Research Reports in Software Engineering and Management 2016:01 ISSN 1654-4870

Proceedings of the 2nd edition of Swedish Workshop on the Engineering of Systems of Systems (SWESOS 2016)

CHALMERS | UNIVERSITY OF GOTHENBURG

Patrizio Pelliccione and Jan Bosch

Department of Computer Science and Engineering

Research Reports in Software Engineering and Management No. 2016:01

Proceedings of the 2nd edition of Swedish Workshop on the Engineering of Systems of Systems (SWESOS 2016)

Patrizio Pelliccione and Jan Bosch

Department of Computer Science and Engineering
CHALMERS | University of Gothenburg

Gothenburg, Sweden 2016

Proceedings of the 2nd edition of Swedish Workshop on the Engineering of Systems of Systems (SWESOS 2016)

Report no 2016:01
ISSN: 1651-4870
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden

Telephone + 46 (0)31-772 1000

Göteborg, Sweden 2016

SWEDISH WORKSHOP ON THE ENGINEERING OF SYSTEMS OF SYSTEMS (SWESOS 2016)

organized by

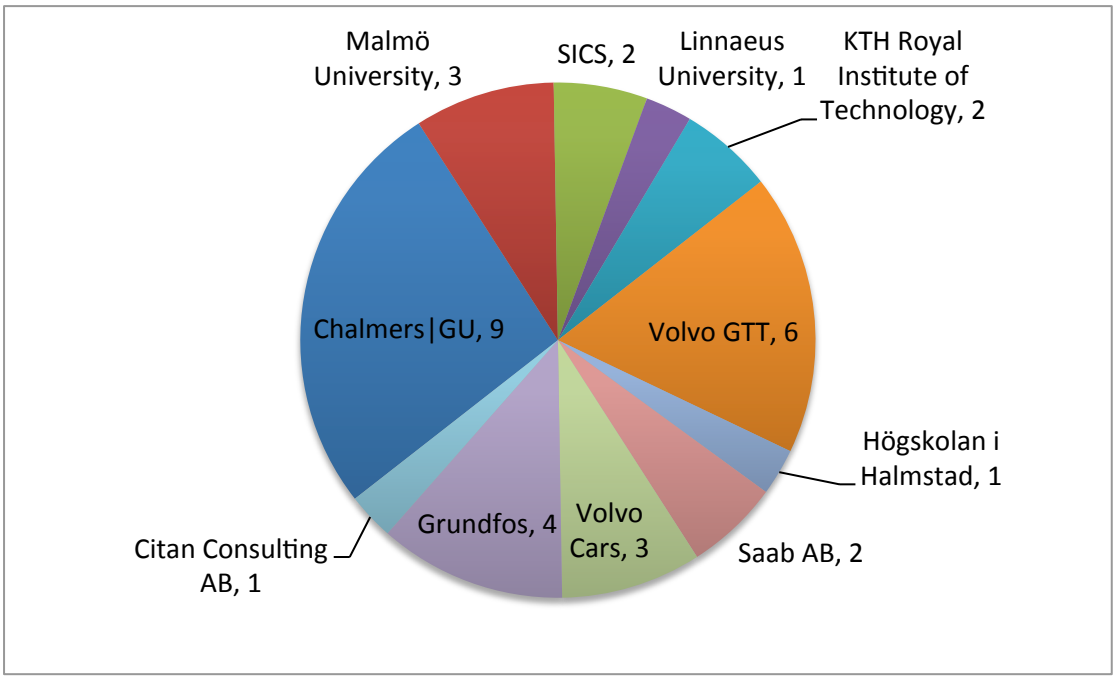
Patrizio Pelliccione and Jan Bosch

Gothenburg, September 9, 2016, Lindholmen

http://swesos2016.github.io/

CHALMERS | GÖTEBORGS UNIVERSITET

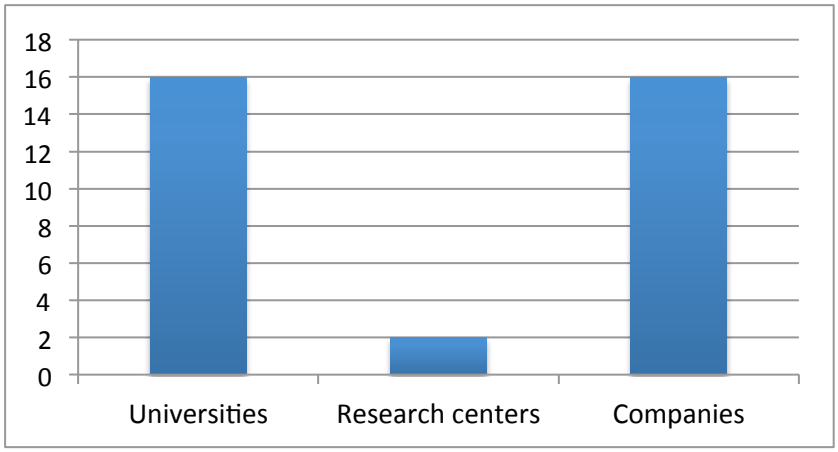
OVERVIEW OF PARTECIPANTS



Registered participants

34 people registered!

- 6 Universities
- 1 research center
- 5 companies



Systems of Systems Concepts for Cars

Eilert Johansson, Tony Larsson, Maytheewat Aramrattana, Patrizio Pelliccione, Magnus Ågren, Göran Jonsson, Rogardt Heldal,

Next Generation Electrical Architecture (NGEA) project
VINNOVA Diarienummer 2014-05599

A System of Systems (SoS) is a collection of often pre-existing and/or independently owned and managed systems that collectively offer a service that is based on their collaboration [4, 3, 5]. Prominent examples of SoSs include intelligent transport systems, integrated air defense networks, applications in healthcare and emergency services. The units that compose an SoS are systems themselves and are called constituents. SoSs may be formed and evolve as triggered by changes in their operating environment and/or in the goals of the autonomous constituent systems [3, 2]. The overall SoS evolution might affect the structure and composition of the constituents, functionalities offered, and/or the functionalities quality. Collaboration between SoSs enables new capabilities, but interdependency implies sensitive to the correctness of the information given to other systems, and that failures can cascade throughout the SoS, creating additional system failures or development delays.

Future transportation systems will be a heterogeneous mix of items with varying connectivity and interoperability. A mix of new technologies and legacy systems will co-exist to realize a variety of scenarios involving not only connecting cars but also road infrastructures, pedestrians, cyclists, etc. In other words, future transportation systems can be seen as a system of systems, where each constituent system can act as a standalone system, but the cooperations among the constituent systems enable new emerging and promising scenarios. Compared to a traditional integrated system, a constituent system within a System of Systems (SoS) has a value in itself and can be used outside the SoS context [1].

When considering SoSs in the automotive domain two different points of view might be considered:

- The viewpoint of the car as a constituent of the SoS, which aims at giving an answer to this question: *How to engineering a car so to be part of a system of systems?*
- The viewpoint of the SoS, which aims at giving an answer to this question: *How to engineer the SoS so that the collaboration among various constituent systems will achieve the SoS goals?* It is important to note that the SoS is owned and evolved by different organizations and constituents of a SoS are most often than not at different points in their life cycles.

This paper focuses on the first viewpoint and takes a bottom-up analysis of technologies that may be needed in future systems of systems for cars. In the

following, we report some essential building blocks necessary to enable future transportation systems.

Distributed end-to-end functionality

A functionality can be distributed, not only between nodes in vehicles, but also between nodes outside the vehicles such as cloud services, other vehicles and infrastructures, etc. Connected vehicles can benefit a lot from having access to cloud services like cloud computing or information from infrastructures and vehicles aggregated in the cloud. A cloud service, or cloud functionality, refers to a network centric service available via the Internet, which extends an existing function in a vehicle, or enable new functionality enabled by cloud data. Therefore, a cloud function is a function that benefits a car and/or its driver by utilizing cloud services mentioned above. For instance, utilizing external data from the cloud to smoothen the speed profile of the vehicle, and consequently reduce the fuel consumption.

Functional safety

Functional safety requirements are expected to apply for functions outside the car, but to be able to handle severity issues in a satisfactory way, one can expect that they not will include operational functions - mainly functions in strategical and tactical planning horizon. However it is important to understand the implication on system design and functional distribution for functional safety.

Services

Services implemented as distributed end-to-end functions will benefit from the possibility to dynamically load software to the on-board electrical architecture. Dynamically loaded software may be executed in one or several physical nodes, and virtual machines may be essential to ensure cyber-security, functional safety and compatibility. Services may also be extended and made version specific with use of data and software in the cloud.

Connectivity

Sufficiently dependable connectivity is essential to enable the expected service level in different places in the System of systems. Making connectivity sufficiently dependable may be possible through the use of many different channels such as through vehicular communication network, Internet, the driver's nomadic devices, etc. However, there has still to be on-board functions that handle graceful degradation of services when connectivity is limited, delayed, or not available at all. Thus, regardless of the availability of the connectivity, the user shall experience a robust behaviour of the functions, especially safety-related functions.

Interoperability

Interoperability is the ability of diverse systems to work together. This general definition has been conjugated in many different ways based on the reference application area and on the many different factors and aspects characterizing them. Interoperability involves standards, protocols, and integration and adaptation of interfaces to enable the effective and efficient communication between constituent systems. Interoperability is the ability of two or more constituent systems that are part of SoS to exchange information and to use the information that has been ex-changed. Unambiguous interpretation of shared data between systems is necessary for interoperation, but it is not sufficient. Despite standards for shared data that provides specification with the objective to enhance the functionality and interoperability, the data encoded using these standards are not necessarily interoperable. For instance, concepts that have the same labels, and somehow even the same meaning, can be used completely differently in different applications. This is for instance the case of the label "speed" within a car that can have different meanings in different applications or contexts unless the semantics is very clearly defined and acted on.

