
Project: COMPASS

Grant Agreement: 287829

*Comprehensive Modelling for Advanced Systems of Systems*


# An Introduction to Systems of Systems: Underlying Concepts

COMPASS White Paper WP06

February 2014

Public Document

http://www.compass-research.eu

# Authors:

Claire Ingram, Alexander Romanovsky, Newcastle University, UK

# Abstract:

This white paper introduces some key concepts relevant to our work in model-based techniques to support development and maintenance of systems of systems. It is intended as a suitable starting point to introduce the key challenges facing engineers in systems of systems (SoSs), particularly when considering the application of model-based techniques. We provide pointers to other relevant COMPASS deliverables which provide further detail on some areas. This white paper makes a suitable starting point for reading about model-based techniques in systems of systems, and a good introduction to COMPASS for those who are interested in reading more about the project or more about a specific area.

# 1 Introduction

This white paper very briefly summarises some concepts and challenges that are relevant for model-based techniques in SoS engineering. It is intended as a suitable starting point to introduce the key challenges facing engineers in systems of systems (SoSs), particularly with respect to the application of model-based techniques. We provide pointers to other relevant COMPASS deliverables which provide further detail on some areas.

Specifically, in this white paper we consider concepts from the domains of:

- systems engineering
- existing SoS concepts such as classifications and properties
- architectural analysis
- fault tolerance and resilience
- formal modelling techniques
- testing and verification

We identify key concepts throughout this paper using italics.

# 2 Role of Case Studies

We provide practical examples of the concepts drawn from COMPASS'S four case studies, as follows:

- An emergency response system, provided by Insiel (described fully in COMPASS D41.1). This SoS co-ordinates different specialised services (fire brigade, ambulance service, police, etc.) situated in northern Italy to deliver cross-service responses to emergency situations.
- A distributed audio/visual system, provided by Bang & Olufsen (described fully in COMPASS D42.1). This SoS allows audio-visual content to be streamed and managed in real time by a variety of devices in line with rules stipulated by DRM.
- An energy management system, provided by GridManager (described fully in COMPASS D43.2). This SoS ensures that customers' energy requirements are fully met in line with pre-determined criteria.
- A traffic management system, provided by West Consulting (described fully in COMPASS D43.2). This SoS manages flows of traffic on an inter-urban road network.

In the following sections we examine some key SoS concepts, illustrate them with examples drawn from the real-world systems, and examine the case studies/challenge problems for similarities and differences.

# 3 Properties of Systems of Systems

Work on a survey of modelling techniques in systems of systems by COMPASS partners (Nielsen et al 2013) has identified has many properties which have been

associated with SoS in the past. This work has also condensed these properties into eight key characteristics which COMPASS particularly associates with SoS. These properties are:

- **Autonomy.** "Autonomy is the extent to which a constituent system's behaviour is governed by its own rules rather than by others external to the constituent" (Nielsen et al 2013). This characteristic is related to (but not necessarily the same as) the concept of "operational independence", proposed by Maier (1998).
- **Independence.** "Independence is the capacity of constituent systems to operate when detached from the rest of the SoS" (Nielsen et al 2013). This characteristic is related to (but not necessarily the same as) concepts such as "managerial independence", proposed by Maier (1998).
- **Distribution.** "Constituent systems are dispersed so that some form of connectivity enables communication or information sharing" (Nielsen et al 2013).
- **Evolution.** "SoSs are long-lasting and subject to change, whether in the functionality delivered, the quality of that functionality, or in the structure and composition of constituent systems" (Nielsen et al 2013).
- **Emergence of behaviour.** "Within the SoS literature, emergence refers to the behaviours that arise as a result of the synergistic collaboration of constituents. Reliance is typically placed on the delivery of some emergent behaviours in order to deliver a higher functionality than delivered by the constituents separately" (Nielsen et al 2013). Some emergent behaviour may be unexpected.
- **Dynamicity of behaviour.** "Dynamic behaviour is the capacity of an SoS to undertake changes to its structure and composition, typically without planned intervention" (Nielsen et al 2013).
- **Interdependence.** "Interdependence refers to the degree of mutual dependency between constituents… If the objective of a constituent system depends on the SoS, then the constituent system itself may have to sacrifice some of its individual behaviour in order to meet the requirements of joining SoS" (Nielsen et al 2013). This characteristic is related to (but not necessarily the same as) concepts such as: "belonging", (Boardman & Sauser 2006) and "open at the top" (Abbott 2006).
- **Interoperability.** "Interoperability refers the ability of the SoS to incorporate a range of heterogeneous constituent systems." (Nielsen et al 2013). This characteristic is related to (but not the same as) concepts such as "*diversity*" (Boardman & Sauser 2006, Baldwin & Sauser 2009) and "open at the bottom", proposed by Abbott (2006).

