Runtime Fault Prediction and Prevention for Emerging Services in System of Systems

Imad Sanduka[1], Tim Lochow[1] and Roman Obermaisser[2]

[1]EADS Innovation Work
Imad.sanduka@eads.net, Tim.lochow@eads.net
[2]University of Siegen
roman.obermaisser@uni-siegen.de

Abstract: SoS Models used for SoS design and requirements elicitation. At runtime operations SoS is under failure risk that resulted from its emergent behavior due to its constituent systems autonomy and system complexity. In order to mitigate SoS failure effects and prevent SoS failure we propose SoS model extensions and system failure prediction and prevention framework that enhance the usage of SoS state of arts models and provide emergent behavior control over SoS runtime operations.

1. Introduction

Emerging services are one of the prime reasons of constructing System of Systems (SoS). SoS considers joining diverse systems capabilities and taking advantages from their positive interaction. However, achieving the constituent systems interaction and collaboration under SoS community brings system engineering complexity and emergent behaviour challenges. SoS emergent behaviour can be desired or undesired behaviour. The desired behaviour is the goal of constructing SoS where it delivers services that can't be provided by one system alone. The undesired emergent behaviour is unexpected behaviour resulted from the interaction of autonomous constituent systems with operational and managerial independence. The undesired emergent behaviour affects negatively the SoS emergent services and increases the risk of system failure which must be avoided. During the SoS life cycle and under run time conditions where the system is in operation, undesired emergent behaviour and failure detection reduces the failure risk and makes the system more reliable.

This paper presents a framework for SoS modelling approach, for failure detection and prevention abilities. The framework enhances the SoS models with dynamic analysis properties and its usability under run time operations for undesired emergent behaviour detection and prevention. We introduce a failure detection approach with

failure handling and risk mitigation methodology supported by prediction of possible future SoS behaviour to guarantee the SoS survivability.

The first section is an introduction of the paper and its scope. In the second section SoS is discussed with its definition and emergent behaviour characteristic. The failure detection and prevention approach is presented in section three and a use case is illustrated in section four. Finally the future outlook and conclusion are presented in section five.

2. System of systems

Definition: System of Systems (SoS) are large-scale concurrent and distributed systems that are comprised of complex constituent systems (Sahin et al. 2007). The constituent systems are distributed over hardware, software, services and organizational systems, and characterized by operationally and managerially independence, evolutionary development and geographically distributed systems (Maier 1998). SoS are used nowadays in many sectors, e.g. military, air traffic management, emergency response, and water management and distinguished from monolithic systems by their unexpected emergent behaviour.

Emergent Services: The purpose of constructing the SoS is to provide emergent services that could not be achieved by one constituent system alone. The emergent services resulted from the cross interaction between the constituent systems that belong to the SoS and offer their services to each other. For example, in a military mission the air force system, surveillance systems, and communications systems work together in order to destroy the enemy tanks, constructing a SoS with clearly defined goals. Another example of emergence (Jamshidi 2008) is the symphony produced by an orchestra. The symphony is produced due to the interaction between different instruments and music players, where none of the music players can produce it in isolation.

While integrating the capabilities of heterogeneous systems brings new desired services, it may also lead to undesired emergent behaviour because of the constituent systems autonomy and their unexpected interactions. According to (Zhou 2011) ‘ The emergence of SoS cannot be foreseen through analysis because it comes from collaboration and autonomy of constituent systems’. In (Jamshidi 2008) the author presents the preferred definition of emergence as ‘something unexpected in the collective behaviour of an entity within its environment, not attributed to any subset of its parts, that it present (and observed) in a given view and not present (and observed) in any other view’. As emergent behaviour is unexpected, a climate facilitating the emergence of desired behaviour and mechanisms for the early detection of undesired emergent behaviour is required (Gorod & Gove 2007).

