allowing controlled external access into its corporate intranet. An extranet gateway must be added to their network.

WR's provisioning application was not designed to separate business services from the user interface. Several the business services, for example generating the files that control the network switches, are implemented in the GUI part of the client application, making them inaccessible to the FYO configurator. A new application must be built that separates out the necessary business services, in order to support access by FYO's application.

The process of exposing these 'facts on the ground' is in fact an iteration between requirements and architecture. By asking the right questions, one discovers the reality of the situation: that the existing applications cannot support the desired application. Given this reality, one must either change the requirements expressed in the intentional model or change the system architecture. In our case, the partners make the latter choice, and Figure 6 presents the result. All of the partners have agreed to supply a gateway application that implements an agreed-upon API. In GSI's case, the gateway application will serve as the extranet gateway. In WR's case, the application will provide the business services implemented in the GUI. In IWM's case, the application will keep track of a small inventory of precious stones, permitting rapid pricing of a fixed number of configurations.

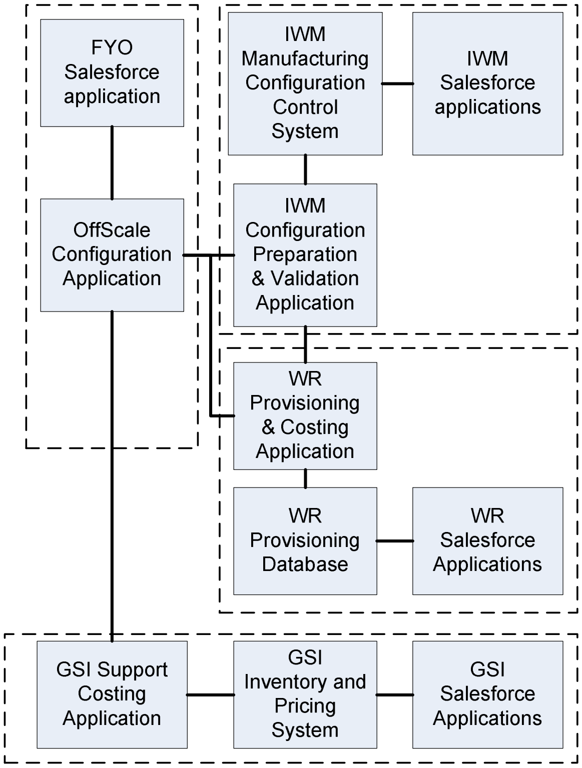
FYO Salesforce application
OffScale Configuration Application
IWM Manufacturing Configuration Control System
IWM Salesforce applications
IWM Configuration Preparation & Validation Application
WR Provisioning & Costing Application
WR Provisioning Database
WR Salesforce Applications
GSI Support Costing Application
GSI Inventory and Pricing System
GSI Salesforce Applications

Figure 6 – Enhanced context diagram

At this point the analyst should redo the arguments, documenting the assumptions behind the new architecture and challenging them where appropriate.

5. Discussion and Future Work

Our position is that analyzing requirements of systems of systems using our proposed combination of tools should provide the analyst with high-quality information early, potentially avoiding costly mistakes by exposing issues earlier in the life cycle. In addition, our method should provide input to due-diligence arguments, helping with the larger decisions (to proceed or not), with the smaller decisions (how much to invest), and should the need arise, with providing a trace of why decisions were made and the information available.

One question that arises is "where do the rebuttals come from?" We contend that many, if not most, of the rebuttals will come from asking three questions about each of the grounds and warrants in an argument: "why is this true", "what could make it not true, and "what happens if it is not true?" Our contention needs to be tested in a real-world project.

Our next step is to further test our proposal by extending our example, adding complexity and further iterations. In addition, we will test our position that these techniques will scale up, helping with requirements analysis of larger systems of systems. After this further exploration, or instead of it if an appropriate project is available, we wish to test our proposal in a real-world setting in a running project. An after-the-fact case study approach could also be used, as long as we have access to the people who were involved with the project.

We conjecture that our approach will provide benefit when analyzing requirements for monitoring and managing SoSs. Wang et al provide a formal framework for system monitoring, where the monitoring requirements come from *i** goal models [22]. However, some issues arise when considering using their framework on an SoS:

- Visibility: information required by the monitor might be in a part of the SoS not visible to the monitor, or may be confidential information that the system owner chooses not to share,
- Scale: it is almost certainly impractical to instrument everything in an SoS, and therefore choices must be made about what to monitor,
- Change: the individual systems will likely not be stable, complicating the analysis of monitored data.
- Consistency: conditions considered anomalous by one member of the system may be considered correct by another.

We postulate that our proposal helps overcome these issues. The intentional model for the SoS will provide many clues about what is important in the SoS and therefore *what is critical* to monitor. The context diagram will help decisions about *where to place* the monitor's probes to overcome visibility and stability problems. The arguments will help determine *correctness*: whether the proposed partial monitoring is sufficient. Our conjecture needs further exploration and validation.