# 4 Classification of Systems of Systems

We differentiate between *virtual*, *collaborative*, *acknowledged* and *directed* SoS, in line with the DoD (Department of Defense Office of the Deputy Under Secretary of Defense for Acquisition and Technology 2008). The COMPASS project also introduces a new classification, *hostile* (COMPASS D21.2)

- **Directed system of systems.** An SoS built and managed to fulfil specific goals. Although the constituents can operate independently, within the SoS they accept some central management to ensure that SoS-level goals are met (Maier 1998).
- **Virtual system of systems.** Virtual SoS lack a central management authority and a centrally agreed-upon SoS-level goal. Large-scale behaviour emerges and may be desirable - but there is no visible active management of the SoS or its goals (Maier 1998).
- **Acknowledged system of systems.** "Acknowledged SoS have recognized objectives, a designated manager, and resources for the SoS, however, the constituent systems retain their independent ownership, objectives, funding, as well as development and sustainment approaches. Changes in the systems are based on collaboration between the SoS and the system" (Dahmann & Baldwin, 2008).
- **Collaborative system of systems.** The SoS has no coercive power over the constituent systems, but they voluntarily choose to collaborate in order to achieve the SoS goals (Maier 1998).
- **Hostile system of systems.** "The SoS has no coercive power over the CS and they do not voluntarily choose to collaborate in a given SoS to achieve the SoS goals" (COMPASS D21.1).

# 5 Concepts from Systems Engineering

Many concepts from the field of systems engineering are naturally also relevant to SoS engineering (SoSE). This area is developed further in (COMPASS D21.3). Systems engineering concepts face added complexities when applied in SoS because of:

- The greater size and *complexity* of the solution space, which can present a combinatorial explosion problem that prevents, e.g., exhaustive testing (this is not unique to SoSs).
- Socio-technical aspects of the SoS.
- *Continous evolution*, which may or may not be synchronised between constituents. There may be little warning about updates to a constituent system which is a "black box", which is a reluctant participant or which is unaware of the SoS. This greatly complicates the activities associated with a development lifecycle, because verification and testing, requirements engineering, implementation, deployment, and the maintenance process itself are all capable of being interrupted with new changes.
- Extremely long *lifecycles*. In some cases, constituent systems which are no longer supported with updates are required to interoperate with brand-new technologies that are designed for a modern architecture. The lifecycle of an SoS is generally expected to be complex, and, unlike the "upfront" analysis of a single system, continuous analysis addresses the dynamic nature of the SoS (Dahmann et al 2011).

- Under-specified requirements and a reliance on an iterative methodology can delay testing, since there will not be precise specifications for developing *test plans* until a relatively late stage.
- *Emergent* functionality. Accurately predicting all possible emergent behaviours is often not possible due to lack of information disclosure.
- The requirements process is complicated by the presence of requirements for each constituent system, and additionally requirements for the SoS as a whole. Any subset of these may conflict with any other subset. Where available, they are likely to represent different and mismatched levels of abstraction. There may be no one actively taking ownership of the SoS-level requirements.
- Setting the *boundary* to separate the *system of interest* from its *environment* is not necessarily straightforward, because the SoS tolerates continued *evolution* and *independence* of its *constituent systems*, meaning that elements of the environment may be considered constituent systems of the SoS or vice versa. For example, when considering an emergency response SoS, the boundary may be set differently depending on the point of view of the current model:
    - From the point of view of the user, all emergency services are logically included in the SoS. This is because the user contacts a central phone number, and the correct respondent then arrives at the site of the emergency. However, from an internal point of view, the dispatch handlers may have differing relationships with the separate emergency services themselves, and they may use different means to share data.
    - Should communication media (radios, mobile phones) be included in the SoS? They are third party "black boxes", about which we know little, and therefore behave like an *environment.* Alternatively, we may choose to reason them about as constituents because the SoS places considerable reliance on them.

