Towards Modelling and Analysing Non-Functional Properties of Systems of Systems

Vanea Chiprianov, Katrina Falkner, Laurent Gallon, Manuel Munier

Towards Modelling and Analysing Non-functional Properties of Systems of Systems

Vanea Chiprianov
LIUPPA, Univ Pau & Pays Adour
Mont de Marsan, France
vanea.chiprianov@univ-pau.fr

Katrina Falkner
School of Computer Science
University of Adelaide
Adelaide, SA, Australia
katrina.falkner@adelaide.edu.au

Laurent Gallon, Manuel Munier
LIUPPA, Univ Pau & Pays Adour
Mont de Marsan, France
name.surname@univ-pau.fr

Abstract – *Systems of systems (SoS) are large-scale systems composed of complex systems with difficult to predict emergent properties. One of the most significant challenges in the engineering of such systems is how to predict their Non-functional Properties (NFP) such as performance and security, and more specifically, how to model NFP when the overall system functionality is not available. In this paper, we identify, describe and analyse challenges to modelling and analysing the performance and security NFP of SoS. We define an architectural framework to SoS NFP prediction based on the modelling of system interactions and their impacts. We adopt an Event Driven Architecture to support this modelling, as it allows for more realistic and flexible NFP simulation, which enables more accurate NFP prediction. A framework integrating the analysis of several NFP allows for exploring the impacts of changes made to accommodate issues on one NFP on other NFPs.*

Keywords: System of system, non-functional property, performance, security, model, analysis, prediction, architecture, framework.

1 Introduction

Systems of systems (SoS) are large-scale concurrent and distributed systems that are comprised of complex systems [19]. Several definitions of SoS have been proposed, some of them historically reviewed in e.g. [14]. SoS are complex systems themselves, and are distributed and characterised by interdependence, independence, cooperation, competition, and adaptation [8].

Characteristics that have been proposed to distinguish between complex but monolithic systems and SoS are [27]: operational independence of the elements, managerial independence of the elements, evolutionary development, emergent behaviour, geographic distribution. Other sets of characteristics of SoS, partially overlapping, have been identified, e.g. [4]: autonomy, belonging, connectivity, diversity, emergence.

Investigating Non-functional Properties (NFP), like performance or security, of SoS comprise all the issues associated with the investigation of NFP of composing systems. Moreover, there are challenges related to the specific nature of SoS [19], [8] [6], [24]. The decentralised, distributed nature of SoS require an emphasis on interface architecting to foster collaborative functions among its composing independent systems. The interaction between systems creates new unexpected behaviours. Hidden failures cascade, becoming more damaging. The loose connectivity between composing systems runs contrary to ensuring a high performance of the overall SoS. All these factors increase the difficulty of analysing NFP of SoS.

One approach for the early checking of meeting performance requirements is performance prediction modelling. In this paper, we adopt the definition introduced by [2]: 'the process of predicting (at early phases of the life cycle) and evaluating (at the end) based on performance models, whether the software system satisfies the user performance goals'. Similarly, we require that our understanding of performance prediction be based upon an existing performance model. Prediction of software performance has developed from early approaches based on abstract models to Model-Driven Engineering (MDE) [3] based approaches. MDE is typically applied to the development of software, but may also be used in the configuration and deployment phases, system execution emulation and analysis. One of the main techniques in MDE is the use of Domain Specific Modelling Languages (DSMLs). A DSML is defined in this paper to be a language that offers expressive power focused on a particular problem domain through appropriate notation and abstractions. This provides the specification, construction and documentation of artefacts of a software-intensive system.

System Execution Modelling (SEM) [17], a recent development from research into measurement-based performance prediction, provides detailed early insight into the performance of a system design. A SEM-based approach supports the evaluation of overall (software) system performance, incorporating component interactions. These approaches are based upon simple models of resource consumption from the component's "business logic" [17], [34], [15], and support detailed performance modelling of software systems, thus enabling predictions of performance through execution of representative source code of behaviour and workload models deployed upon realistic hardware testbeds. SEM and MDE may be used in combination to support the

emulation of system components and performance models, enabling performance data to be used to redesign and reconfigure the system, prior to any construction of the corresponding real system.

Identifying challenges to security engineering within SoS is the first step in engineering security within SoS. As highlighted by [28], a very desirable research direction would be an integrated description and analysis method that can express and guarantee user level security, reliability, and timeliness properties of systems built by integrating large application layer parts - SoS. Moreover, systems engineering must incorporate consideration of threats and vulnerabilities into the requirements, architecture, and design processes; the importance and the challenges of applying system security engineering beyond individual systems to SoS has been recognised [9].