Desired and undesired emergent behaviour*:* Emergent behaviour is often unexpected behaviour resulted from the interaction of different autonomous systems. It could be a desired behaviour, represents an opportunity to the system, or undesired (bad) behaviour that forms a risk for the SoS. Desired emergent behaviour is the purpose of build-

ing the SoS in the first place, where a monolithic system cannot fulfil the requirements alone. Since the number of interactions between constituent systems increases exponentially with their number, emergent behaviour cannot be predicted in the current state-of-the-art. The choice for the constituent systems needed for constructing the SoS depends on their capabilities, communication characteristics, and their individual constrains, without considering the behaviour of these systems when they are working together. The problem with undesired emergent behaviour is its consequences on the environment, the SoS, and the constituent systems. Unexpected behaviour can prevent the SoS from offering its services, which can even imply the loss of lives in safety-critical systems such as military or emergency cases.

SoS failure: A failure is defined as the deviation of the behaviour from the specification that prevents a system from providing its intended services. In SoS we can distinguish two levels of failures: constituent systems failure and SoS failure. The consequences of failures at constituent systems level depend on the failure type and role of the constituent system itself in the SoS. Failures can occur due to physical effects that hinder the system from providing the required operations or due to software faults resulting in improper system behaviour. Failures can also be caused by faulty inputs related to the human behaviour and human decisions that could be mitigated by training and experience.

SoS fails when its desired and expected emergent services deviate from the specification. There are different sources of SoS failures such as design errors, constituent systems failure, environment parameters, and undesired emergent behaviour. Constituent systems failures can be tolerated using fault-tolerance mechanisms such as active redundancy defined at design-time .However failure caused by emergent behaviour or unexpected environmental parameters can only be mitigated by adaptation at runtime. The main challenge then is the detection of undesired emergent behaviour and the prevention of ensuing SoS failures.

Predicting SoS failure: SoS is a very large system where failures can cause a great damage. Thus predicting the SoS failure and preventing its occurrence is often preferred than dealing with the failure after its occurrence. In the design phase it is hard to predict all possible SoS failures due to several challenges:

- Complexity: SoS is usually large complex system consisting of heterogeneous and geographically distributed systems. Autonomous behaviour of the constituent systems increases the complexity of the SoS and its operations. During its life cycle the SoS evolves over time by joining new systems while others leaving and by targeting new missions and goals. The SoS operation depends on the interference between different systems with inherent dynamic changes and evolution through the SoS life cycle.
- SoS boundaries and environment: The boundaries of SoS are often ambiguous and cannot be determined. Due to the constituent systems diversity and their dynamic interactions under different systems boundaries, it is difficult to observe and enclose the interactions between constituent systems, and the interaction between the SoS with their environment. The SoS environment is not clear too, the constituent

systems interact with their own environment and are considered as a part of the others environment, which increases the SoS environment complexity. SoS keeps evolving over the time, considering constituent system evolution and technology adaptation, causing a dynamic changes on SoS boundaries and making them unstable. Constituent systems are also in dynamic changes; during the SoS life cycle, there are systems leaving the SoS and other new systems joining it, changing the SoS capabilities and offered services. The dynamicity of the SoS and its evolution over the time increase the uncertainty of its environment and its interaction with the environment. Another important property of the SoS that affects its environment uncertainty is the wide variety of its stakeholders due to its constituent system diversity.

- SoS Life Cycle: The system life cycle is defined as 'The evolution over time of a system-of-interest from conception through to retirement' (Haskins 2011). Monolithic systems have a clear life cycle that starts from the development of the system until it is out of commission. However, the definition of SoS life cycle depends on the type of the SoS and how it is constructed. Some SoS are constructed after the constituent systems realize that they can work together better under SoS environment, others are designed by the owner of the SoS in order to achieve a specified purpose. SoS that are constructed to achieve pre-defined objectives, the end of their life-cycle can be determined by achieving these objectives. For example, systems that work under a SoS in the military that is developed to achieve a specified mission, once the mission is ended, the SoS is no longer exist. In other SoS the life-cycle is not clear and there is no end for the life-cycle. In emergency case SoS, where the constituent systems work together to deal with emergency cases, there is no limitation on time. The constituent systems themselves have their own life cycle, some of them will be out of commission and leave the SoS while others will join. The SoS is typically subject to continuous evolution, it tries to deal with its diverse environment and so its life-cycle cannot be determined.
- Emergent behaviour: As mentioned before the emergent behaviour is a result of constituent systems interaction in the SoS and it cannot be predicted at design time.