Evolution may refer to changes in SoS requirements, or the current number of constituent systems and the architecture that allows them to co-operate. For example, the selection of constituent systems in a single traffic management system (TMS) has changed considerably as new technologies become available. SoS requirements have also evolved considerably; early requirements for a TMS centred on dealing with local issues when they arise, but a modern TMS is expected to take a proactive approach and actively plan for traffic management.

In some cases SoSs are required to accept changes in the set of constituent systems as a normal element of the SoS's behaviour. For example, the audio-visual SoS supplied by Bang & Olufsen needs to cope with constituents being added or removed arbitrarily by the user. In this case, the architecture is selected to support this activity and adding a new constituent system is part of normal operating behaviour, not evolution. In an SoS like this, evolution can still arise if the underlying architecture changes, if the requirement that the SoS cope with new constituents is changed, or the SoS is extended to support new types of constituents that were not previously supported.

# 6 Concepts relevant for SoS Architecture

There are some specific behaviours associated with SoS that need to be addressed when considering the SoS *architecture*, including:

- Continuous *evolution*, typically associated with SoSs. Architectures therefore need to be adaptable and, as far as possible, ensure that changes made in one constituent system are not propagated onto others unnecessarily. Techniques for ensuring a modular design are helpful here, possibly including an appropriate *architectural style*, and design techniques such as "design by *contract"* and precisely specified *interfaces.* Developing and analysing an architectural model can help with this. Maier has emphasised "leverage at the *interfaces*" as a key architectural principle for SoSE (Maier 1998).
- Lack of trust between constituent systems, which may have commercial reasons not to make full disclosures. This can constrain design choices and lead to unexpected or undesirable *emergent* behaviour.
- Many constituent systems may be unaware that they participate in an SoS, or perhaps are simply unaware of the SoS's current goal (e.g., legacy constituent systems). The architectural assumptions of such a constituent may be unavailable, and it may be correspondingly difficult to reason reliably about its position in the architectural design. The optimal design for the SoS as a whole may be unachievable if it does not suit one of the constituents and that constituent is unable or unwilling to make adaptations.
- An emphasis on re-use. As in the related field of component-based software engineering, the SoS engineering process for an SoS must include consideration at design time of the capabilities supplied by legacy or existing systems.
- Extremely long SoS-level *lifecycles*. Lifecycles of the SoS or its constituent systems are unlikely to be synchronised. This can make it very difficult to achieve a comprehensible, optimised architectural design.
- A demanding operational environment, featuring *distribution* and therefore potential connectivity problems, unreliable availability and constant change. SoS domains often demand high availability. *Dynamic reconfiguration* or similar techniques may be necessary so that the SoS can adapt to internal or external changes. This requires an ability to monitor current behaviour, and select appropriate responses (Kephart 2005). For reconfiguration, constituent systems must exhibit *substitutability*: if one system is removed, another which offers at least the same level of service can be employed instead. It must be possible to determine what constitutes "at least the same level of service" and also whether the substitution will bring any additional undesirable behaviour.
- "Fuzzy" *boundaries* which may be difficult to define. Where the boundary is placed may change depending on the current goal of the modeller and/or the viewpoint of the actors/*stakeholders.*
- Without a central authority for designing and decision-making, there is no one constituent who controls the definition of the system's external *interfaces*. If interface standards are missing or conflicting it will be difficult to manage data exchanges across systems (INCOSE 2011). Therefore *interoperability* considerations are a key challenge for the SoS architect - ensuring the systems

are able to interoperate and will continue to do so. *Architectural frameworks* are important techniques that allow heterogeneous constituents to interoperate, despite the continuous evolution, long lifecycles and devolved control.