Final remarks

As shown by the concerns emerging from the concepts spanning the entire system of systems, engineering efforts are not only needed on the different system levels, but also on the level of the system of systems as a whole. With potentially each constituent system being developed by a different organization, a separate set of questions arise with respect to the coordination of engineering efforts:

- How shall system of system goals and requirements on these be defined?
- How shall creation of requirements on the system of systems as a whole be coordinated?
- What processes are needed to handle development requests between separate systems?
- How shall development of interoperability specifications be coordinated?
- How shall the SoS services be supervised, managed and maintained over the different life cycles of the constituents?

Within the project we will try to give an answer to these questions.

Acknowledgement

This work was partially supported by the NGEA Vinnova project

References

1. Jakob Axelsson. Systems of Systems research and innovation agenda. SoS Agenda. `https://www.sics.se/sites/default/files/pub/sics.se/upload/groups/sse/sos_agenda.pdf.`, 2015.
2. J. Boardman and B. Sauser. System of systems - the meaning of of. In *2006 IEEE/SMC International Conference on System of Systems Engineering*, pages 6 pp.–, April 2006.
3. John Fitzgerald, Peter Gorm Larsen, and Jim Woodcock. *Foundations for Model-Based Engineering of Systems of Systems*, pages 1–19. Springer International Publishing, Cham, 2014.
4. M. Jamshidi. System of systems - innovations for 21st century. In *2008 IEEE Region 10 and the Third international Conference on Industrial and Information Systems*, pages 6–7, Dec 2008.
5. Claus Ballegaard Nielsen, Peter Gorm Larsen, John Fitzgerald, Jim Woodcock, and Jan Peleska. Systems of systems engineering: Basic concepts, model-based techniques, and research directions. *ACM Comput. Surv.*, 48(2):18:1–18:41, September 2015.

System-based Safety Analysis of Automotive SoS Applications

Avenir Kobetski and Jakob Axelsson

Swedish Institute of Computer Science
{avenir.kobetski, jakob.axelsson}@sics.se

Abstract. Systems of systems (SoS) typically consist of complex and heterogeneous systems, such as humans, machines, servers, etc., interacting with each other in sometimes unpredictable ways. While this is challenging from the functional perspective, it is even more so when it comes to analyzing such emergent properties as safety. In this work, a systems theoretic approach to safety analysis was evaluated on an example SoS application, consisting of trucks driving in cooperation (platoon). SoS-specific characteristics, as well as comparison with currently employed standards, such as ISO 26262, were included into the evaluation. One conclusion of this work is that systems theoretic approaches are valuable, yet they might need some adjustments to SoS specifics.

1 Introduction

The term *systems of systems (SoS)* started to become relevant some 20 years ago, and accelerated as a research area during the last decade. A distinguishing feature of an SoS is that the elements, or constituent systems, exhibit an operational and managerial independence [1], meaning that they can operate outside the SoS context, and have different owners. They choose to collaborate in order to achieve a common goal, manifested as an emergent property of the SoS, i.e. a property not existent in any of its parts in isolation.

With the decreasing cost of communication technology, bringing along a surging number of applications of connected devices, SoS is no longer a far-fetched dream, but rather a reality within a wide variety of application domains. However, while technological development leads the current evolution of the SoS field, non-functional properties, such as safety and security, seem to be lagging behind [2]. This leads to the conclusion that there is a need for research on methods for safety analysis of SoS.

Lately, different SoS ideas have been gaining popularity within the automotive industry, ranging from route planning to intelligent intersection management systems, etc. Vehicle platooning, where a lead truck is followed by a number of other vehicles that are driven more or less autonomously at a very short distance between each other, has already been tried out in a number of test applications. The trucks typically communicate using short-range radio to synchronize their movements and keep the right distance.

The motivator for platooning is primarily to improve fuel consumption by reducing aerodynamic drag, which is good both for the economy of the truck

operator and for the environment. However, due to the automation and the short distances between the trucks, safety obviously becomes an issue. The whole point of platooning is to move the trucks outside their normal safety envelop. Clearly, the platoon is an SoS since each truck can also operate outside the platoon, and the trucks have different producers and owners.

The automotive industry has a long tradition in improving safety, and the best practices have recently been standardized as ISO 26262. In this standard, hazards are classified at different safety integrity levels based on the associated risk, and this classification is then used to derive requirements on components and on the product life-cycle processes. The focus in applying the standard is for a vehicle manufacturer to ensure that their product is safe to use.

However, when the product is to become a part of an SoS, carrying out the safety analysis only on the product is not sufficient. As stated in [2], safety is an emergent property that has to be dealt with on the level of the SoS. In the case of the vehicle platoon, this means that an analysis has to be carried out at the platoon level to identify principles for the safety of the SoS, and then these principles have to be translated to safety goals and requirements on the individual trucks.

In the following, specific challenges of SoS safety analysis are outlined, followed by a presentation of a systems theoretic approach to safety analysis. The method is shortly exemplified with excerpts from a case study on a platooning application, carried out by SICS in collaboration with Volvo Group Trucks Technology (GTT).

2 Challenges

When it comes to analyzing safety of SoS, there are a number of important challenges, stemming from SoS characteristics. They include the following list:

- *Managerial and operational independence* - since no one owns the platoon, all safety requirements have to be agreed upon by potential participants, who must then take measures to implement these requirements in their products while making the best trade-offs with other requirements on the individual vehicles not related to their use in the platooning SoS.

- *Evolveability* - the high-level design or constitution of the SoS, e.g. which vehicles are involved and how they interact, is expected to change over time. Thus, the safety analysis needs to be modular in its nature, providing for flexible re-analysis when operating conditions of an SoS change.

- *Partial design / Missing requirements* - when designing individual systems, it will not always be the case that the exact functional and architectural design of an SoS will be present. Sometimes, it might exist only informally or even not exist at all. Somehow, the safety analysis of SoS needs to take missing requirements into consideration.

- *Complex interactions* - this is a common safety risk in most complex systems. However, it becomes even more pronounced in SoS, since the individual systems are expected to make trade-offs between their own gain and the goals of the SoS. This will, in turn, affect safety aspects.

- *Socio-technical systems* - Since SoS typically belong to the class of socio-technical systems, the safety analysis should include not only interactions between systems (or machines), but also between humans (e.g. operators, decision makers, etc.), organizations, as well as the actual software and hardware systems.

- *Emergent properties* - the safety of SoS should be analyzed not only on product level, but also on higher hierarchical levels, thus considering how the emergent properties affect both the participating and surrounding systems.

In conclusion, it seems imperative to consider the SoS as a whole when performing a safety analysis. This calls for a safety analysis method based in systems thinking.

3 Systems Theoretic Safety Analysis

Systems theory was developed for complex systems with a certain degree of underlying structure (i.e. non-randomness). In general, systems approach assumes that some system properties can only be treated if the system is analyzed in its entirety, taking into account both social and technical aspects [3]. Hierarchy, emergence, communication, and control are key concepts in such analysis.

Systems-Theoretic Accident Model and Processes (STAMP) is a systems theoretic approach to safety analysis [4]. In contrast to traditional accident causality models, such as *Failure Mode and Effect Analysis (FMEA)*, the focus is shifted from chains of failure events to a systemic view of possible undesired losses. Interactions between different (sub)systems are not only considered, but receive special attention, with the focus being not on designing "fail-free" components in isolation, but rather on preventing losses in the entire system (or SoS) through appropriate control mechanisms. Treating safety as a control problem between interacting components on different hierarchical levels is at the heart of the STAMP approach.

In STAMP, systems are treated as interrelated, dynamic processes that are continuously adapting to changing internal and external (environmental) conditions. Accidents are considered to be the result of flawed control processes, involving interactions among people, societal and organizational structures, engineering activities, and physical system components [4]. In this respect, STAMP seems perfectly suitable for SoS applications.

The STAMP workflow starts with creating a control model of a system, consisting of a number of processes and control algorithms, affecting each other through sensors, actuators, and direct or indirect communication. Each controller is assumed to have some sort of mental model of the process that it is

supervising. In the case of SoS, the challenge of partial requirement definition will be an important obstacle to model construction. However, some preliminary results addressing this issue have been proposed [5], based on the idea of iterative model creation by extracting information from existing, but most often vague and undocumented, operation concept descriptions. The question of applying these ideas to SoS setting, with different producers and owners of the constituent systems, remains to be investigated. Once the model is constructed, it is analyzed for possible safety threats, using a structured approach, supported by a number of tools and checklists.

4 Discussion and Conclusions

The STAMP approach was applied to a simple platooning application, consisting of two trucks, run at a short distance from each other. The SAE level of driving automation [6] was assumed to be 2-3, i.e. partial or conditional automation, with active drivers in both vehicles. While the first driver fully controlled its truck, the second was only responsible for the steering, while the speed was adjusted by an advanced cruise controller (CC). Besides the driver and CC, a central route management system was assuming responsibility for the overall functioning of the platoon.

The case started with a rather informal definition of the responsibilities of the constituent systems, and a possible concept of operations was developed and formalized into a control model, in cooperation with Volvo GTT. The model was analyzed using the STAMP approach, resulting in a number of safety (and even security) related requirements. The approach was also compared with the current practice of ISO 26262.

