30th Annual INCOSE international symposium
Virtual Event
July 20 - 22, 2020

Towards a Unified Approach to System-of-Systems Risk Analysis Based on Systems Theory

Jakob Axelsson
Mälardalen University
PO Box 883, SE-721 23 Västerås, Sweden
jakob.axelsson@mdh.se

Abstract. In systems-of-systems (SoS), trustworthiness is a key concern, which includes operational risks such as safety, security, and privacy. These are particularly challenging to achieve in an SoS, due to the shared responsibility among independent constituent systems. This paper investigates a unified approach based on systems thinking for analyzing risks in SoS. Having a common framework is important, since many risk areas are related and affect each other. The paper also discusses how traditional static risk analysis can be complemented by dynamic techniques that collect data over time to fuel SoS evolution with respect to risk reduction.

Introduction

Systems-of-systems (SoS), with their origins in primarily the defense sector, are now rapidly becoming increasingly relevant in commercial applications as a result of the software-driven digital transformation and automation of industry and society. Examples exist in domains such as transportation, energy, health care, manufacturing, and smart cities.

Very often, the constituent systems (CS) of the SoS perform control of physical devices, which makes them critical from several perspectives. This includes safety, since the physical devices may cause harm to humans, but also security and privacy, since they will handle information whose exposure could imply significant loss of value. Further, the fulfillment of the SoS mission is critical, and reliability and robustness to changing circumstances are key aspects.

This is also reflected in the SoS scientific literature, where risk management is an important topic and includes properties such as sustainability, safety, security, and reliability (Axelsson, 2015). Klein and van Vliet (2013), who surveyed 200 papers on SoS architecture, had security, safety, and risk among the top 10 concerns, and Bianchi et al. (2015) concluded from a survey of 40 papers that security, reliability, safety, and dependability are among the top 10 quality attributes.

For this reason, risk analysis becomes essential in the development of SoS, and the particular SoS characteristics, where the SoS can be very long lived, and where each CS evolves over time, make it necessary to have a broad and life-cycle oriented perspective on risk analysis. Often, the different properties already mentioned become interrelated in the context of an SoS, as exemplified by the fact that a security vulnerability that allows someone to tamper with physical devices in the SoS can also lead to safety risks. Therefore, it makes sense to have a general risk analysis method for SoS, rather than specific methods for different properties. A generic method also allows SoS engineers to trade off different risks against each other within the same framework.

The contribution of this paper is to outline a unified SoS risk analysis method, and to show how it can be applied to safety, security, and privacy related risks. The paper is structured as follows: In the next

section, some basic terminology is introduced, and it is explained where risk analysis fits in the SoS engineering (SoSE) process. Then, systems theory-based risk analysis methods are introduced. These methods are subsequently applied to problems related to SoS safety, security, and privacy. The methods are however static and used prior to putting an SoS in operation, and it is discussed how they can be complemented with dynamic approaches that also monitor SoS execution. Finally, some related work is introduced before summarizing the conclusions of the paper.

Establishing trustworthiness through system-of-systems engineering

Establishing trustworthiness in a system can be seen as reducing the presence of operational risks. ISO (2009) uses the term "risk" to mean the "effect of uncertainty on objectives". This definition is however not consistent with the everyday usage of the word, which normally associates risks with negative values, since it also allows positive effects to be regarded as risks (something that would usually be called opportunities instead). In our work, the aim is to reduce negative effects, and hence we will use the following more restrictive definition: *Risk is the negative effect of uncertainty on objectives.*

A "negative effect on objectives" is a loss of value to some stakeholder, and "uncertainty" is the occurrence of events over which the system does not have full control. An event is typically that a particular system state is entered, or that the environment is in a certain condition, and it is the combination of these two that results in the loss. A typical risk can thus be formulated as follows:

<Loss of value occurs> **if** <System is in hazardous state> **when** <Environmental condition applies>.