- Constituent systems are drawn from varying *domains*, written in different languages and with different expectations of synchronicity. *Architectural Description Languages* (ADLs) for SoSs need to support descriptions provided in different languages (different domains naturally tend to have different ADLs).
- *Architectural mismatch*, when constituent systems make incorrect assumptions about the overall system *architectural style*. For example, a constituent may assume that the system operates as a pipe-and-filter, which could be problematic if the SoS actually does not have a linear process.
- The *complexity* of the system increases non-linearly as elements are added (INCOSE 2011), making it more difficult to design or model an appropriate architecture. Well understood *architectural patterns* appropriate for SoS could present a possible means of resolving complexity or lack of disclosure. An initial survey of architectural patterns for SoS is presented in COMPASS D22.3. Choice of *architectural pattern* may be affected by concerns such as development schedule and availability of existing constituent systems or COTS.

In COMPASS we characterise constituent systems as *black boxes, white boxes* or *grey boxes* (COMPASS D21.2):

- A constituent is considered as a "white box" if we know a great deal about its internal services and its performance guarantees. A white box constituent system can be adapted to suit the SoS. For example, in our emergency response SoS, constituent systems are public sector departments, and their internal organisation and service guarantees are available to their collaborators.
- A "black box" constituent allows us to make no assumptions about the internal services and we cannot expect to make any changes to adapt the system to the needs of the SoS. For example, the Bang & Olufsen audio-visual SoS case study may feature interoperating constituent systems manufactured by direct competitors.
- A grey box constituent system allows dynamic installation of SoS applications, which function as integration code between the greybox CS and the SoS. For example, an Android based constituent system may allow loading of Android applications.

The Bang & Olufsen audio-visual SoS has a service oriented architecture (SOA) that allows for more connections between constituents than a centralised architecture. This type of architecture potentially increases possibilities for *dynamic reconfiguration*, since constituents can interact directly. The possibilities for unpredicted *emergent* behaviour may also theoretically increase, since, potentially, any constituent can consume the services of any other constituent as long as it can meet the terms of the interface contract.

A centralised (or partially centralised) architectural pattern may be an appropriate choice for an SoS if it does not place a high value on dynamic reconfiguration. For example, for the emergency response SoS, dynamic reconfiguration is undesirable, as it is likely to result in confusion over command chains on the ground and a lack of accurate record keeping. Employing a centralised architecture certainly does not preclude unexpected emergence or dynamic reconfiguration, but it does restrict the potential number of interactions between constituent systems. Architectural patterns for SoS are discussed in more detail in (COMPASS D22.3).

# 7 Concepts relevant for SoS Modelling

We present an initial description of challenges for model-based techniques in SoSE:

- Some *emergent behaviour* could be unexpected or even undesirable. It may be impossible to predict accurately all emergent behaviours due to lack of disclosure, which may also hinder representation of internal and external interfaces or the creation of a model showing the SoS from the point of view of one *constituent system*. A notation must be able to express desired behaviours to represent *emergence*. A minimal behaviour for constituent systems needs to be agreed as part of the SoS.
- *Verification* of SoS-level properties while individual systems are diverse and *autonomous* can be difficult, because:
    - the SoS relies on emergent behaviour which may be difficult to predict
    - the constituent components are themselves large and *complex*
    - constituents may leave or join the SoS dynamically
- The desired SoS-level capabilities or requirements are frequently *under-specified* initially, which delays and complicates processes such as *verification* and testing.
- Modelling *concurrency* and *synchronisation* across organisational boundaries adds to the *complexity*.
- Constituent components are *heterogeneous*, drawn from different *domains*, described in different *architectural description languages* or notations, designed for different *architectural styles* and deployment platforms, and have different expectations of synchronicity, data format and *dependability requirements*. *Underspecification* is a common technique which may be helpful here, and *abstraction* can be employed to address concrete problems in the modelling of an SoS, like constituent systems joining or leaving the system.
- Many constituent systems may be Commercial Off-the-Shelf (COTS) systems or legacy systems with little in the way of formal documentation available. A solution is to wrap a formal *specification* around a constituent and to implement *fault tolerant* techniques based on the wrapper.
- The modeller needs to be able to focus on individual problems for *verification*, but retain confidence that the results can be safe when considered along with the rest of the SoS and its *environment* (this is not limited to SoSs) and so SoS modelling notations must support *composition*.
- Lacking a central authority, many SoSs see no single organisation with oversight of all external *interfaces* of the SoS. Continuous *evolution* means