SoS analysis can drive changes in the composing systems to exploit opportunities or correct problems that were not originally anticipated. A key architectural tool may be the use of predictive modelling and simulation to compare architectural alternatives [21]. Approaches to modelling and analysing system security also comprise quantitative evaluation and prediction, using attack models, e.g. CWE [7] and CVSS [13], and security models, e.g. Secure i* [10] and SecureTropos [32], or UMLSec [20] and CARiSMA [39], in addition to system models.

How could these approaches for modelling and analysing performance and security of systems be extended in the context of SoS? How could they be combined in a coherent architectural framework that supports modelling and analysing of both performance and security issues of SoS? Such a framework would open the way to an integrated solution for predicting other NFP of SoS, like safety or reliability. An integrated solution would allow exploring architecture alternatives by analysing not only one NFP in isolation, but also the impacts of changes made to satisfy one NFP on other NFP.

In what follows, we identify, describe and analyse challenges to modelling and analysing performance and security of SoS, in Section 2. From these challenges, we deduce requirements an architectural framework for predicting performance and security of SoS should fulfil, in Section 3. We then define an architectural framework, in Section 4. We conclude and indicate future work.

2 Challenges to Modelling and Analysing NFP of SoS

NFP engineering for SoS needs to take into account the characteristics specific to SoS, and how they impact NFP of SoS. At a general, abstract level, these impacts on performance and security of NFP include [37], [14]:

Operational Independence: The component systems of an SoS may be operated separately, under different policies, using different implementations. This can lead to potential incompatibilities and conflicts between the security or the performance of each system, including different requirements, protocols, procedures, technologies and culture. Additionally, some systems may be more vulnerable to attacks or have less critical real-time constraints than others, and compromise or non-fulfilment of time constraints of such systems may lead to compromise or non-fulfilment of constraints of the SoS.

Managerial Independence: Component systems may be managed by completely different organisations, each with their own agenda. Activities of one system may produce difficulties for the security or performance of another system. What rights should one system have to specify the security or performance of another system for SoS activities and independent activities? How can systems protect themselves within the SoS from other component systems and from SoS emerging activities?

Evolutionary Development: An SoS typically evolves over time. Therefore, the security mitigations and performance optimisations in place for an evolving SoS will be difficult to completely specify at design time, and will need to evolve as the SoS evolves.

Emergent Behaviour: SoS are typically characterised by emerging or non-localised behaviours and functions that occur after the SoS has been deployed. These could clearly introduce security and performance issues for the SoS or for its component systems. Who should respond and where are responses needed?

Geographic Distribution: An SoS is often geographically dispersed, which may cause difficulties in trying to secure the SoS as a whole if national regulations differ or in ensuring performance requirements are met due to numerous and long paths of interaction.

NFP are often taken into account only at the end of the development life-cycle of a system. Consequently, any a posteriori modification is expensive. That is why, they must be taken into account as soon as possible and designed-in rather than relying on hardening of systems post implementation [23], [26]. But how can security and performance be integrated into the SoS architecture [5]? How to represent the SoS in a form that enables detailed analysis, especially when full details of the component systems may not be readily available? A key architectural tool in this respect may be the use of predictive modelling and simulation to compare architectural alternatives. In any process for improving a SoS, alternative architectures would need to be carefully considered and modelled to ensure the SoS is not compromised or undesirable emergent behaviours result. Moreover, the SoS may not be responsive to a single analysis. Should, therefore, SoS analysis be incremental and the SoS should be available for testing almost on a continuous basis [21]?

Interdependency analysis is concerned with examining the possible cumulative effects of a single security or

performance incident on multiple systems. Such cascading failures or bottlenecks could result in a complete blackout. Hence, it is important to identify them before they happen [29]. How to identify threats that may appear insignificant when examining only first-order dependencies between composing systems of a SoS, but may have potentially significant impact if one assesses multi-order dependencies [25]? How can such an approach be integrated in a SoS risk assessment methodology [12]?

Designing SoS needs to be addressed different from the traditional process of stove-piped systems [11]. Which would be the best suited process for architecting SoS? Should it be iterative, agile, or model-based, etc? How does the type of dependencies (strong, loose) between the development of SoS and that of its constituent systems influence the design process of the SoS?