Therefore, run time monitoring of SoS behaviour is required in order to predict an imminent SoS failure. At run-time the environment parameters are more clear and stable within a short period of time. The following sections present more details for run-time monitoring of SoS behaviour and run-time analysis for failure prediction.

3. Imminent SoS Failure Detection Based on Simulation of Future Behaviour

State of the art architecture: The purpose of the architecture synthesis process is to construct a complete SoS model that supports the operational and behavioural analysis of the SoS. SoS model is used by a variety of stakeholders at the development stage of the SoS, e.g. system integrators investigating the interactions between con-

stituent systems, suppliers implementing constituent systems, and public authorities and certification bodies assessing the safety of the SoS.

The current state-of-the-art modelling approach for SoS is to use architecture frameworks that describe the SoS from different viewpoints. The mainly used frameworks in this field are US Department of Defence Architecture Framework (DoDAF) (DoD Architecture Framework Working Group 2003), British Ministry of Defence Architecture Framework (MODAF) (British Ministry of Defense 2010), and NATO Architecture framework (NAF) (Anon 2007). The Unified Profile for DoDaF and MoDAF (UPDM) (OMG 2010) was created by OMG group to enable modelling of SoS based on the DoDAF and MoDAF architectures. It supports the ability to model a wide range of complex systems at different levels of abstraction. UPDM has the capability to describe the operations and functions of SoS, but it does not support the specification of SoS behaviour and its constituent systems. UPDM introduces multiple views that depict different aspects of the system. It is useful in requirements elicitation, system components specifications definition, and constituent systems interfaces description and their interaction points. Using UPDM, the required functions to be achieved by the SoS operations can be defined. These functions are considered as the major point in selecting the constituent systems to achieve SoS goals.

Fig. *1* illustrates the model growth through the requirements elicitation and analysis process.

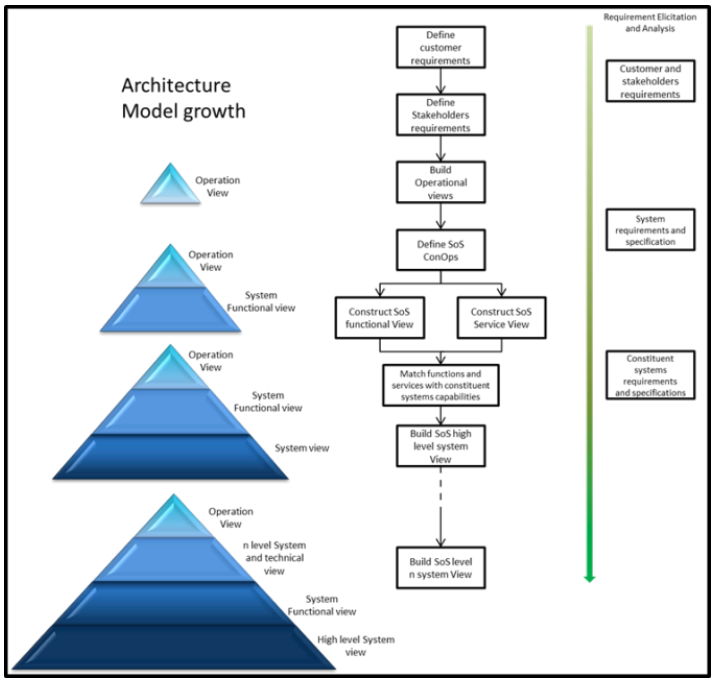

Fig. 1 SoS model propagation

The process starts from the customer side in order to collect the customer needs and convert them to requirements. Together with the stakeholders' requirements, the requirements are documented and analysed to build the first high-level specifications of the SoS. The first step in building the SoS architecture model starts with constructing an operational view which used to illustrate the operational aspects of the SoS. It gives a general view of what the system will achieve and through which operations. A scenario-driven process can be used in order to reduce the complexity of the operational flow construction process. At this stage the requirements are modified and new requirements are added and documented.