In summary, STAMP seems to be a promising method for safety analysis of SoS, with potential benefits increasing along with the complexity and heterogeneity of the constituent systems. However, in some areas, not least when it comes to missing requirements, independence, and evolveability, more work on method development might be needed.

References

1. Maier, M. W.: Architecting Principles for Systems-of-Systems. INCOSE International Symposium, vol. 6, nr. 1 (1996)
2. Axelsson, J.: Systems-of-systems for border-crossing innovation in the digitized society: A strategic research and innovation agenda for Sweden. Swedish Institute of Computer Science (SICS), Report T2015:07, (2015)
3. Ramo, S., St Clair, R.: The systems approach. Systems Concepts: Lectures on Contemporary Approaches to Systems, pp. 13–32 (1973)
4. Leveson N: Engineering a safer world: Systems thinking applied to safety. Mit Press (2011)
5. Fleming, C.H.: Safety-driven Early Concept Analysis and Development, Ph.D. thesis, MIT (2015)
6. SAE International's Levels of Driving Automation for On-Road Vehicle, http://www.sae.org/misc/pdfs/automated_driving.pdf (2016)

A Model-based Development, Execution, and Evolution Platform for Dependable Cyber-Physical System-of-Systems

DeJiu Chen

Mechatronics, Department of Machine Design, KTH Royal Institute of Technology, Sweden

chen@md.kth.se

Abstract. The inherent nature of cyber-physical (CP) system-of-systems (SoS) in terms of open dynamic configuration and interoperation of heterogeneous uncorrelated agents, safety- and time-criticality, requires several radical engineering paradigm shifts. A significant difference in comparison to contemporary monolithic systems is related to how the development, execution and evolution of SoS go across the boundaries of traditional technological and managerial domains. This paper discusses a model-based approach to an integrated development, execution, and evolution of dependable systems with knowledge-in-the-loop. It aims to pave the way for a next generation ICT platform for qualified CP SoS in general and intelligent transport systems in specific. Using EAST-ADL as one base technology, such an ICT platform addresses the synergy of model-based system development (MBD), well-managed product lifecycles and intelligent post-deployment services across the eco-systems and lifecycle phases. The goal is also to support formal quality assurance in regard to *a priori* unknown operational situations as well as situations where a degree of uncertainty is intrinsic in the state description.

Keywords: Systems-of-Systems (SoS), Cyber-Physical (CP), Model-Based Development (MBD), EAST-ADL, Knowledge-in-the-Loop.

1 Introduction

A System-of-Systems (SoS) is composed of independent systems that cooperate or collaborate adaptively for dealing with certain social and technological issues (Maier 1998; Sage 2001; Nielsen 2015). These independent systems (i.e. the constituent systems) execute their tasks autonomously as intelligent agents in multi-agent system (Wooldridge 2002), while being functionally and technologically heterogeneous with different capabilities. The operation of SoS often relies on advanced information and communication technologies for the discovery, configuration and provision of services. A SoS can become cyber-physical (CP) system by having both physical dynamics or energy flows under control and the corresponding control and cognitive loops

Swedish Workshop on the Engineering of Systems of Systems (SWESoS) 2016.

across its constituent systems. Motivated by various societal and economic benefits, Cyber-Physical (CP) System-of-Systems (SoS) is becoming increasing popular as the underlying technology for intelligent transport systems, automotive vehicles, manufacturing systems (i.e. Industry 4.0), etc.

As an overall requirement, a SoS must be able to deal with uncertainties in its operation and lifecycle and accordingly to adapt the planned missions, configuration and maintenance tasks to cope with changes of operational conditions. In particular, for an effective management of safety, security, and their complex interplay, a systematic treatment of emergent behaviors and related system anomalies due to the open dynamic configuration of heterogeneous agents becomes necessary. This calls for quality assurance by formal methods and tools as well as advanced run-time services. To this end, a significant difference of SoS in comparison to contemporary monolithic systems is related to how the development and operation of SoS should go across the boundaries of organizations, domains and disciplines.

This paper discusses a model-based approach to an integrated development, execution, and evolution of dependable CP SoS, primarily in an automotive context. It aims to pave the way for a next generation ICT platform for qualified cyber-physical system-of-systems in general. In regard to the uncertainties of SoS, such an ICT platform promotes effective quality assurance through a synergy of formal system specification and intelligent system services. The approach emphasizes a knowledge-in-the-loop quality assurance process that goes across multiple eco-system and lifecycle phases. In particular based on EAST-ADL, such a process combines formal methods and tools for development-time quality planning and control with intelligent services for post-deployment time knowledge inference, self-verification and validation.

The rest of this chapter is structured into the following sections: Section 2 elaborates the challenges and research needs by related work. Section 3 introduces the envisioned ICT platform and its base technology. The paper concludes with Section 4.

2 Related Work

SoS exhibits emergent behaviors arising from complex interactions of constituent systems as well as from unmanaged system anomalies and security attacks. As many of such behaviors are a priori unknown, they may never be fully specified, analyzed or tested during system development. Anthony (2007) identifies two classes of *SoS emergence*. Emergent behavior is defined as first order emergence, and is considered to be the run-time characteristics of SoS, while the ability of related heterogeneous systems to evolve is described as second order emergence (e.g. a form of design-time change of functional or technical properties, over many iterations). This second order emergence implies specific challenges in system design and quality management, arising in the communication and collaboration across disciplines and organizations.

For dealing with the partially unknown or inaccurate definition of open dynamic configuration and interoperation conditions in design-time, the specification of systems with abstract goals, control policies and contracts together with the provision of related intelligent services and platform support will be necessary. For example, Silva

(2015) proposes a mission goal description based approach to the identification of required capabilities for the constituent system, operations, connections, emergent behavior, among other elements that characterize a SoS. Bryans (2014) has proposed a contractual description of constituent systems interfaces, to address the imprecision and uncertainty inherent in the description of constituent systems. For quality assurance and certification, such contractual support needs to be defined and managed seamlessly along with the lifecycles of system development, componentization and maintenance.

In recent years, domain specific frameworks have been developed for model-based development (MBD) of cyber-physical systems in many industrial domains. There are many modeling frameworks that can be used to support the description and management of SoS, such as SysML[1], AADL[2] and EAST-ADL[3]. For SoS, additional modeling constructs to support the expression of desired or prohibited emergent properties, which are normally not be able to be predicted accurately at system development time, become necessary. In regard to this, COMPASS[4] represents an approach that extends SysML to support a model-based engineering for SoS requirements. The approach also provides a framework, consisting of a collection of viewpoints, for a systematic reasoning about faults in SoS. To cope with the evolutionary nature of SoS in terms of evolving, adaptive and iterative life-cycle, the DANSE[5] project has contributed with methodological support based on a formal semantics for SoS interoperations. Nevertheless, there is still no consensus on common constructs for the description or characterization of SoS, in particular in regard to the software services for dynamic configuration and coordination control (Guessi 2015). The challenges are also related to the difficulties in industrial adoptions of the concepts and technologies.

As a generic constraint, any operation-time reasoning must be not only fast and resource efficient, but also verifiable and validatable in the sense that for safety-critical scenarios the formal quality assurances of decisions have to be supported. This calls for run-time support for data management, architecture for deterministic execution, and services for quality of service guarantee. One base technology to be exploited here is Autonomic Computing (AC) which is essentially concerned with the deployment and management of run-time reasoning schemes (Eze 2012). This includes support mechanisms e.g. in the case of policy based systems there is a need for infrastructure to hold the policy repository and to load them into components. Adaptation decisions at runtime require exact and actual information about the current system state (Pelc 2009; 2011), as well as data from the design models such as in regard to the software allocation and timing constraints, but being evolved along with the adaptations (Bencomo 2014).

A great many techniques have been developed and explored for advanced verification and validation of complex systems, both statically at design-time and dynamically at run-time. As the exhaustive exploration of the complete state space, even from

---

1 http://www.omgsysml.org/
2 http://www.aadl.info/
3 http://www.east-adl.info/
4 http://www.compass-research.eu/
5 http://www.danse-ip.eu/

the view of one single agent in SoS, is not feasible in practice. One particularly promising type of model checking for SoS is the so-called on-line model checking, as explored in Althoff (2014). In this approach, the resulting combined state space is continuously monitored against critical safety properties, or used to compute safe trajectories. Another approach that combines model-checking with dynamic model validation and creation based on machine-learning is learning-based testing (LBT) (Meinke, 2014). A further approach is adaptation-based programming (ABL) (Groce, 2012; Bauer, 2013), which tries to take advantage of machine learning methods including reinforcement learning for the improvement of testing. New methodological approaches to operational risk assessment include statistical analysis of near-miss and incident data using Bayesian theory to estimate operational risk value and the dynamic probabilities of accidents sequences having different severity levels (Meel, 2006); and the application of simulation models to analyze scenarios using dynamic fault trees (Xu, 2004). However, when dealing with complex CP SoS it is often impossible even to establish the probabilities of interest. In such cases approximations (one´s belief) of the probabilities of interest must be calculated instead. Useful techniques in this regard, include Dempster-Shafer Theory (SAFESPOT 2007) and Belief Propagation (Mooij 2008).

3 Concepts and features of the ICT platform

The envisioned ICT platform aims to act a framework consisting of models, methods, tools and services, and thereby provide the capability of enabling a qualified continuous development of CP SoS. It follows the *knowledge-in-the-loop* approach proposed by Chen (2015), where qualified ontological models are synthesized automatically for effective model-predictive operation-time self-managed decision making and other knowledge inference tasks. See Fig. 1 for an overview of the concept.