Security of SoS that have strong real-time constraints needs that devices properly mutually authenticate themselves to prevent insertion of malicious devices or messages in case of attacks like man-in-the middle. The main challenge in the design and implementation is to retain the temporal properties of a real-time system. Any additional and unpredictable delay is critical for the communication and consequently for the access control and traffic separation based on the time-triggered protocol [36]. Of course, the authentication case can be generalised to other security mechanisms that may introduce delays in time-constrained SoS. Such cases show how assuring one NFP may affect negatively another NFP. How to analyse the interplay between several NFP of SoS?

Authorisation is concerned with the management and control of the authorisation schemes used and the ability to grant SoS authentication to interested parties [22]. In a SoS, users with different backgrounds and requirements should be granted accesses to different resources of each composing system [40]. How would delegation of rights be handled? Who would be responsible for it?

Accounting/Auditing is necessary for the record of events and operations, and the saving of log information about them, for SoS and fault analysis, for responsibility delegation and transfer, and even digital forensics. However, there is no well-defined best-practice guideline on accounting agreed upon and adopted [40]. Where will this information be stored and who will be responsible for the generation and maintenance of logs [22]?

NFP metrics specific for SoS constitute another challenge. What could be security- and performance-specific metrics and aggregation functions for an SoS [9] [38] [18]?

What kind of data should meta-data contain? What kind of meta-data should be legally permitted to collect and employ? Should meta-data tags include data classification to provide controlled access, ensure security, and protect privacy? Should meta-data be crypto-bound to the original data to ensure source and authenticity of contents [11]?

3 Requirements for an Architectural Framework

From the challenges identified before, we deduce a series of requirements which an architectural framework for modelling and analysing NFP of SoS should fulfil:

1. *Loose coupling*: The systems that form an SoS are independent, but also need to interoperate and interact. Towards this, a mechanism that allows the description of loosely coupled interactions is needed.

2. *Interoperability of composing systems*: Different formalisms or modelling languages may be used to model NFP prediction across the SoS. A means to interconnect all these heterogeneous models is necessary.

3. *Interaction specification*: The mechanism allowing the specification of loose interactions between composing systems of the SoS needs to be precise enough so as to limit the emergence of unexpected interactions to the point that they can be analysed as part of the NFP prediction process. This also implies that the mechanism should allow for repeatability.

4. *Time and data distribution*: Because of its distributed nature, the NFP prediction model of an SoS needs time and data distribution mechanisms between its composing NFP models.

5. *Adaptability:* Both at the composing system and the SoS level, architectural reconfigurations within the NFP models are necessary to analyse different architectural alternatives. Thus, a mechanism to generate code for a specific architectural configuration is necessary.

6. *Sustainable evolution*: The NFP model of an SoS needs to accommodate for models of composing systems being added, removed, and changed. Therefore it needs to enable open-ended extension.

7. *User interaction:* The interaction specification may be done in a repeatable, manner, or in an interactive manner, in which the user manipulates the models at runtime.

8. *Logging mechanism*: To record events, operations and NFP metrics, a logging mechanism is essential.

9. *Authorisation specification*: A proper authorisation mechanism is necessary for the composing systems to cooperate together and allow users with different backgrounds and requirements to access different resources of each composing system.