4. Model Extension

SoS and constituent systems behaviour: Dynamic analysis requires both the SoS behaviour and the constituent systems behaviour. At a high level of abstraction discrete behavioural models are efficiently used to describe the system behaviour. Discrete models are generated using SysML state charts. The state charts or state machine diagram describes the lifecycle behaviour of a system; it depicts the different states of the system and their transitions in response to events during the block life cycle (Friedenthal et al. 2006).As UPDM is an extension profile to SysML and UML, the SysML profile is integrated. Using the system view 10a(SV10a) in DoDAF within the UPDM profile, SoS behaviour using the state chart model is described. The same applies to the constituent systems where another SV10a is generated to include the constituent systems behaviour. As soon as we get all the specification of required systems behaviour, SV10a views are connected in the systems view, where the SoS architecture is built, and the constituent systems interactions are described. The next required step is to develop an extension profile that connects the operational and functional flow views with the SoS behaviour chart. This profile will make it possible to synchronize the SoS status with its operations and functions.

Operations constrains: The failure detection process depends on indicators and constrains defined at the operational level. It is required to define the critical constraints and map them to the SoS operations in the operational flow diagram to represent a behavioural reference for SoS desired behaviour. The failure detection process depends on monitoring these constraints and make sure that they are satisfied. If not, it indicates systems failure (deviation from the desired behaviour). It is possible to add these constraints as attributes within the UPDM SoS model for each operation. Constrains are used as indicators which facilitate failure detection using temporal logic definition.

Architecture Patterns: As described in (Kalawsky 2013) architecture patterns are used to facilitate the system reconfiguration and evolution. According to (Kalawsky 2013) Patterns could be used as a template for the structure and behaviour of the system. Using the system view from DoDAF under UPDM and by providing Constituent

systems to functions mapping, and constituent systems capabilities, possible architecture patterns are generated in the system view. At the design phase different patterns could be generated and trade studies between these patterns are implemented. Due to the SoS complexity and evolution, the wide variety for the constituent systems and their interaction, and the instability of SoS environment, one pattern is not sufficient for all SoS statuses and operations. For example, some of the patterns are cost effective, others are operational effective while others are time effective. The choice of which pattern to be used depends on the current and nearest future objectives and constrains of SoS.

By connecting the constituent systems to the generated architecture patterns we end up with executable model that includes the SoS behaviour as well as the constituent systems behaviour.

5. Run-Time Fault Prediction and Preventing Engine

The next step is to extend the model for analysis and failure detection. For this purpose a Runtime Fault Prediction and Prevention Engine (RFPPE) will be developed. The engine will be configured with the specification of the SoS and the constituent systems. The purpose of RFPPE is to monitor the SoS and its constituent systems behaviour. Figure 2 illustrates the main parts of RFPPE:

- Failure detection unit
- Next step simulation engine
- System reconfiguration
- External rules controller

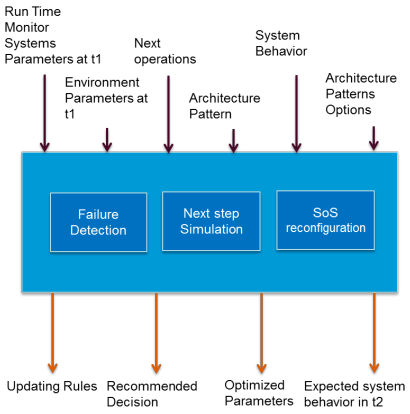

Fig. 2 RFPPE

The RFPPE targets directed and acknowledge SoS. Directed where SoS has a specific purposes for which it is built and managed to Achieve (Maier 1998) and the constituent systems share part of their ownership and funding. Acknowledge SoS has a defined goals and objective and its own management and resources while the constituent