that elements (e.g., external or internal interfaces) may change without warning. *Traceability* of *stakeholders* and *requirements* is important. Modelling and analysing interfaces beforehand helps to reduce the negative consequences of unexpected changes. *Serialization*, *reflection* and *higher order logic* are concepts for combatting this problem.

- An SoS is *distributed*, meaning that constituent systems may experience unreliable connectivity and availability. *Dynamic reconfiguration* techniques (discussed in Section 6) can be important, allowing the SoS to adapt.
- Human factors that perform functions within the SoS need to be accounted for; concepts from *human factors engineering* may be relevant here. Human behaviour may require some *stochastic verification*.

Note that issues in modelling SoSs are surveyed thoroughly in (Nielsen et al 2013).

As part of the COMPASS project, we have developed a new notation, CML, that incorporates elements of state as well as concurrency, synchronisation and timing. CML is the first language to be designed specifically for the modelling and analysis of SoSs. It is a rich notation that permits many kinds of analysis to be conducted. Discussions and examples of interfaces in CML are presented in (Bryans et al 2013). CML is defined more fully in (COMPASS D23.4a).

# 8 Dependability and Fault Tolerance Concepts

The particular characteristics of systems of systems can present challenges in the field of SoS *dependability*: "SoS, because of their *complexity*, the *dynamic* state of their design characteristics and their *emergent behaviour* are particularly vulnerable to catastrophic *failure*." (Valerdi et al, 2008). COMPASS has developed an architectural approach to reasoning about fault tolerance, which is described in (Andrews et al 2013) and in (COMPASS D24.2). Some key issues relevant for dependability in SoS engineering include:

- *Interdependence* is a major issue for dependability. Within the SoS, dependencies may exist between two independently-owned constituent systems. This has implications when deriving global properties from properties of individual constituent systems, particularly when constituents are unaware of the SoS and make no guarantees as to their availability.
- SoS-level functionality is often crosses organisational boundaries. It is not always clear where responsibility lies for ensuring reliable delivery of SoS-level functions. Governance is an issue: who is financially responsible for any *failures*? *Traceability* of *stakeholders* and *requirements* can be useful for governance.
- *Emergent behaviour* may vary from what is expected and constituents may make erroneous assumptions that certain functions are provided (or definitely not provided) by their peers. *Independent evolution* of constituents means an update to one constituent may unexpectedly invalidate assumptions made by other constituents. Unexpected functionalities that are only apparent at runtime may invalidate assumptions made during analysis.