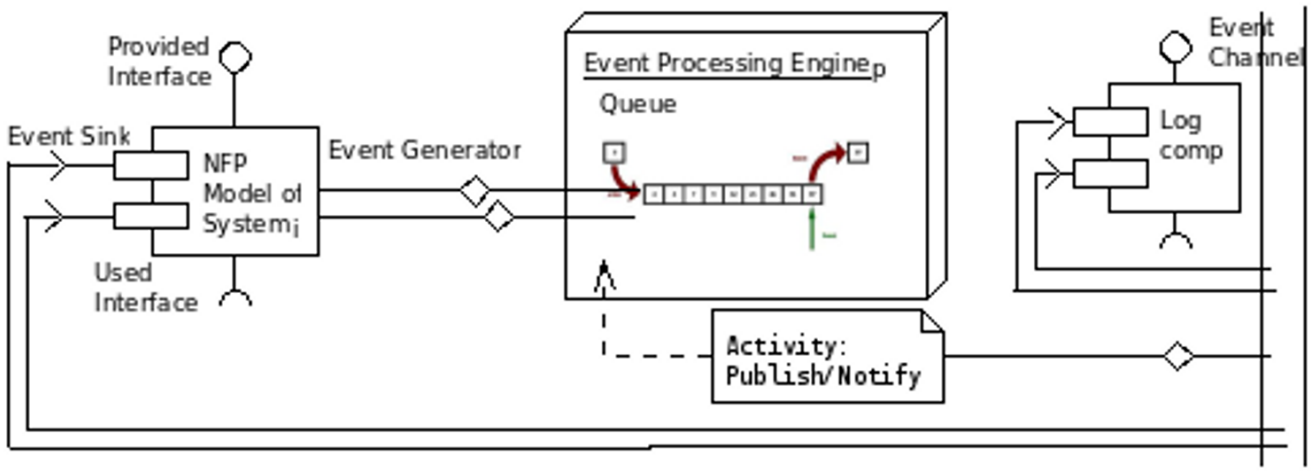

Illustration 1: Architectural Framework for NFP Prediction of Systems of Systems

4 Architectural Framework for Modelling and Analysing NFP of SoS

To address the requirements, we propose an architectural framework for predicting in an integrated manner both security and performance of SoS. This addresses the challenge of analysing the interplay between several NFP.

The architectural framework addressing the requirements defined above is presented in Figure 1, described in a formalism inspired from UML component diagrams. To predict the NFP of a SoS, the NFP of each of the composing systems needs to be predicted. Therefore, for each composing system, an NFP model is necessary. For each of these composing NFP models a SEM approach can be used. In addition to the composing NFP models, a mechanism to specify the interaction of the loosely coupled interoperable composing systems is necessary.

The NFP Models of a System (NFPMS) are integrated in our architecture based on the Event Driven Architecture (EDA) [30]. In an EDA, an event is immediately disseminated to all interested parties. The interested parties evaluate the event, and optionally take action. The creator of the event (event generator) has no knowledge of the event's subsequent processing, or of the interested parties (event sink). This makes EDA an extremely loosely coupled and highly distributed architecture. After an event has been triggered, a notification is produced and propagated to an event processing engine. The engine may order events according to priority criteria, or may do other processing specified in the activity associated with the event. Next, it publishes the notification on the event channel, which propagates it to all interested parties. The event sinks detect it and decide whether to consume it.

In addition to addressing the loose coupling requirement, EDA also addresses the sustainable evolution requirement. Since the EDA event generator knows nothing about the event sinks, this enables an open-ended extension approach, in which event generators do not need to be modified to include new event sinks. The EDA event channel can be enhanced with a time and data distribution management bus. Such a bus, as long as it is independent of technologies used to describe NFP models, and is distributed, enables the interoperability of the NFPMS.

The NFPMS may be thought of as a component that provides an interface, and may use other interfaces, cf. Figure 1. It is described using a specific formalism. Independent of this formalism, we model the interactions with NFPMSs of other systems using event generators and sinks. The event generators of an NFPMS produce notifications that are sent to an Event Processing Engine, which orders them in a queue using priority criteria. The engine processes the first event, and executes its associated activity. It publishes the event notification on the Event Channel., which may use different patterns, such as Reactor or Proactor [35], and propagates the event notification to all interested parties. The event sinks of all other NFPMS detect it and may decide on an action. Event generators and sinks are introduced in the NFPMS in ways specific to their NFP modelling formalism.

The events generated by all NFPMS are stored by the logging component. These events, together with associated meta-data are used to compute measurements of performance and security of component systems and of the SoS. These measurements can be aggregated into more complex ones, serving as the basis for the analysis and the prediction of SoS NFP. Therefore, the logging component is essential for collecting all these measures.

We advance a Scenario DSML. It is built on top of the SoS NFP prediction architecture framework, containing concepts specific to EDA, and thus generic with respect to the composing NFPMS. It is used to describe interactions between NFPMS. It is presented in more detail in Section 4.1. The Scenario DSML uses MDE and code generation techniques to facilitate adaptability. For example, from the scenario model, code can be generated to different middleware implementations of the event channel or bus, which addresses the adaptability requirement.