systems keep their own independency regarding ownership, funding and goals (USA Department of Defense 2008) . In both cases when a constituent system decides to belong for the SoS it shares information about its environment, status, and capabilities with SoS management. RFPPE approach collects information about the SoS constituent systems states and parameters, the SoS environment parameters (collected from services systems e.g. surveillance , traffic management), the current state of the SoS (from SoS internal report) , the next step for SoS (from the operational view) , and the different options for the SoS architecture patterns that could be used. Depending on the current status and the coming steps, RFPPE provides the best option (that fits with the SoS goals and constrains, and prevents future failure) to be used while predicting the results using a short simulation for the next operational steps. Using the short simulation process, it predicts the future emergent behaviour that will affect the system operation and tries to mitigate or eliminate its bad effect by reconfiguring the SoS structure and its constituent systems parameters.

The purpose of RFPPE is to effectively:

1. Manage SoS operations: controlling the expected behaviour of SoS using rules
2. Enhance decision making process: by predicting the future results of decision options and providing the optimum.
3. Detect and Prevent SoS Failures: by monitoring the SoS behaviour, detecting failures, and preventing future failures.
4. Mitigate undesired effect of system emergent behaviour: responding to emergent behaviour by reconfiguring the SoS to reduce the bad emergent behaviour effect.
5. Prevent future undesired emergent behaviour effect for the next operational steps: reacting to predicted future emergent behaviour before its occurrence
6. Leverage, modify, and update system working rules: saving feedback information for failure handling process and SoS configuration to be used for repeatable failure modes.

Failure detection unit: Used to detect failures at SoS level by monitoring the SoS behaviour and constituent systems behaviour. UPDM model provides information about SoS operational and functional flow. At each operation SoS parameters and indicators are defined. For example assume an operation number x ($op(x)$) , Knowing that the expected value for system indicator A at $op(x)$ must be within the range of $y \leq A_{op(x)} \leq z$, failure occurs when the system indicator is out of its expected value. Failure unit detects such behaviour deviation and alerts the system for failure existence.

Next step simulation engine: The purpose of simulation engine is to predict the future behaviour of the SoS under certain configuration. By connecting the SoS model with SoS behaviour and the constituent system behaviour the future behaviour could be predicted by simulating the next step of the SoS under the current systems and environment parameters. The Operational flow indicates the next step of the SoS, and by extending the SoS model with the connection of functional flow diagram and the constituent systems behavioural model, the exact position of the current status is defined. Knowing the current status and parameters, next steps and operations, and cur-

rent systems configuration, SoS future behaviour is simulated and together with the failure detection engine, any future failure or undesired emergent behaviour will be detected and reported.

System reconfiguration: The system reconfiguration task is to reconfigure the SoS in the case of future failure is predicted out of the current configuration. Its purpose is to find the possible configuration patterns that could be used to overcome the current failure. The new configuration will be also checked for future failures by simulating the systems again with the new configuration.

External Controller: Within our SoS model failures are classified to:

1. Expected failures: failures that are expected at design stage and could not be avoided or the risk for failure elimination cost is more than failure handling. In this case the failure handling process and the required SoS reconfiguration for failure effects mitigation process are already defined.
2. Unexpected failure due to emergent behavior or external environmental impact.
3. Historical failure: failures that occurred before and handling process is already defined and saved.

The purpose of external rules controller is to increase the failure handling process speed and efficiency. It considers the expected and historical failures where predefined procedure and SoS reconfiguration are defined before and no further analysis required. Once unexpected failure occurs and handling process is defined, the controller will be updated and the failure conditions will be registered. Figure 3 shows the integration of RFPPE within the whole model.