- An SoS is susceptible to problems in quality of service – for example, as a result of its *distributed* nature or because of changes in a volatile *environment*. *Dynamicity* is often required, which complicates attempts to reason about dependability and *verification* in advance. One approach is to reason about possible configurations and/or negotiate and specify in advance the minimum requirements that a reconfigured version of the system must meet, using a technique such as a *reconfiguration policy language.*
- An SoS model typically has "fuzzy" *boundaries*. An SoS model should aid analysis of the extent to which an SoS is vulnerable to *environmental* changes.
- Evidence that can be gathered - e.g., for a safety case - may be limited, especially about individual constituent system functionality, because constituents may not be able to make full disclosures. In addition, different evidence may be needed for different *domains*, and across the SoS there may be mixed *criticality*. An SoS modelling system must facilitate the compilation of evidence with clear *provenance* to facilitate reasoning about *vulnerabilities*.
- Humans often provide the "glue" that enables an SoS to *interoperate*, or interact in complex ways with the SoS's emergent behaviour. Many of the vulnerabilities in an SoS may be due to humans and so an SoS model should be capable of modelling a wide range of behaviour. Concepts from *human factors engineering* may be useful here. *Stochastic modelling* may be necessary.
- A modelling method for representing a *fault tolerant* system which justifies a certain level of *trust* should include some way of representing recovery and *resilience*.
- The *classification* of an SoS has implications for dependability (e.g., whether an SoS may be classified as *virtual* or *collaborative* impacts on the requirements for conformity, documentation and evidence that can be demanded of constituents).
- Constituent systems are likely to define a *transaction* in terms of their own particular functions, and not in terms of a collection of functions across the SoS. Transactions represent a traditional technique for managing dependability in a *distributed* setting; a transaction guarantees *atomicity*, *consistency*, *isolation* and *durability*.
- The selected *architectural pattern* may affect fault tolerance. In a centralised architecture, the central “hub” or “manager” constituent marshals the services offered by other components and its goals are usually coincident with SoS goals. In a decentralised SoS, such as the audio-visual SoS, global requirements are composed from the available functionalities of the constituent systems. The constituents may not be aware of the SoS, which can make it more difficult to reason about reliable performance. Some architectural patterns permit any constituent to communicate with any other, and in many SoSs new constituents can be introduced at any time. In this case a constituent cannot guarantee that it is communicating with a non-malicious device. This risk is somewhat alleviated in a centralised architecture, in which all or most constituents restrict their communications to a single trusted central manager.

COMPASS employs definitions of fault-tolerant terms described in (Avizienis et al., 2004). Some of these terms are re-defined for SoS. A system-level *failure* from the point of view of a constituent system (i.e., a constituent system deviates from delivering its expected behaviour) may in turn become a component-level *fault* from the point of view of the SoS.

With a large number of constituents which may have overlapping functionality, there is often considerable possibility for *resilience* to be built into an SoS. SoSs are often a solution to large and complex safety critical or mission critical functions, and therefore resilience is an important *non-functional requirement.* For example, both the traffic management challenge problem and the emergency response examples are safety critical and demand a high degree of resilience. The GridManager challenge problem presents an example which is not safety critical, but where substantial losses may be incurred if the SoS fails to deliver on its requirements.

Resilience can be achieved through *dynamic reconfiguration* but does not have to be. For example, if some part of the traffic monitoring system becomes unavailable it can cope relatively seamlessly in a variety of ways, such as substituting predicted data based on data from adjacent locales if a traffic monitoring device ceases to provide data. Redundancy can be employed in monitoring devices to provide some error-checking (e.g., sensors may be deployed in groups before and after junctions to provide sets of data that can be cross-checked).

Dependability requirements may be one criteria useful for helping to determine where the boundaries of the system should lie. For example, it's not completely clear where the *boundary* of the emergency response SoS should be located. One criteria which can be used to determine boundary location is the level of dependability that is required. For example, because there is a high degree of reliance on the radio system and it is a known possible point of failure, we include it as a black box constituent system to analyse impact of radio failure, and possible recovery strategies.

# 9 Conclusions

In this white paper we have outlined very briefly some key concepts and challenges that affect attempts to employ model-based techniques in SoS, drawing on our four case studies to provide some illustrative examples. Specifically, we consider concepts from the domains of systems engineering, architectural analysis, fault tolerance and resilience, formal modelling techniques and testing and verification. The COMPASS project seeks to develop tools and techniques to tackle many of these challenges, which will continue to appear in future phases of the project.

# References

Russ Abbott. Open at the top; open at the bottom; and continually (but slowly) evolving. In System of Systems Engineering, 2006 IEEE/SMC International Conference on. IEEE, April 2006.

Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing, 1:11–33, 2004.