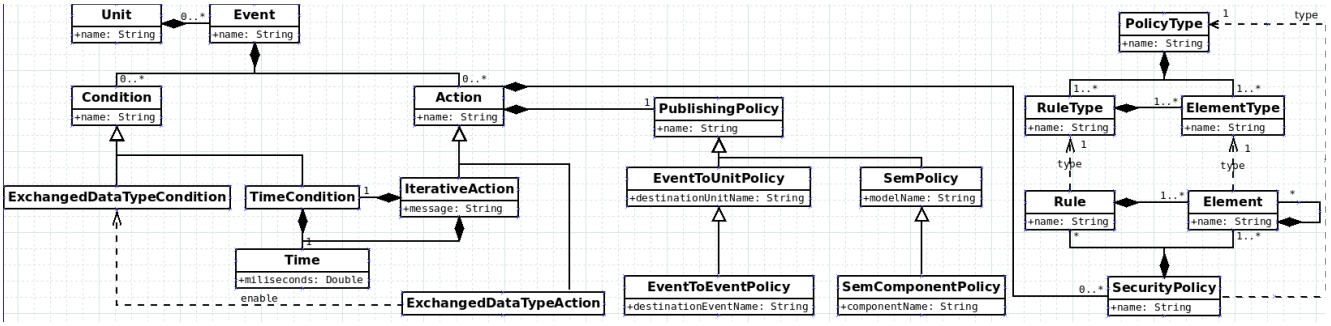

Illustration 2: Meta-model of the Scenario DSML

4.1 Scenario DSML

The Scenario DSML meta-model is presented in Figure 2. Most of its concepts are generic, related to EDA, and independent of the domain of the NFPMS. We introduce three groups of concepts, related to modelling events, publishing policies and security policies.

Modelling events are detailed in the following. The *Event* concept has several *Conditions* that have to be met in order for the event to be triggered. Conditions can be of different types, either referring to the data that is exchanged between events - *ExchangedDataTypeCondition*, or to the time when an event is triggered - *TimeCondition*. Once the event has been triggered, *Actions* can be performed. These can be one time actions, or *IterativeActions*, in which case a *frequency* defining the iteration and an *offCondition* need to be indicated.

Publishing policy entities are needed to indicate how the messages are routed between Units, (*EventToUnitPolicy*, *EventToEventPolicy*), or between Units inside the SEM of a composing unit and other SEMs (*SemPolicy*) or SEM components (*SemComponentPolicy*).

Security policies form the basis of several security formalisms like OrBAC [1]. The entities we introduce in the Scenario DSML are based on the meta-model introduced by [31]. An *Action* may have several *SecurityPolicies*, which contain several *Rules* and *Parameters*. A *SecurityPolicy* is of a certain type, *PolicyType*, and the same is true for *Rules*, which have a *RuleType*, and *Elements*, which have an *ElementType*. This meta-model allows defining security policies. This answers the authorisation specification requirement we have identified in Section 3.

From the Scenario DSML meta-model, using MDE meta-tools like Eclipse Modeling Framework (EMF) and Graphical Modeling Framework (GMF), a graphical editor can be generated, with which a Scenario model can be described. From such a Scenario model, code, e.g. in C++, can be generated, with a model-to-text transformation written in e.g. Eclipse Xpand. The MDE meta-tools allow the rapid generation of tools associated with the Scenario DSML. This facilitates the evolution of the Scenario DSML, changes in its meta-model being rapidly and semi-automatically propagated in its associated tools. Model-to-text transformations allow generating code for several middleware platforms. This enables a dynamic architecture at the SoS level, ensuring the adaptability of our architectural framework for NFP prediction of SoS.

4.2 Architecture Framework for NFP Modelling and Analysing of Standalone Systems

For completeness, we include here a discussion on NFP analysis and prediction process for a standalone system. For modelling the composing systems, DSMLs like PICML [16] - for modelling the architecture; for modelling behaviour - CBML [15] can be used. For modelling performance workload, DSMLs like WML [15] or MARTE [33] could be used. To model system security, DSMLs and tools like Secure i* [10] and SecureTropos [32], or UMLSec [20] and CARiSMA [39] could be used. For attack models, dictionaries and tools like CWE [7] and CVSS [13] could be used.

The SEM obtained after the modelling phase, including the architectural structure and behaviour, weaved with the NFPMS, is deployed on a hardware testbed to be simulated and analysed. The SEM models how the system is predicted to execute in a generic situation. Scenarios are used to analyse the system performance and security under various constraints, describing data flow into the system from external sources. The SEM, together with scenarios and information to the platforms on which the SEM will be deployed can be combined in various possible experiments. An experimental plan chooses from the different available alternatives for deploying the SEM on available hardware. Executing the SEM code within its indicated deployment produces execution traces and basic metrics about the system NFP.