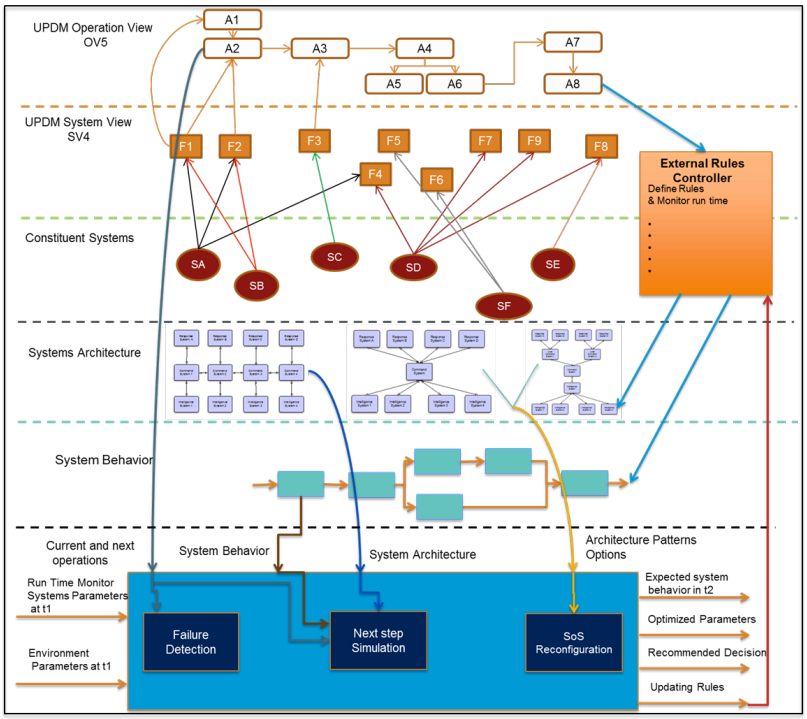

Fig. 3. RFPPE with model integration

Functional Description: Figure 4 describes the functional flow for the whole system. The run time operations are monitored for the current and next step of operation. If the system faces a known and expected behavior that is pre-defined in the rules, the constituent systems will be reconfigured according to the rules and the next step will be proceed. If the system faces unexpected or emergent behavior where there are no rules were defined, RFPPE will be lunched to choose the best configuration of the system that mitigates the bad effect of the unexpected behavior and prevents the system from running into further bad emergent behavior by predicting the next step results. The RFPPE collects all the information and parameters needed to describe the current status and look through available configuration patterns that could be used to deal with such situation. It will consider these patterns and run a simulation for the nearest next steps. By analyzing the simulation results and detecting any further emergent behavior that can be resulted from the chosen pattern, the System Reconfiguration part will suggest the best pattern to be used for the current situation and under the current environment parameters values.

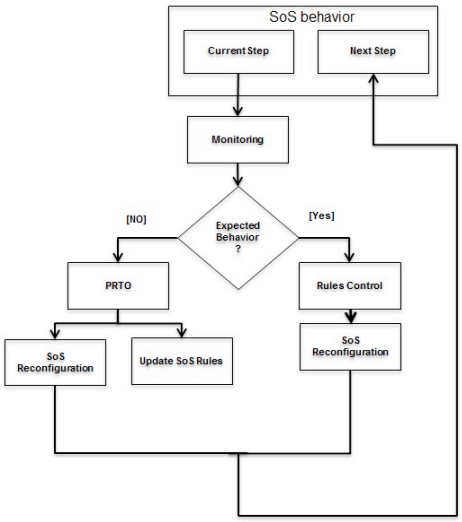

Fig. 4. RFPPE Process description

6. Example Scenario

In Emergency response system diverse systems collaborate together to provide emergency services for civilians. Command and Control Center (CCC) as a part of the emergency response system is a typical example of SoS where heterogeneous systems interacting with each other to efficiently mitigate the risks and consequences of emergency case. CCC as a system node coordinates the operations of other nodes i.e. Police headquarters, fire brigade, and medical units, and uses the emergent services out of this collaboration to deal with emergency cases. Figure 5