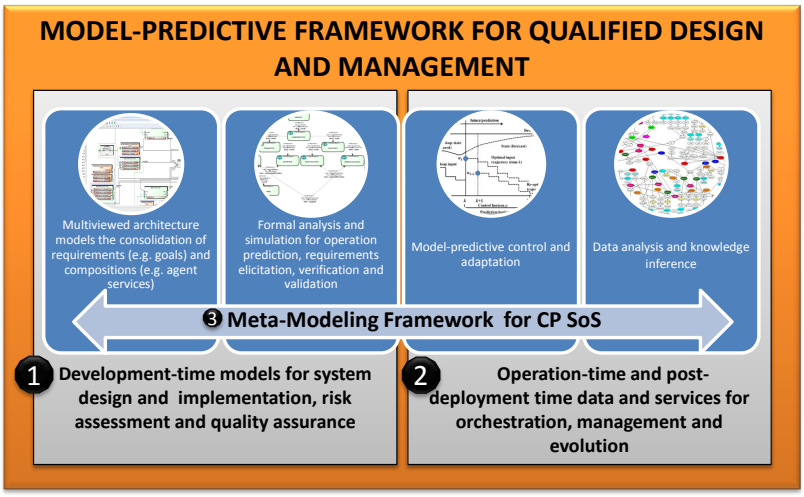


Fig. 1. Conceptual overview of the ICT platform for CP SoS.

As shown in the figure, the scope of support ranges from (1) *Development-time knowledge and models for system design and implementation, risk assessment and quality assurance*; to (2) *Operation-time and post-deployment time data and services for orchestration, management and evolution*; and to (3) *A meta-modeling framework for knowledge modeling, inference, integration and management.* For the development-time, the support is centered on a multi-viewed description of CP SoS requirements (e.g. goals) and compositions (e.g. agent services) with formal analysis and simulation for risk management and quality assurance. The operation-time and post-deployment support is related to model-predictive system control and adaptation (e.g., for situation awareness, orchestration and risk assessment), as well as model-based operation data analysis and knowledge inference (e.g. for system validation and knowledge enrichment). One particular feature of this ICT platform is its support for automated synthesis of ontological models and necessary system services and contracts for operation monitoring and control with underlying intelligent filters and networks (e.g. Kalman Filter, Artificial Neural Network). To this end, the meta-modeling framework is used to consolidate, connect, and transform the knowledge and models across development-time, operation-time and post-deployment. This allows an evolutionary development of CP SoS with continuous flows of operation-time observations and other post-deployment knowledge back to the development and maintenance phases for anomaly treatment, system validation, maintenance and evolution.

The modeling framework provides with necessary formalisms, methods and tools, operation-time and post-deployment data and services for CP SoS development, operation, management and evolution. One key base technology to be adopted for this ICT platform for CP SoS is the modeling framework EAST-ADL (Kolagari 2016; Chen 2011), which represents one key European initiative towards a standardized multi-viewed description of automotive electrical and electronics systems. It is a result of a series of consecutive projects: ITEA EAST-EEA, EU FP6 ATESST I and EU FP7 ATESST II, and EU FP7 MAENAD. By integrating many generic system description frameworks (e.g. SysML) and automotive specific methodological and technological considerations (e.g. RIF/ReqIF), EAST-ADL, through its meta-model, constitutes a fundamental knowledge-model for the descriptions of cyber-physical systems in general. For example, it allows a wide range of functional safety related concerns (e.g. hazards, faults/failures, safety requirements) to be declared and structured along with the lifecycle of nominal systems. Although constituting a very good basis for capturing and formalizing various system aspects, current EAST-ADL does not provide any explicit support for the modeling and analysis of CP SoS with regard to emergent behaviors, dynamic risk assessment, uncertainties and related design verification and validation issues. Therefore, regarding the support for CP SoS, meta-model extensions and specializations of EAST-ADL need to be developed. These include additional modeling constructs to support the expression of run-time ontologies for effective reasoning for desired or prohibited emergent properties, which are normally not be able to be predicted accurately at system development time, as well as additional

methods for automated transformation and traceability between design-time and run-time models.

4 Conclusion

A SoS is a type of open system in stark contrast with conventional monolithic system with well-defined boundary and isolated and interactions. This requires several radical engineering paradigm shifts. While the challenges are currently being tackled by different industry and research efforts, significant issues remain in the areas of quality assurance, uncertainty and risk treatment, change and evolution management, etc. This paper discusses a model-based approach to an integrated development, execution, and evolution of dependable systems, aiming to pave the way for a next generation ICT platform for CP SoS, primarily in an automotive context. The approach emphasizes a knowledge-in-the-loop decision-making that goes across multiple ecosystem and lifecycle phases. In particular based on EAST-ADL, it provides a modeling framework for integrating design-time formal analysis and simulation, with operation-time model-predictive system control and adaptation and other post-deployment time data analysis and knowledge inference capabilities.

References

[Maier 1998] Maier, M.W. :"Architecting Principles for Systems-of-Systems". Systems Engineering. 1(4), pp. 267-284, 1998.

[Sage 2001] Sage, A.P., Cuppan, C.D. :"On the Systems Engineering and Management of Systems of Systems and Federations of Systems". Information, Knowledge, Systems Management 2 (4): 325–345. Volume 2 Issue 4, December, 2001.

[Nielsen 2015] Nielsen, C.B. et al.:"Systems-of-Systems Engineering: Basic Concepts, Model-based Techniques, and Research Directions". ACM Computing Surveys (CSUR), Volume 48 Issue 2, November 2015. ACM New York, NY, USA.

[Wooldridge 2002] Wooldridge, M. :"An Introduction to MultiAgent Systems". John Wiley & Sons. 2002. ISBN 0-471-49691-X.

[Anthony 2007] Anthony, R., Butler, A., Ibrahim M.:"Exploiting Emergence in Autonomic Systems" Autonomic Computing: Concepts, Infrastructure, and Applications, Editors Parashar M and Hariri S, Taylor and Francis, 2007, USA.

[Silva 2015] Silva E.; Batista, T.; Oquendo, F.: "A Mission-Oriented Approach for Designing System-of-Systems". Proc. of the 10th System-of-Systems Engineering Conference (SoSE), May 2015, pp. 346-351.

[Bryans 2014] Bryans J., Fitzgerald J., Payne R., Miyazawa A., and Kristensen K.: "SysML Contracts for Systems of Systems". June 2014, IEEE Systems of Systems Engineering Conference 2014.

[Chen 2015] Chen D., Meinke K., Östberg K., Asplund F., Baumann C.: "A Knowledge-in-the-Loop Approach to Integrated Safety&Security for Cooperative System-of-Systems". Proc. of IEEE 7th International Conference on Intelligent Computing and Information Systems, ICICIS'15, Dec 12-14 , 2015.

[Guessi 2015] Guessi, M. et al.: "Characterizing Architecture Description Languages for Software-Intensive Systems-of-Systems". Proc. of the 3rd ACM/IEEE ICSE Int. Workshop

on Software Engineering for Systems-of-Systems (SESoS), Eds. F. Oquendo et al., May 2015, pp. 1-8.
[Eze 2012] Eze, T. O., et. al.: “Autonomic Computing in the First Decade: Trends and Direction”. The Eighth International Conference on Autonomic and Autonomous Systems (ICAS 2012), St. Maarten, Netherlands Antilles, pp. 80-85, IARIA, March 25-20, 2012.
[Pelc 2009] Pelc M., et. al.: “Policy Supervised Exact State Reconstruction in Real-Time Embedded Control Systems”. Proceedings of ACD 2009, Zielona Gora, Poland (electronic materials). 2009.
[Pelc 2011] Pelc M. :”Architecture For Hierarchical Policy-Supervised Fault Detection Component / Middleware”. Proceedings of DPS.11, Measurements Automatic and Control Journal, Gliwice, Poland, nr 9, pp. 1005-1010, 2011.
[Bencomo 2014] Bencomo, N. et. al. :“Models@run.time. Foundations, Applications, and Roadmaps”. Programming and Software Engineering, Series Volume 8378, Springer. 2014.
[Althoff 2014] Althoff, M. et. al.: “Online Verification of Automated Road Vehicles Using Reachability Analysis”. IEEE Trans. on Robotics 30(4), 2014.
[Meinke 2014] Meinke, K. et. al. ‘Incremental Learning-Based testing for Reactive Systems’, Proc. Int. Conf. on Tests and Proofs TAP 2011, LNCS 6706, Springer Verlag, 2011. IEEE Transactions on Robotics 30(4): 903-918. 2014.
[Groce 2012] Groce, A. ‘Learning-based test programming for programmers’. Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change, pp. 572-586. Springer Berlin Heidelberg, 2012.
[Bauer 2013] Bauer, T.. "Adaptation-based programming." PhD dissertation, Oregon State University. 2013.
[Meel 2006] Meel, A. “Plant-specific dynamic failure assessment using Bayesian theory”. Chemical Engineering Science 61, pp. 7036–7056. 2006.
[Xu 2004] Xu, H.: “Combining Dynamic Fault Trees and Event Trees for Probabilistic Risk Assessment”. Reliability and Maintainability, Annual Symposium - RAMS ISBN: 0-7803-8215-3, pp. 214-219. 2004.
[SAFESPOT 2007] SAFESPOT, Cooperative Systems for Road Safety; SP1 SAFEPROBE; Deliverable No. D1.3.3. 2007.
[Mooij 2008] Mooij, J. “ Understanding and Improving Belief Propagation”, Doctoral Thesis, Radboud University. 2008.
[Kolagari2016] Kolagari T. ., Chen, D., Lanusse, A., Librino, R., Lönn, H., Mahmud, N., Mraidha, C., Reiser, M.-O., Torchiaro, S., Tucci-Piergiovanni, S., Wägemann, T., Yakymets N.: “Model-Based Analysis and Engineering of Automotive Architectures with EAST-ADL: Revisited”. International Journal of Conceptual Structures and Smart Applications (IJCSSA). Vol 3, Issue 2, July - December 2015. IGI Global Publishing, Hershey, USA. ISSN: 2166-7292; EISSN: 2166-7306.
[Chen2011] Chen, D., Johansson, R., Lönn, H., Blom, H.,Walker, M., Papadopoulos, Y., Torchiaro, S., Fulvio T., Sandberg A.: “Integrated Safety and Architecture Modeling for Automotive Embedded Systems”. e&i - elektrotechnik und informationstechnik, Volume 128, Number 6, Automotive Embedded Systems. Springer Wien, 2011. ISSN 0932-383X / 1613-7620. DOI 10.1007/s00502-011-0007-7.