Andrews, Z.; Fitzgerald, J.; Payne, R. and Romanovsky, A. 2013. "Fault Modelling for Systems of Systems" in Proceedings of the 11th International Symposium on Autonomous Decentralised Systems (ISADS 2013), pp. 59--66.

W.C. Baldwin and B. Sauser. Modeling the Characteristics of System of Systems. In System of Systems Engineering,(SoSE 2009). IEEE International Conference on. IEEE, 2009.

John Boardman and Brian Sauser. System of Systems – the meaning of "of". In Proceedings of the 2006 IEEE/SMC International Conference on System of Systems Engineering, Los Angeles, CA, April 2006. IEEE.

Bryans, J., Payne, R., Holt, J. and Perry, S. "Semi-Formal and Formal Interface Specification for System of Systems Architecture". In IEEE Systems Conference, Orlando, FL(USA). 2013.

COMPASS D21.1. "Initial Report on Guidelines for Architectural Level SoS Modelling". Document Number: D21.1. Technical report [Online] http://www.compass-research.eu, 2013.

COMPASS D21.3. "Initial Report on Guidelines for Systems Engineering for SoS". Document Number: D21.3. Technical report [Online] http://www.compass-research.eu, 2013.

COMPASS D22.3. "Report on Modelling Patterns for SoS Architectures". Document Number: D22.3. Technical report [Online] http://www.compass-research.eu, 2012

COMPASS D23.4a. "CML Definition 3 – Denotational Semantics". Document Number: D23.4a. Technical report [Online] http://www.compass-research.eu, 2013

COMPASS D24.2. "Report on Timed Fault Tree Analysis". Document Number: D24.2. Technical report [Online] http://www.compass-research.eu, 2013

COMPASS D34.1. "Test Automation Support". Document Number: D34.1. Technical report [Online] http://www.compass-research.eu, 2013

COMPASS D41.1. Accident Response Case Study Engineering Analysis Report Using Current Methods & Tools. Document Number: D41.1. Technical report [Online] http://www.compass-research.eu, 2013.

COMPASS D42.1. A/V/HA Ecosystem Prototype Using Current Methods and Tools. Document Number: D42.1. Technical report [Online] http://www.compass-research.eu, 2013.

COMPASS D43.2. Challenge Problem Analysis Using Existing Methods and Tools. Document Number: D43.2. Technical report [Online] http://www.compass-research.eu, 2013.

Judith Dahmann, George Rebovich, Ralph Lowry, JoAnn Lane, Kristen Baldwin. "An Implementers' View of Systems Engineering for Systems of Systems". Proceedings of the 2011 IEEE International Systems Conference. Montreal, Canada. 2011.

Judith Dahmann and Kristen Baldwin. Understanding the Current State of US Defense Systems of Systems and the Implications for Systems Engineering. In IEEE Systems Conference. IEEE, April 2008.

DoD. Department of Defense Dictionary of Military and Associated Terms, Joint Publication 1-02. 8th November 2010 (As Amended Through 15 November 2012). www.dtic.mil/doctrine/new_pubs/jp1_02.pdf (Last visited 9th December 2012).

INCOSE. International Council on Systems Engineering. Systems engineering handbook. A guide for system life cycle processes and activities version 3.2.1. Technical report INCOSE-TP-2003-002-03.2.1, INCOSE, 7670 Opportunity Rd, Suite 220 San Diego, CA, January 2011.

Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. Communications of the ACM, 44(10):29–32, October 2001.

Mark W. Maier. Architecting Principles for Systems-of-Systems. Systems Engineering, 1(4):267–284, 1998.

Claus Ballegaard Nielsen, Peter Gorm Larsen, John Fitzgerald, Jim Woodcock and Jan Peleska. "Model-based Engineering Systems of Systems". COMPASS Technical Report, 2013. Available from: http://www.compass-research.eu/resources/sos.pdf

Richard J. Payne and John S. Fitzgerald. Evaluation of architectural frameworks supporting contract-based specification. Technical Report CS-TR-1233, School of Computing Science, Newcastle University, December 2010.