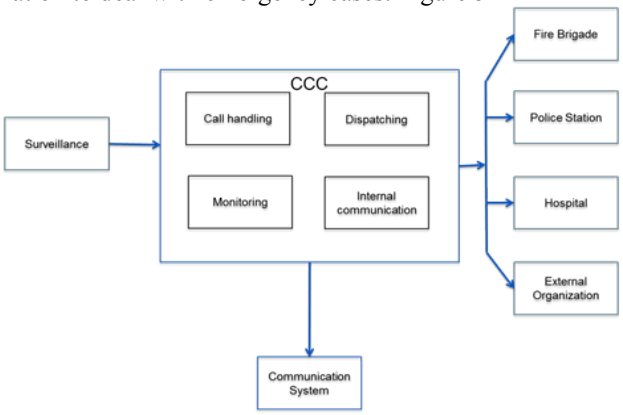

Fig. 5 Command and Control Center

Each of the system nodes represents an interaction unit and includes different operational nodes where autonomous and heterogeneous constituent systems work together to provide the required services and operations. Figure 5 depicts an example of different operational nodes (e.g. Call handling) that working together within the CCC. The operational nodes join the capabilities of diverse constituent systems within different activities to do its operations that require an interaction of hardware, software, services and organizational systems. CCC receives notifications about emergency case from external sources (i.e. people) through its call handling node. The call handling node gathers emergency case information from the callers and distributed surveillance systems and forwards it to the dispatching node where the emergency case information is analysed and the required emergency response is issued together with the dispatching plan. The dispatching plan is then distributed to the emergency units (i.e. Police HQ, Fire Brigade, Hospitals). The emergency units handle the emergency operations on the site and report back the site information to the CCC, which uses this information together with the surveillance information for monitoring the emergency case and taking the required actions and forwarding it back to the emergency units.

The communication systems provide the communication and data exchange services that connect the constituent systems together and facilitate their interaction. At a high level of abstraction there are two parts of communication; internal and external. The internal provides communication services for the constituent systems that are distributed within one system node, while the external one is responsible for the communication between the system nodes. For example, exchanging the emergency report between the call handling and dispatching unit is done using the internal communication system while exchanging this report with the police HQ is done using the external communication system.

Out of this constituent systems collection, there are many causes of emergent behaviour that could affect the emergent services of our SoS. Failure could occur at each of the constituent systems causing unexpected reactions of other systems. The same could happen where unconsidered behaviour by one of the systems resulting in a conflict for other systems. The environment parameters and environment changes represent a very important source for emergent behaviour where the SoS environment can't be defined at the design phase and thus not all the environment parameters are considered in design.

Communication systems failure, in our use case, considered as a critical failure that affects negatively the SoS behaviour and services. The constituent systems connectivity depends on the communication services that provided by the communication systems. Once the connection is lost, the SoS could not deliver its emergent services as the systems will be disconnected and their autonomous behaviour leads their interaction under the current situation. As an example of communication failure, consider an emergency rescue for people detained by fire in a building. Joint operations required between the police and firemen in order to evacuate and rescue the detained people. CCC is responsible for organizing this situation and issuing the required functions and dispatching plans for the police HQ and fire brigade.

There are pre-defined rules for this kind of operations where the number and kind of functions that must be sent to the site are defined. After the notification is received an explosion causes damage in the communication network between the CCC, PoliceHQ, and Fire Brigade preventing data exchange process between the systems at these nodes. At this kind of problem each of the systems starts to operate autonomously according to its pre-defined rules without knowing any information about the other systems. The dispatching systems at each node issue the required functions for the situation and try to fulfil the required operations by its own resources, which will double that functions that will be sent to the site. This behaviour causes chaos at the site and blocks the roads to the building due to the double number of rescue cars which consequently hinders the evacuation process resulting in loss of lives and increase the number of injured people.

The failure of the communication system could be detected by monitoring the data transfer rate between the constituent systems. By simulating the next step for the emergency response process, the deviation of the number of functions sent to the site from the required one implies undesired emergent behaviour out of the communication failure.