In addition to validating the approach itself, we want to investigate whether existing tool support is adequate, and if not, what support is needed.

6. References

[1] T.J.M. Bench-Capon and G. Staniford, "PLAID: Proactive Legal Assistance," *Proceedings of the 5th International Conference on Artificial Intelligence and Law*, College Park, MD, USA: ACM Press, 1995, pp. 81 - 88.

[2] T. Bench-Capon and H. Prakken, "Argumentation," *Information Technology & Lawyers: Advanced technology in the legal domain, from challenges to daily routine*, A.R. Lodder and A. Oskamp, eds., Springer, 2005, pp. 69-90.

[3] P. Bresciani, A. Perini, et al., "Tropos: An Agent-Oriented Software Development Methodology," *Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, May. 2004, pp. 203-236.

[4] P.G. Carlock and R.E. Fenton, "System of Systems (SoS) enterprise systems engineering for information-intensive organizations," *Systems Engineering*, vol. 4, no. 4, Oct. 2001, pp. 242-261.

[5] C.I. Chesñevar, A.G. Maguitman, and R.P. Loui, "Logical models of argument," *ACM Computing Surveys*, vol. 32, no. 4, Dec. 2000, pp. 337-383.

[6] S. Easterbrook, "Scale Changes Everything: Understanding the Requirements for Systems of Systems," Keynote presented at 6th IEEE International Conference on COTS-based Software Systems (ICCBSS'07), Banff, Alberta, Canada, 1 Mar 2007.

[7] C. Ghezzi, "Service-Oriented Computing: Where Does It Come From? A Software Engineering Perspective," Keynote presented at 3rd International Conference on Service-Oriented Computing, Amsterdam, The Netherlands, Dec 2005.

[8] T.F. Gordon, "The Pleadings Game: Formalizing Procedural Justice," *Proceedings of the 4th International Conference on Artificial Intelligence and Law*, Amsterdam, The Netherlands: ACM Press, 1993, pp. 10-19.

[9] R. Haenni, B. Anrig, J. Kohlas, and N. Lehmann, "A Survey on Probabilistic Argumentation," *Proceedings of the Adventures in Argumentation Workshop, held at the Sixth European Conferences on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'01)*, A. Hunter and S. Parsons, eds., Toulouse, France: 2001, pp. 19-25.

[10] C.B. Haley, R. Laney, J.D. Moffett, and B. Nuseibeh, "Security Requirements Engineering: A Framework for Representation and Analysis," *Transactions on Software Engineering (IEEE)*, vol. 34, no. 1, Jan. 2008, pp. 133-153.

[11] D. Hornford, "Definition of Service-Oriented Architecture," The Open Group, Jun 2006; http://opengroup.org/projects/soa/doc.tpl?gdid=10632 (accessed 18 Apr 2008).

[12] A. Hunter, "Making Argumentation More Believable," *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, San Jose, CA, USA: The AAAI Press, 2004, pp. 269-274.

[13] IfM and IBM, *Succeeding through Service Innovation: A Discussion Paper*, University of Cambridge Institute for Manufacturing, Cambridge, UK, 14 Jul 2007.

[14] M. Jackson, *Problem Frames*, Addison Wesley, 2001.

[15] A. Jøsang, "Artificial Reasoning with Subjective Logic," *Second Australian Workshop on Commonsense Reasoning*, Perth, Australia: 1997.

[16] V. Kotov, *Systems of Systems as Communicating Structures*, Technical Report HPL-97-124, HP Labs, Oct 1997.

[17] L. Northrop, P. Feiler, et al., *Ultra-Large-Scale Systems*, Software Engineering Institute, Pittsburgh, PA, USA, Jun 2006.

[18] B. Nuseibeh, "Weaving Together Requirements and Architectures," *IEEE Computer*, vol. 34, no. 3, Mar. 2001, pp. 115-117.

[19] M.P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing: State of the Art and Research Challenges," *Computer*, vol. 40, no. 11, 2007, pp. 38-45, doi:10.1109/MC.2007.400.

[20] S.E. Toulmin, *The Uses of Argument*, Cambridge, UK: Cambridge University Press, 1958.

[21] S.E. Toulmin, R.D. Rieke, and A. Janik, *An Introduction to Reasoning*, New York, NY, USA: Macmillan, 1979.

[22] Y. Wang, S.A. McIlraith, Y. Yu, and J. Mylopoulos, "An automated approach to monitoring and diagnosing requirements," *Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE'07)*, Atlanta, Georgia, USA: ACM, 5 Nov 2007, pp. 293-302, doi:10.1145/1321631.1321675.

[23] E. Yu, "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering," *Proceedings of the Third IEEE International Symposium on Requirements Engineering (RE'97)*, Annapolis, MD, USA: 6 Jan 1997, pp. 226-235.