5 Conclusions and Future Work

In this paper, we have identified, described and analysed challenges to modelling and analysing Non-functional Properties (NFP) of Systems of Systems (SoS), focusing on security and performance. We proposed an integrated architectural framework to iteratively analyse alternatives of the SoS architecture by executing predictive NFP System Execution Models (SEM). To describe these NFP models, we proposed the use of Model Driven

Engineering DSML families from which middleware-specific code can be generated. To describe the interactions between the NFP SEM, we proposed an Event Driven Architecture and a Scenario DSML. The process used in this framework is an iteratively, model-based one. While this architectural framework answers most challenges and requirements, there are still some challenges that have not been answered. A mechanism for interdependency analysis of multi-order dependencies needs to be integrated. What type of meta-data can and is legally permitted to collect is still an open question.

References

[1] A. Abou El Kalam, R. El Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel and G. Trouessin. "Organization Based Access Control", *IEEE 4th Intl Wksh on Policies for Distributed Systems and Networks*, Lake Come, Italy, 2003.

[2] S. Balsamo, A. D. Marco, P. Inverardi, and M. Simeoni, "Model-based performance prediction in software development: a survey," IEEE Transactions on Software Engineering, vol. 30, pp. 295–310, 2004.

[3] S. Beydeda, M. Book, and V. Gruhn, Eds., Model Driven Software Development. Spinger-Verlag, 2010.

[4] J. Boardman and B. Sauser. System of systems - the meaning of OF. In System of Systems Engineering, 2006 IEEE/SMC International Conference on, pages 6 pp.–, April 2006.

[5] D.J. Bodeau. "System-of-systems security engineering". In *10th Annual Computer Security Applications Conf*., pages 228–235, 1994.

[6] 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*, ser. LNCS, C. Choppy and O. Sokolsky, Eds. 2010, vol. 6028, pp. 59–82.

[7] *CWE - Common Weakness Enumeration*, 2014, 17 March, http://cwe.mitre.org.

[8] C. H. Dagli and N. Kilicay-Ergin, System of Systems Architecting. John Wiley & Sons, Inc., 2008, pp. 77–100.

[9] J. Dahmann, G. Rebovich, M. McEvilley, and G. Turner. Security engineering in a system of systems environment. In Systems Conference (SysCon), 2013 IEEE Intl, pages 364–369, 2013.

[10] G. Elahi, E. Yu: A goal oriented approach for modeling and analyzing security trade-offs. University of Toronto, Department of Computer Science. Technical report, 2007.

[11] D.L. Farroha and B.S. Farroha. Agile development for system of systems: Cyber security integration into information repositories architecture. In IEEE Systems Conf, pages 182 –188, 2011.

[12] I.N. Fovino and M. Masera. Emergent disservices in interdependent systems and system-of-systems. In IEEE Intl Conf on Systems, Man and Cybernetics, volume 1, pages 590–595, 2006.

[13] L. Gallon and JJ. Bascou - *Using CVSS in attack graphs* -The Sixth Intl Conf on Availability, Reliability and Security, 2011 – Vienna, Austria.

[14] A. Gorod, R. Gove, B. Sauser, and J. Boardman. System of systems management: A network management approach. In System of Systems Engineering, 2007. SoSE '07. IEEE International Conference on, pages 1–5, April 2007.

[15] J. Hill, D. Schmidt, J. Edmondson, and A. Gokhale, "Tools for continuously evaluating distributed system qualities," IEEE Software, vol. 27, no. 4, pp. 65–71, 2010.

[16] J. Hill, D. Schmidt, A. Porter, and J. Slaby, "CiCUTS: Combining System Execution Modeling Tools with Continuous Integration Environments," in Eng. of Computer Based Systems, 2008, pp. 66 –75.

[17] J. Hill, D. Schmidt, and J. Slaby, Designing Software-Intensive Systems: Methods and Principles, IGI Global, 2008, ch. System Execution Modeling Tools for Evaluating the Quality of Service of Enterprise Distributed Real-time and Embedded Systems, pp. 335–371.

[18] Jackson, D.; Sedrick, G.; Tayeb, K., "Algorithmic development of effectiveness prediction for system of systems," *System Theory, 41st Southeastern Symposium on* , vol., no., pp.164,168, 2009.

[19] M. Jamshidi, "System of systems engineering - new challenges for the 21st century," *Aerospace and Electronic Systems Magazine, IEEE*, Vol. 23, No. 5, pp. 4–19, 2008.