To avoid the undesired emergent behaviour, the CCC should change the communication pattern between the constituent systems to another one that could avoid the communication damage, and do the appropriate reconfigurations. One pattern could be used is to use the mobile communication unit that provides a temporal communication service at the emergency site and mitigates the risk of the communication failure. Another pattern could be changing the control and organization rules to local collaboration units that use the Radio communication system. The simulation results of the next steps after choosing one of the patterns ensure if a future emergent behaviour will occur out of using this pattern. If so, the pattern will be excluded and another one will be used.

7. Conclusion

The RFPPE approach is a method used to construct an adaptive robust SoS model that mitigates the effect of undesired emergent behavior, and offers a solution for reducing the SoS complexity by automatically chose the optimum design and pattern that fits for the current SoS situation. It can be used in two stages in the SoS development; Real time operations and Design phase under simulated environment. Different technologies are planned to be considered within the PRTO solution developments, Design Patterns, Design by Rules, and UPDM modeling as well as SyML modeling languages. Other technologies that could be part of the solution are the Agent based behavioral models that models the SoS evolution and game theory that enhances the model adaptability.

Acknowledgments This work was supported in part to European Commission for funding the Large-scale integrating project (IP) proposal under the ICT Call 7 (FP7-ICT-2011-7) ‘Design for Adaptability and evolutioN in System of systems Engineering (DANSE)’ .

References:

Anon, 2007. *NATO ARCHITECTURE FRAMEWORK v3*,

DoD Architecture Framework Working Group, 2003. DoD Architecture Framework: Volume I: Definitions and guidelines. , I(April 2007). Available at: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:DoD+Architecture+Framework+Volume+I+:+Definitions+and+Guidelines#1 [Accessed February 19, 2013].

British Ministry of Defense, 2010. MOD Architecture Framework. Available at: https://www.gov.uk/mod-architecture-framework#use-and-examples-of-modaf.

Friedenthal, S., Moore, A. & Steiner, R., 2006. OMG systems modeling language (OMG SysML) tutorial. *INCOSE Intl. Symp*. Available at: http://eng.umd.edu/~austin/enes489p/lecture-resources/SysML-Friedenthal-Tutorial-INCOSE2006.pdf [Accessed October 23, 2012].

Gorod, A. & Gove, R., 2007. System of Systems Management : A Network Management Approach. *System*, pp.1–5.

Haskins, C., 2011. *Systems engineering handbook* v3.2 ed., INCOSE. Available at: http://smslab.kaist.ac.kr/Course/CC532/2012/LectureNote/2012/INCOSE Systems Engineering Handbook v3.1 5-Sep-2007.pdf [Accessed May 22, 2012].

Jamshidi, M., 2008. *System of Systems Engineering: Principles and Applications*, CRC Press.

Kalawsky, R.S. et al., 2013. Using Architecture Patterns to Architect and Analyze Systems of Systems. *Procedia Computer Science*, 16, pp.283–292. Available at: http://linkinghub.elsevier.com/retrieve/pii/S1877050913000318 [Accessed April 8, 2013].

Maier, M., 1998. Architecting principles for systems-of-systems. *Systems Engineering*, 1(4), pp.267–284. Available at: http://www.infoed.com/Open/PAPERS/systems.htm [Accessed May 16, 2012].

OMG, 2010. *Unified Profile for the Department of Defense Architecture Framework (DoDAF) and the Ministry of Defence Architecture Framework (MODAF)*, Available at: http://www.omg.org/spec/UPDM/2.0.

Sahin, F., Jamshidi, M. & Sridhar, P., 2007. A Discrete Event XML based Simulation Framework for System of Systems Architectures. *IEEE*, pp.1–7.

USA Department of Defense, 2008. *Systems Engineering Guide for Systems of Systems*,

Zhou, B. et al., 2011. Modeling system of systems: A generic method based on system characteristics and interface. *2011 9th IEEE International Conference on Industrial Informatics*, pp.361–368. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6034903.