Collaborative Innovation In Business Ecosystems: A Strategy Selection Framework

Helena Holmström Olsson[1] and Jan Bosch[2]

[1] Malmö University, Faculty of Technology and Society, Nordenskiöldsgatan 1, 211 19 Malmö, Sweden
helena.holmstrom.olsson@mah.se
[2] Chalmers University of Technology, Department of Computer Science & Engineering, Hörselgången 11, 412 96 Göteborg, Sweden
jan.bosch@chalmers.se

Abstract. Today's organizations are highly distributed and dynamic in nature. Similar to 'Systems of Systems' (SoS), they involve communication and coordination between its different parts and typically, they evolve over time as new organizational units, roles and people are introduced. From this perspective, the notion of business ecosystems has many common characteristics with what is referred to as 'Systems of Systems' (SoS). In this paper, we explore how innovation takes place in business ecosystems and we see that most innovation strategies involve a mix of internal, collaborative and external elements in which external partners collaborate. Due to the dichotomy in approaches however, companies often fail to select the optimal innovation strategy for the specific innovation challenge at hand. In this paper, we provide guidelines on the optimal selection of strategies.

Keywords: Innovation strategies; collaborative innovation; strategy selection.

1 Introduction

Business success requires intentional management of ecosystem partners [1], [2]. This is true for both large keystone players as well as for smaller players in the network. Especially, and in order to accelerate innovation and value creation, companies need guidelines that help them identify innovation partners, select innovation strategies and successfully combine internal, collaborative and external innovation elements. In many ways, business ecosystems are similar to 'Systems of Systems' (SoS) as they involve distributed organizations that evolve over time, and in which communication and collaboration between different parties is critical.

In this paper, we study innovation in business ecosystems and we conclude that often innovation initiatives include a mix of internal, collaborative and external strategies. As a result, the challenge for companies is not to select among a dichotomy of innovation approaches, but rather to identify the strategy with the optimal

combination of the different innovation elements. In this paper, we address this challenge and we provide guidelines on how to select the optimal innovation strategy.

2 Background

Traditionally, innovation initiatives in software-intensive systems companies are viewed as either internal innovation, such as technology-driven innovation based on ideas generated within a company, external innovation in which companies adopt strategies to capture and expand on ideas created by other stakeholders, or collaborative innovation where a number of stakeholders openly share and benefit from results within the network [3], [4], [5]. First, ***internal innovation*** refers to a situation in which ideas are generated internally within a company, and in which these ideas are then validated and brought to the market either by the company itself or by a partner in the innovation ecosystem [3]. Second, ***external innovation*** refers to situations in which companies adopt strategies that help them capture value that is produced by other stakeholders. Vanhaverbeke et al [4], describe this innovation paradigm as an "option-creation process" that allows innovating companies to sense developments in a wide range of external initiatives in order to learn about new technologies with uncertain payoffs. Third, ***collaborative innovation*** is when companies create different forms of alliances and partnerships with external partners with whom they co-create value. In such partnerships, the intention is to establish close relationships to external innovation partners to have continuity in joint innovation activities and to over time involve them in development of differentiating functionality as part of the core product offering.

3 Research methodology

The findings reported in this paper are based on longitudinal multiple case study research in six software-intensive companies in the embedded systems domain. The research project was initiated in January 2014 and is on-going. During the time of our research, we have conducted a literature review, an interview study in each of the six companies, multiple cross-company workshops at which company representatives from all companies meet, focus group interviews and validation sessions and workshops.

4 Findings

In all case companies, the ***ideation,*** i.e. the creation of new ideas is often conducted internally. Facilitated by strategic investments, innovation incubators and reward systems, the case companies are successful in continuously identifying new business opportunities based on their in-house competence and skills. In concept creation and customer validation activities however, they are increasingly shifting towards more

collaborative practices and all case companies report on the importance of having external partners, such as e.g. customers, involved as soon as there is a concept to test. Also, early customer validation helps the company to efficiently push innovative functionality to the core product offering in order to make it available for the larger customer base. However, and as can be seen in the case companies, they are increasingly moving towards collaborative innovation approaches in order to either benefit from skills similar to their own but inherent in other organizations, or to complement their in-house skills with competence they do not currently possess.

For ***concept creation and customer validation,*** the case companies report on increasingly collaborative approaches. While there are many reasons for this, the primary reason is the attempt to "fail faster" and avoid investing resources in an innovation initiative with little customer value. In all case companies, there are several examples of innovation initiatives with very little market value and where significant R&D resources have been spent on new functionality that have no proven customer value. To avoid this, the companies are moving towards early proof of concept and prototyping techniques that allow fast customer feedback. Also, and in order to facilitate collaborations, several of the companies are creating separate organizational units for innovation, with its primary task to host innovations and stimulate cross-company collaborations.

5 Discussion

Internal strategies to ideation, concept creation and validation are typically selected when a company is the keystone player with a very strong position in the ecosystem and when technological know-how is superior. Our case companies select ***internal strategies*** for the different innovation activities when they know what the solution should look like before engaging in potential collaborative activities to validate the idea. Also, our findings show that internal ideation and concept creation activities are critical to foster creativity and to encourage a constant flow of new ideas. The challenge however, and as experienced in the companies, is to find a balance between a creative innovation culture where you let "thousands flowers bloom" while at the same time identify ways to efficiently kill unsuccessful innovations before they start consuming resources. ***Collaborative strategies*** to ideation, concept creation and validation are selected when there is some level of uncertainty, for example when entering new markets or when developing new service offerings that include stakeholders that were not part of the traditional ecosystem. In such situations, companies typically look to share costs and risks, and they seek additional knowledge that will help them assess a certain situation or need. Also, the collaborative elements are increasingly used to help companies validate whether an innovation adds customer value or not. Finally, ***external strategies*** to ideation and concept creation are selected when a company identifies a promising innovation that can be brought in-house "as is" or by some internal modification. The majority of the case companies continuously scan the market for promising innovations. In Table 1, we summarize our analysis and we provide guidelines on how to select the optimal innovation strategy.

Innovation strategy	Innovation activities
Internal	• Mature markets/technology • Keystone player • Superior know-how
Collaborative	• Emerging markets/technology • Sharing of risks and costs • Non-proven customer value
External	• Complement existing offerings • Minimize development investments • Proven customer value

Table 1. Innovation strategy selection framework.

6 Conclusion

In this paper, we show that although innovation is traditionally viewed as internal, collaborative or external, most innovation initiatives involve a mix of these elements. Also, our research shows that the collaborative element is becoming increasingly important in order for companies to accelerate new product development and maximize the benefits of their ecosystem.

References

1. Bosch, J. (2016). Speed, Data, and Ecosystems: The Future of Software Engineering. IEEE Software, Vol. 33, No.01 - Jan.-Feb., pp. 82-88.
2. Olsson, H.H., and Bosch, J. (2015). Strategic Ecosystem Management: A multi-case study on challenges and strategies for different ecosystem types. In Proceedings of the 41st Euromicro Conference series on Software Engineering and Advanced Applications (SEAA), August 26-28th, Madeira, Portugal.
3. Poot, T., Faems, D., and Vanhaverbeke, W. (2009). Toward a dynamic perspective on open innovation: A longitudinal assessment of the adoption of internal and external innovation strategies in the Netherlands. International Journal of Innovation Management, Vol. 13, No. 2 (June).
4. Vanhaverbeke, W., Van de Vrande, V., and Chesbrough, H. (2008). Understanding the advantages of open innovation practices in corporate venturing in terms of real options. Creativity and Innovation Management, Vol. 17, No. 4.G. Eason, B. Noble, and I.N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529-551.
5. Adner R., and Kapoor R. (2010). Value creation in innovation ecosystems: How the structure of technological interdependence affects firm performance in new technology generations. Strategic Management Journal, Vol. 31, No. 3, pp. 306-333.

Aligning Architectures for Sustainability

Jesper Andersson, Mauro Caporuscio

Linnaeus University – Växjö (Sweden)
jesper.andersson@lnu.se, mauro.caporuscio@lnu.se

Abstract. Industry digitization has drastically changed the competitive landscape by requiring a higher degree of specialization and shorter time to delivery, which affect the design properties a software platform should satisfy. The platform architecture must sustain continuous and rapid change to the business architecture, which in turn is affected by external forces: i.e., forces drive the velocity of change. In this paper we explore the effects of digitization, characterizing internal and external forces that impact on business strategies and trigger the continuos realignment of the platform, and outline a research agenda to mitigate the effects.