[20] J. Jurjens, "UMLsec: extending UML for secure systems development". *The Unified Modeling Language, Model Engineering, Languages Concepts and Tools*, 5th Intl Conf, 2002.

[21] Roy S. Kalawsky. The Next Generation of Grand Challenges for Systems Engineering Research. Procedia Computer Science, 16(0):834 – 843, 2013. Conf. on Systems Engineering Research.

[22] Michael Kennedy, David Llewellyn-Jones, Qi Shi, and Madjid Merabti. System-of-systems security: A survey. In 11th Annual Conf on Convergence of Telecommunications, Networking & Broadcasting, 2010.

[23] R. Koelle and M. Hawley. "Sesar security 2020: How to embed and assure security in system-of-systems engineering?" In *Integrated Communications, Navigation and Surveillance Conf*,, pages 1–11, 2012.

[24] H. Kopetz, "System of systems challenges," in *Proc. of the 29th Intl conf. on Computer safety, reliability, and security*, 2010, pp. 480–480.

[25] Panayiotis Kotzanikolaou, Marianthi Theoharidou, and Dimitris Gritzalis. Interdependencies between critical infrastructures: Analyzing the risk of cascading effects. In Sandro Bologna, Bernhard Hmmerli, Dimitris Gritzalis, and Stephen Wolthusen, editors, Critical Information Infrastructure Security, volume 6983 of LNCS, pages 104–115, 2013.

[26] S.J. Lukasik. Vulnerabilities and failures of complex systems. Int. J. Eng. Educ., 19(1):206–212, 2003.

[27] Mark W. Maier. Architecting principles for systems-of-systems. Systems Engineering, 1(4):267–284, 1998.

[28] M.W. Maier. Research challenges for systems-of-systems. In Systems, Man and Cybernetics, 2005 IEEE International Conference on, volume 4, pages 3149–3154, Oct 2005.

[29] M. Merabti, M. Kennedy, and W. Hurst. Critical infrastructure protection: A 21st century challenge. In Communications and Information Technology (ICCIT), 2011 International Conference on, pages 1–6, 2011.

[30] B. M. Michelson, "Event-driven architecture overview," Patricia Seybold Group, Tech. Rep., 2006.

[31] T. Mouelhi, F. Fleurey and B. Baudry. "A Generic Metamodel For Security Policies Mutation," *Software Testing Verification and Validation Workshop, IEEE Intl Conf. on* , vol., no., pp.278 - 286, 2008.

[32] H. Mouratidis, P. Giorgini: Secure tropos: a security-oriented extension of the tropos methodology. Int J Softw Eng Knowl Eng 17(2):285309, 2007

[33] OMG, UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems V 1.1, OMG Std., 2011.

[34] S. Paunov, J. Hill, D. Schmidt, S. Baker, and J. Slaby, "Domain-specific modeling languages for configuring and evaluating enterprise dre system quality of service," in 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems, 2006.

[35] D. C. Schmidt, M. Stal, H. Rohnert, and F. Bushmann, Pattern-oriented Software Architecture: Patterns for Concurrent and Networked Objects, Volume 2. Wiley, 2000.

[36] Florian Skopik, Albert Treytl, Arjan Geven, Bernd Hirschler, Thomas Bleier, Andreas Eckel, Christian El-Salloum, and Armin Wasicek. Towards secure time-triggered systems. In Proc of the 2012 Intl Conf on Computer Safety, Reliability, and Security, pages 365–372, 2012.

[37] A. Waller and R. Craddock. Managing runtime re-engineering of a system-of-systems for cyber security. In System of Systems Engineering (SoSE), 2011 6th Intl Conf on, pages 13–18, 2011.

[38] David Warshawsky and Dimitri Mavris, Choosing Aggregation Functions for Modeling System of Systems Performance, Procedia Computer Science, Volume 16, 2013, Pages 236-244.

[39] S. Wenzel, D. Poggenpohl, J. Jürjens, and M. Ochoa. Specifying model changes with UMLchange to support security verification of potential evolution. *Comput. Stand. Interfaces* 36, 4 (2014), 776-791.

[40] Zhizhong Zhang, Chuan Wu, and David W.L. Cheung. A survey on cloud interoperability: Taxonomies, standards, and practice. SIGMETRICS Perform. Eval. Rev., 40(4):13–22, 2013.