1 Setting the context

The continuos increase in system complexity has become an invariant for software engineers. Software systems are often business or mission critical subsystems, responsible for a vast part of the Sociotechnical system's value. A Sociotechnical system connects several technical systems and people who either interact with or is part of the system itself [2]. Indeed, they are defined to explicitly include operational processes and people (e.g., the operators), which are considered inherent parts of the system. Most sociotechnical systems are organizational systems that provide support for an enterprise to achieve their business goals.

Due to their embeddedness in an enterprise, the procurement, development, and deployment of these systems will be heavily influenced by an enterprise's structure and processes. Structures and processes are in-turn affected by changes in the enterprise's environment, e.g., the market or regulations. Enterprises need to adapt quickly to these forces, in order to mitigate risks or take advantage of opportunities they create: *forces drive the velocity of change.*

Osterwalder and Pigneur describe 4 groups of forces that affect a market and indirectly an enterprise's business models [8]. Three groups, *industry*, *market* and *trends* are close to an enterprise, whereas the fourth, *macro-economics*, is a health-monitor to balance strategic decisions. A sustainable enterprise is agile and responsive to changes, continuously adapting its business strategies, manifested in a business architecture. The main challenge, aligning the information technology support with the organizational need, is discussed by Tallon and Pinsonneault [10]. They argue for the importance of agility and alignment for the enterprise's overall performance.

Engineering best practices suggest the exploitation of Enterprise Architecture (EA), which enables enterprises to continually evolve in response to ever-changing environment forces [9]. EA describes the structure of the enterprise in

terms of subsystems and the relationships between them, as well as the relationships with the external environment. Further, EA defines the set of principles guiding the design and evolution of the enterprise. Indeed, EA plays a key role by tracking forces and providing insight into how they affect each area of the enterprise. The platform architecture captures the technology facet of EA, that is, the organization of information technology components within an enterprise. These components provide business critical functions to the enterprise, including its business functions. Thus, it is crucial that the platform architecture is maintained and evolved in response to changes in the business architecture.

2 Co-evolution for Sustainability

It is well-known from systems theory that the health of the overall system depends on the performance of its subystems. The forces discussed above represent points of uncertainties, uncertainties that may lead to risks or opportunities for the enterprise [6]. These causalities are becoming increasingly important for an enterprise mange to maintain market sustainability. Business and platform architecture co-evolution is critical for market sustainability. The key to unlock a possible solution is to develop new organizational capability [7] that harnesses speed and specialization, and addresses the co-evolution requirement. Developing organizational capabilities require that the enterprise attains or develops knowledge about how to develop platform architectures and products that match the business architecture requirements. Below we discuss some capabilities that need to be developed further, namely *People & Processes*, and *Technology & Tools* [7].

2.1 People & Processes

Extend the team – The alignment of business and platform architecture is described as a prerequisite for market success [7]. It is however not guaranteed that the best business architecture supported by the best platform architecture will prevail on the market, which has been proven by numerous counter examples over time. One way forward towards better alignment is to bring business architecture knowledge closer to the platform architects and platform development.

Open-up and out-source – The key market trends that cause uncertainty and opportunity for enterprises are speed and specializations. The consequences are increasingly higher demands on specialized organizational capability. Developing and maintaining specialized capability to support market fragments or niches is not sustainable for most enterprises. Abandoning a fragment or niche is not a straightforward business decision, since the next dominant design always start as a niche. Trading risks in these situations is difficult and out-sourcing for specialized capabilities will be a more sustainable strategy than in-sourcing.

Tighten the loop – Within sociotechnical systems, several interactions occur simultaneously at different levels of abstraction. This inherent complexity causes *uncertainty*, which must be mitigated to achieve sustainable platform architectures. One mitigation approach is to tighten the loop even more, preparing enterprises and there platform architectures for *adaptability* and *evolvability* [11]. *Adaptability* aims at mitigating uncertainty with respect to a system's ability to continuously satisfy a requirement, whereas *evolvability* accommodates risk mitigation strategies that address uncertainty caused by new market forces.

Life-cycle blur – A consequence of tightening the innovation cycle is that the clear separation of development and run time blurs when on-line and off-line activities intertwine to promptly address forces that affect the business architecture, while the system is running.

2.2 Technology & Tools

The people and process capabilities discussed above, can be enabled or inhibited by the platform architecture. It is essential to make the platform architecture compatible with the enterprise to enable its capabilities. "If the parts of an organization (e.g. teams, departments, or subdivisions) do not closely reflect the essential parts of the product, or if the relationship between organizations do not reflect the relationships between product parts, then the project will be in trouble" [4]. To this end, we discuss hereafter a set of key design principles that allow architects to define business and platform architectures that are able to co-evolve. It is worth noting that such design principles are not new, but they have been discussed and taught since decades. Rather, the novelty here is tied up with "why" and "how" we leverage on them to design business and platform architectures centered around enterprise's customers: i.e., *developers*, *architects*, and *stakeholders*.

Isolation and Single Responsibility – Isolation is considered one of the most important design principles, as its systematic application allows architects to slice up the architecture, and organize the teams and responsibilities accordingly. A responsibility is defined as "a reason for change" [5], which in turn is driven by a force (see Section 1). Indeed, when a new force emerges in the market, it will affect the system and make responsibilities of components within the platform change. Hence, the more responsibilities a component assumes, the more forces will affect it. On the long-term, this will lead to a fragile design, and both business and platform architecture degradation.

Robustness, Redundancy and Diversity – Once having a set of autonomous and isolated components, they must be composed together, and collaborate with each other to solve problems. Indeed, it is in collaboration that opportunities and challenges emerge. Components composition puts the system at hight risk, since

the overall behaviour can not be guaranteed in case interaction deviate from the expectation – e.g., due to failures, system overloading, etc. To this extent, it is important to take precautions and be robust with respect to communication [1]. Adhering to *Robustness* principle "be conservative in what you do, be liberal in what you accept from others" improves interoperability among components and facilitates their versioning and evolution.

Fluidity – Due to adaptability and evolvability, the platform architecture is characterized by a highly dynamic structure where both the components and their interconnections may change over time. To accommodate the required level of adaptivity and evolvability, the platform architecture should be able to accommodate continuous structural change without aversely affecting the platform. Indeed, the platform architecture is required to be "fluid" and able to accommodate continuous architectural changes [3]. To this end, enabling properties are: (*i*) *loose coupling*, components are deployed and executed independently of other entities, (*ii*) *flexibility*, components can be added and removed into the running application, (*iii*) *dynamism*, components of interests are discovered and bound into the running application, and (*iv*) *serendipity*, unforeseen components are accommodated into the running application.

3 Future work

We conjecture that the higher degree of specialization with a shorter time-to-deliver of new products and services to the market impacts organizational capabilities directly and platform architectures indirectly. Enterprises must become more resilient to forces developing strategies for sustainable business, enterprise, and platform architectures.

First, we must better understand how to bring business architecture closer to platform architecture. This is the first step towards a tighter innovation-cycle. Second, we develop knowledge for how to systematically open up platforms to invite third-parties to develop products and services for market niches. The third capability is concerned with an enterprise's ability to remove the distinct border in a platform's or product's life-cycles. Technologies supporting online and offline evolution and adaptation will be combined to tighten the innovation cycle even more. This poses several challenges to organizational capabilities that must be better understood, including processes for development activities and design principles for platform architectures.

References

1. E. Allman. The robustness principle reconsidered. *Commun. ACM*, 54(8):40–45, Aug. 2011.
2. G. Baxter and I. Sommerville. Socio-technical systems: From design methods to systems engineering. *Interact. Comput.*, 23(1):4–17, Jan. 2011.

3. M. Caporuscio and C. Ghezzi. Engineering future internet applications: The prime approach. *Journal of Systems and Software*, 106:9–27, 2015.
4. J. O. Coplien and N. B. Harrison. *Organizational Patterns of Agile Software Development*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.
5. R. C. Martin. *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2003.
6. H. McManus and D. Hastings. A framework for understanding uncertainty and its mitigation and exploitation in complex systems. *Engineering Management Review, IEEE*, 34(3):81–81, Third 2006.
7. W. Miller and L. Morris. *Fourth Generation R&D: Managing Knowledge, Technology, and Innovation*. Wiley, 1999.
8. A. Osterwalder and Y. Pigneur. *Business model generation : a handbook for visionaries, game changers, and challengers*. John Wiley & Sons, Hoboken (N.J.), 2010.
9. J. W. Ross, P. Weill, and D. Robertson. *Enterprise architecture as strategy: Creating a foundation for business execution*. Harvard Business Press, 2006.
10. P. P. Tallon and A. Pinsonneault. Competing perspectives on the link between strategic information technology alignment and organizational agility: insights from a mediation model. *Mis Quarterly*, 35(2):463–486, 2011.
11. D. Weyns, M. Caporuscio, B. Vogel, and A. Kurti. Design for sustainability = runtime adaptation u evolution. In *Proceedings of the 1st International Workshop on Sustainable Architecture: Global collaboration, Requirements, Analysis (SAGRA)*, pages 62:1–62:7, Sep 2015.

Self-Learning, Self-Actuation and Decentralized Control: How Emergent System Capabilities Change Software Development

Helena Holmström Olsson[1] and Jan Bosch[2]

[1] Malmö University, Faculty of Technology and Society, Nordenskiöldsgatan 1, 211 19 Malmö, Sweden
helena.holmstrom.olsson@mah.se
[2] Chalmers University of Technology, Department of Computer Science & Engineering, Hörselgången 11, 412 96 Göteborg, Sweden
jan.bosch@chalmers.se

Abstract. With recent and rapid advances in areas such as online games, embedded systems and Internet of Things, the traditional notion of what constitutes a system is fundamentally changing. Similarly to Systems of Systems (SoS) these systems are heterogeneous, autonomous and allow dynamic and emergent configurations that evolve and adjust over time. Also, these systems allow automated optimization of system performance. Regarded as the new digital business paradigm, these types of systems offer fundamentally new ways for software development companies in their service- and value creation. At the same time, they present challenges in these organizations. In this paper, and based on multiple case study research in three different domains, we identify emergent system characteristics that pose new challenges on software development and we outline the transition towards new ways-of-working in software development.

Keywords: Online games, embedded systems, Internet of Things, self-learning systems, self-actuation, decentralized control.

1 Introduction

With recent advances in software technology, we are experiencing a fundamental shift in how people interact with software-intense systems and what is expected from these systems. In different domains, new types of systems are emerging with characteristics that make them very different from the systems we are used to and that software development organizations have traditionally developed. As one example, Internet of Things systems offer fundamentally new opportunities for value creation, and are rapidly permeating our everyday lives by transforming the way we interact and perceive information technology [1]. Similar to Systems of Systems (SoS), these systems incorporate functions such as e.g. sensing, actuation and control, and they are heterogeneous, autonomous and often distributed. Moreover, they use advanced data

collection and analysis mechanisms to initiate actions and to make decisions in a predictive or adaptive manner [2, 3], and they allow devices in the network to collaborate through dynamic configurations that evolve over time. As a result, these systems foster new user behaviors and allow new forms of interaction, they enable new service- and value creation and they allow innovative business models and opportunities.

However, while new types of systems offer a wide range of opportunities, they also pose significant challenges to the organizations developing these. In this paper, and based on multiple case study research in three different domains, we identify emergent system characteristics that pose new challenges on software development and we outline the transition towards new ways-of-working in software development.

2 Background: Emergent System Capabilities

Due to rapid advances in technology, new types of systems are emerging with characteristics and capabilities that we didn't experience up until now. As a consequence, the ways in which software systems are developed, and the ways in which development organizations and teams traditionally work, are being disrupted. These emergent systems characteristics are: (1) self-learning systems, (2) self-actuation systems, and (3) decentralized control.

First, ***self-learning systems*** refer to adaptive systems whose operation algorithm improves based on trial and error. This self-learning characteristic allows systems to experiment with different behaviors and learn from these experiments to more rapidly adjust their behaviors according to e.g. user preferences. Second, ***self-actuation systems*** refer to systems that actively initiate actions based on input from the environment in which they operate [4]. As a result, they require less user interaction the more they learn about the user. Finally, ***decentralized control*** refers to systems in which each master in the network has all data. This supports local decision-making and allows for rapid actions to be taken in the network.

3 Research methodology

The research presented in this paper builds on longitudinal multi case study research [5] and close collaboration with software development companies in the online gaming, the embedded systems and the Internet of Things domain. In all the companies we studied, new system capabilities are emerging, and as a result the companies face new and exciting ways for service- and value creation, at the same time as they face a number of challenges in relation to the development of these systems. In each company, we conducted interview studies, group interviews workshops, observations and validation sessions with people representing the software development teams, the release organization, project and product management and sales and marketing.

4 Findings

As a result of the advancements in technology, the case companies involved in our study experience a number of challenges. These challenges relate to their current software development practices, as these will need to change in order to cater for development of systems with new system characteristics. In Table 1, we summarize the challenges we identified in the case companies. We summarize the challenges in relation to (1) R&D process, (2) Data collection and use, and (3) Business and organization.

Area of concern:	Challenges:
R&D process	• The transition from early specification of requirements towards emerging system characteristics. • The transition standardized systems towards dynamic systems that continuously evolve. • The transition from long-term planning and pre-defined milestones towards continuous experimentation. • The design of architectures to manage control and access in decentralized systems consisting of interconnected objects and devices.
Data collection and use	• The collection, analysis and visualization of data from multiple devices. • The collection of real-time data for dynamic optimization of systems. • The collection of data revealing user behaviors not only in relation to one system, but in relation to a larger system of which an individual device is only one part.
Business and organization	• The alignment of data collection practices and decision-making and prioritization processes. • The interplay between human development teams and smart, automated systems.

Table 1. Summary of the challenges we identified in the case companies.

5 Towards New Ways-of-Working in Software Development

In Table 2, we picture the transition from traditional towards continuous and experimental software development, with agile development and data-driven development being the intermediate steps. We define the main differences for each development approach in relation to (1) R&D process, (2) Data collection and use, and (3) Business and organization.

Characteristics	Traditional development	Agile development	Data-driven development	Continuous and experimental development
R&D process	• Long cycles with defined milestones • Waterfall and sequential	• Rapid cycles with short development sprints • Iterative and	• Frequent and automated experimentation.	• Automatic experimentation by systems in the field

		incremental	• Data analytics • Dynamic and evolutionary	• Systems exploring and optimizing alternatives	
Data collection and use	• Pre-study • Requirements specifications	• Customer collaboration • Product owners and/or customer-specific teams as proxy	• Data collection from systems in the field • Continuous validation with customers	• Automated experimentation. • Embedded analytics in systems in the field	
Business and organization	• Discipline oriented organization • Technology-driven innovation	• Cross-functional R&D teams • Customer-driven innovation	• Integrated R&D, PdM and data analytics team • Data-driven innovation	• R&D teams and smart systems • Synergy-driven innovation	

Table 3. Towards continuous and experimental software development.

6 Conclusion

In this paper, and based on multiple case study research in three different domains, we identify emergent system characteristics that, in similar with Systems of Systems (SoS) allow distributed devices to communicate, collaborate and take proactive decisions. As new characteristics of online games, embedded systems and IoT systems, these characteristics pose new challenges on software development. We identify these challenges and we present the transition from traditional development towards continuous and experimental software development.

References

1. Levy, H. What's New in Gartner's Hype Cycle for Emerging Technologies, 2015, http://www.gartner.com/smarterwithgartner/whats-new-in-gartners-hype-cycle-for-emerging-technologies-2015/.
2. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): A vision, architectural elements, and future directions. Future Gener. Comput. Syst. 29, 1645–1660 (2013).
3. Kinsner, W. 2007. Challenges in the Design of Adaptive, Intelligent and Cognitive Systems. In Proceedings of the 6th IEEE International Conference on Cognitive Informatics, August 6-8, Lake Tahoo, CA, pp. 13-25.
4. Peters, G. 2015. Six Necessary Qualities Of Self-Learning Systems: A Short Brainstorming. In Proceedings of the International Conference of Neural Computation Theory and Applications, pp. 358-364.
5. Yin, R.K.: Case study research: design and methods. Sage Publications, Los Angeles, Calif. (2009).

IoT-based Systems of Systems

Fahed Alkhabbas, Romina Spalazzese and Paul Davidsson

Department of Computer Science
Internet of Things and People Research Center
Malmö University, Malmö (Sweden)
fahad.khabbass@gmail.com, {romina.spalazzese, paul.davidsson}@mah.se

Abstract. *Systems of Systems* (SoS) and the *Internet of Things* (IoT) have many common characteristics. For example, their constituents are heterogeneous, autonomous and often distributed. Moreover, both IoT and SoS achieve intended goals by means of the highly dynamic cooperation among their constituents. In this paper we study the relation between IoT and SoS. We discuss the characteristics of both concepts and highlight common aspects. Furthermore, we introduce the concept System of Emergent Configurations (*SoECs*) to describe IoT-based SoS.

Keywords: System of Systems (SoS), Internet of Things (IoT), Emergent Configurations (ECs), System of Emergent Configurations (SoECs)

1 Introduction

The Internet of Things (IoT) enables heterogeneous and distributed smart objects to connect, communicate and collaborate to achieve common goals [4]. Such objects involve, e.g., sensors, actuators, computer-based systems, smart-phones and other smart physical objects. The number of devices connecting to the Internet is exponentially increasing and is expected to reach 24 billions by 2020 [6]. Thus, the IoT is expected to emerge in almost every aspect of the daily life.

The concept System of Systems (SoS) involves the dynamic collaboration of distributed and heterogeneous systems to achieve common goals. The evolution of integration and communication technologies enabled legacy, existing and new systems to collaborate and realize the concept SoS. The (SoS) term is widely used in the literature and several efforts have been done to define the concept. However, due to the wide range of applicable definitions stemming from various research backgrounds and perspectives, common characteristics are defined to better describe the concept.

In this paper, in Section 2, we review the characteristics of IoT and SoS domains and highlight the shared aspects between them. In addition, we analyze the concept of Emergent Configurations in the context of SoS. Moreover, a case study is presented to explain IoT-based SoS. Section 3 concludes the paper.

2 The Internet of Things as System of Systems

This section illustrates the relationship between the Internet of Things and the System of Systems concepts. The characteristics of both domains are identified and compared in order to clarify the correlations between the two concepts.

2.1 A review of SoS and IoT Characteristics

IoT characteristics. This part presents several characteristics of the IoT [5]: (i) *Devices heterogeneity*: IoT involves a variety of connected heterogeneous objects of which some are smart and can make decisions autonomously. (ii) *Distributed components*: huge number of smart objects join IoT -thus the IoT is constantly *evolving*. (iii) *Ubiquitous data exchange capabilities*: smart objects and computational units exchange tremendous amount of information. (iv) *Localization and tracking capabilities*: IoT objects are uniquely identified and traceable. (v) *Self-organizing capabilities*: IoT components may have self-adaptable capabilities due to dynamic IoT environments. (vi) *Semantic interoperability and data management*: intelligent data management techniques are required to analyse exchanged data among IoT components. Semantic interoperability requires designing well-defined and standard semantic models.

SoS characteristics. Maier defined five characteristics of SoS [7]: (i) *Operational Independence*: SoS constituents are autonomous and can keep achieving their goals even if detached from the SoS. (ii) *Managerial Independence*: SoS constituents are self-controlled and managed. (iii) *Geographical Distribution*: SoS constituents might be distributed in several locations. (iv) *Evolutionary Development*: SoS evolves as constituents evolve continuously. (v) *Emergent Behaviour*: as dynamic and heterogeneous constituents interact, new behaviours are introduced to the SoS level.

Maier considers systems which possess operational and managerial independence to be SoS regardless of the complexity and geographical distribution of relevant constituents [8]. Boardman and Sauser add three more characteristics [1]: (i) *Belonging*: achieving overall goals is attributed to SoS and not to any individual constituent. (ii) *Connectivity*: constituents can exchange messages and information. (iii) *Diversity*: constituents are heterogeneous, evolving and *interoperable*. Another important characteristic is *interdependence* [2]. Interdependent constituents rely on each other to satisfy common goals.

2.2 Discussion

It is clear that IoT and SoS share some major characteristics. For instance, like SoS, IoT components are heterogeneous, autonomous, able to communicate, often distributed, operational and managerial independent. Both domains are evolving and operate in dynamic situations leading to emergent behaviours. In this context, the Internet of Things can be regarded as a System of IoT-based systems.

In the context of IoT, we refer to the term Emergent Configurations (ECs) of Connected Systems as "*a set of things with their functionalities and services that connect and cooperate temporarily to achieve a goal*" [3].

Within SoS, the emergent behaviour concept has been defined as "*the behaviours that arise as a result of the synergistic collaboration of constituents*" [2]. Functional and non-functional emergent properties result from having heterogeneous and distributed constituents collaborating in dynamic environments. Emergent behaviours are desirable in case they contribute positively to SoS goals. On the contrary, they are undesirable in case they adversely affect achieving goals. Systems monitor, detect, analyze, and self-adapt to undesirable emergent behaviours in order to provide consistent functionalities. Self-adaptation mechanisms are out of the scope of this work.

Considering the EC concept, the IoT can be seen as a collection of ECs. In other words, we see an IoT system as an EC. Since constituents of an IoT-based SoS are IoT systems, an IoT-based SoS is referred as a *System of Emergent Configurations* (SoECs). The section below introduces the smart street lamppost case, and illustrates how this IoT-based System of Systems is considered as a SoECs.

2.3 Smart Street Lamppost System

In this section, we take the case[1] of the *Smart Street Lamppost System* (basic system) described in [3] and extend it (scenario) to illustrate the concept of SoECs discussed above.

Basic System. The main idea, in this case, is to adapt a smart lampposts sphere of light for each road user. This means that the number of lampposts that light up depends on the speed of vehicles, bikes or pedestrian. Yellow lights are dimmed down, to save energy, when there is no traffic. Lampposts turn on red lights when drivers go over speed limits -this contributes to increase safety and traffic awareness. Each lamppost has the following capabilities: (i) Detects the presence of vehicles, bikes or pedestrian through motion detection sensors. (ii) Measures the speed of moving objects through a computation unit. (iii) Decides the brightness of its light, using either yellow or red lights through a pair of actuators. (iv) Computes the number of neighbour lampposts that should light up (yellow or red lights). (v) Exchanges messages with neighbour lampposts. Lampposts are grouped into areas and an Area Reference Unit (ARU) manages each lampposts group. ARUs have powerful computational capabilities and storage capacities. The collaboration of the above components is an Emergent Configuration. The main emergent behaviour of the case is to switch on red lights when car B exceeds the speed limit. An emergent property is to warn car A about the speed of car B.

Scenario. To extend the work done on the case above we introduce the scenario below. (i) Lampposts collaborate with ARUs to detect accidents through

[1] This case has been realized, including the hardware and software, in a collaboration between Malmö University, Internet of Things and People (IoTaP) Research Center, and Sigma Technology under the ECOS project [9]

the GPS technology, see [10]. (ii) An ARU reports an accident to the relevant Health Management System including the location and number of cars involved. (iii) The driver of car B suffers from serious heart disorders, so she/he puts on wearable health monitoring sensors regularly. These sensors report her/his medical status to the HMS *(emergent configuration)*. (iv) The HMS specifies the number of ambulances needed and redirect the closest ones. Meanwhile, a specialist in the operations room analyses the data being received from the driver's health monitoring sensors.

As illustrated in the above scenario, the two emergent configurations (systems) collaborated together composing a System of IoT-based Systems.

3 Conclusion

In this paper we analyzed the characteristics of the Internet of Things and System of Systems domains. It can be noted that the domains share the core characteristics like operational and managerial independence, evolving systems and emerging behaviors/properties. Moreover, we look at SoS from the perspective of IoT emergent configurations (ECs). In this context IoT-based SoS can be seen as a System of Emergent Configurations (SoECs). We presented the Smart Street Light case, and extended it to illustrate temporal collaboration between two emergent configurations composing an IoT-based System of Systems.

References

1. J. Boardman and B. Sauser. System of systems—the meaning of "of.". *IEEE/SMC International Conference on System of Systems Engineering*, 2006.
2. J. Fitzgerald J. Woodcock C. Nielsenand, P. Larsen and J. Peleska. *Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions.*
3. Federico Ciccozzi and Romina Spalazzese. Mde4iot: Supporting the iot with mde. In *10th International Symposium on Intelligent Distributed Computing.*
4. G. Morabito L. Atzori (Eds.) D. Giusto, A. Iera. *The Internet of Things.* Springer, 2010.
5. F. Pellegrini D. Miorandi and I. Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, page 1497–1516, 2012.
6. S. Marusica M. Palaniswamia J. Gubbia, R. Buyyab. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29:1645–1660, 2013.
7. M. Maier. Architecting principles for systems-of-systems. *Annual International Symposium of the International Council on Systems Engineering.INCOSE*, 1:267–273, 1996.
8. M. Maier. Architecting principles for systems-of-systems. *Systems Engineering 1,4*, 1:267–284, 1998a.
9. Emergent Configurations of Connected Systems (ECOS). http://iotap.mah.se/ecos, accessed [2016-06-25].
10. Md. Amin ; M. Bhuiyan ; M. Ibne Reaz ; S.Nasir. Gps and map matching based vehicle accident detection system. *Research and Development (SCOReD), 2013 IEEE Student Conference on*, 2013.

Frameworks for innovation – studies of technology innovation and the emergence of systems of systems (SoS) for energy, defence and security

Extended abstract for a paper at a conference on SoS (SweSoS 2016) to be held in September 2016 excerpted and based on a PhD thesis entitled Frameworks for innovation – studies of technology innovation and systems for energy, defence and security

Bengt A Mölleryd, TeknLic

The paper concerns technology innovation and systems for energy, defence and security with a view and an outline of a theory for the emergence and the building of systems of systems (SoS).

Attention is focussed on radical or *disruptive innovations* induced or influenced by technology development and change. Innovations bring great values to society, for competitiveness and sustainable social wellbeing, and may contribute to solving the great challenges of our time, as for example climate change, secure supply of energy and preservation of the environment.

Disruptive innovation which encompasses *transformative processes and transitions* has two sides, one large or even immense but uncertain advancements. The other, negative side may exhibit extensive and brutal consequences. Digitisation with associated net technologies is a prime example of technologies that simultaneously accomplish brand new services and competitiveness and disruptions of systems, industrial branches and individual firms, and even sectors of society.

Innovations in the present view are considered as *technological systems that emerge by integration of systems (establishing systems of systems, SoS) in evolutionary processes*. As systems integration and emergence of SoS are key factors and pivotal for innovation a theory for systems integration and emergence of SoS for innovation is elaborated and duly explicated.

In the thesis, also referred in the paper there is a proposition for an *innovation view* aiming to suggest ways and means, and to provide guidance to promoting technological innovations or value-adding new technologies, systems, services and products in the energy, defence and security areas. The innovation view comprises an *architectural framework* and *a rudimentary model* to support *governance of technological innovation*.

The innovation view and the architectural framework for innovation are proposed on the basis of case studies of patterns of innovations in selected areas, reviewing the literature and theories of governance of innovations and innovation processes supported with application of methodologies and tools from systems engineering and enterprise architectural frameworks.

The innovation view has a main mission to provide strategic guidance to the governance of technological innovation as a complex and evolutionary process which associates and integrates systems (into SoS). The evolutionary process and transformation is inherent with risks and has to rely on experimenting and learning. The view is perceived as complement rather than replacement of recognised enterprise architectural frameworks, systems engineering standards, protocols and established procedures (e.g. ISO 15288, ISO 42010, NISP).

The innovation view is in the thesis and the paper compared and tested (a sort of dry test) with recognised architectural frameworks and configurations alleged with innovation or with strong evidences of innovative applications and outcome. It is a number of configurations, for example triple helix and platform design that are compared and discussed from the perspective of the architectural framework and innovation view which are proposed.