As an example, a traffic related risk could be "Pedestrian gets injured" **if** "Car brakes do not work" **when** "Pedestrian walks into road in front of car". Both the system being in the hazardous state and the environmental conditions are uncertain events which may or may not apply, and the loss of value is thus an effect of uncertainty. But it is the combination of the system state and environmental condition that leads to the loss, and not just one of them. Risk analysis and mitigation is mostly about identifying the elements of such risk expressions, and trying to eliminate them, or at least to reduce their probability of occurrence or their severity.

Risk related taxonomies

Losses. A set of general loss categories are recurrent and can be used as a starting point for finding which losses of value to consider for a particular system or SoS:

- *Mission loss*. The system's mission is not fulfilled and the benefits are not fully achieved.
- *Human death or injury*. The system causes harm to people, such as operators or bystanders.
- *Material damage*. The system causes damage to material within or outside it.
- *Information losses*. The system causes information to be exposed to non-authorized entities.
- *Economical damage*. The system causes fines or loss of reputation to its owner because of not meeting legal or contractual requirements.

Hazards. A taxonomy of the hazards that can cause risks in SoS has been proposed by Redmond (2007). It distinguishes between *single system hazards*, that are attributable to an individual CS within the SoS, and *emergent hazards* that are caused by the interactions. On the SoS level, it is primarily the latter that are relevant since the single system hazards can be seen as the responsibility of the organization that owns that system (whether it is part of an SoS or not). The emergent hazard types are, according to Redmond (2007):

- *Reconfiguration hazards*: Result from the transfer of an SoS from one state to another, such as when forming or dissolving constellations.
- *Integration hazards*: Emerge as the result of integrating systems into an SoS, and is further subdivided into:
    - *Interface hazards*: Caused by transfers over a defined interface.
    - *Proximity hazards*: Caused by transfers by other means than a defined interface.
    - *Resource hazards*: Result from insufficient shared resources or resource conflicts.
- *Interoperability hazards*: Occur when an information transfer is interpreted by the receiver in a manner inconsistent with the intent of the sender.

SoS states. Axelsson (2019a) describes a set of generic CS states and these can also be used to provide some light on the hazards in the context of an SoS:

- *Ignorant*: A system which has the relevant capabilities but does not meet the requirements of the SoS and is hence unable to participate. Only single system hazards are present.
- *Prepared*: A system which meets the requirements of the SoS but has not yet become part of it. The preparation for an SoS made to the system can introduce new single system hazards when used in its original context if the SoS related behavior is erroneously activated there.
- *Passive*: A CS that has joined the SoS but is not participating in any constellation. This usually involves the exchange of credentials and can lead to intrusion related integration hazards.
- *Active*: A CS that is participating in a constellation of the SoS. Relying on data from other CS can lead to wrong behavior. All kinds of emergent SoS hazards are present here.

Risk management in SoSE

An important part of SoSE is to ensure that emergent hazards do not occur, and this is primarily achieved by defining requirements on the CS behavior. The SoS architecture provides the context of those requirements, including what CS and mediating systems to include, what their roles and responsibilities are, and what means of communication should be present. Since many CS may be pre-existing, and have a role to play also outside the SoS, the SoS designers need to consider the non-SoS related requirements and constraints on each CS. This is illustrated in Figure 1.

SoS risk analysis has some challenges related to scope and quantification. Concerning the scope, the purpose is to focus on the SoS emergent hazards. Therefore, it is important to view the CS as black boxes as far as possible. It is always tempting to dig into every detail of the CS, in particular since they are often pre-existing and there can be an abundant amount of information and documentation available. However, this can result in too much time being spent on the CS level, with risks on the SoS level being overlooked.

When it comes to quantification, likelihood of events is a key concept of the risk definition stated above. However, the SoS level risks are usually unprecedented, and there is a lack of probability data. It is often counter-productive to try to guess these probabilities, and instead qualitative reasoning is needed. Also, evolution is a defining characteristic of SoS, and even small changes in the SoS or a CS can lead to large effects on probability of events, thus making an even-so detailed initial quantitative analysis invalid as time runs.

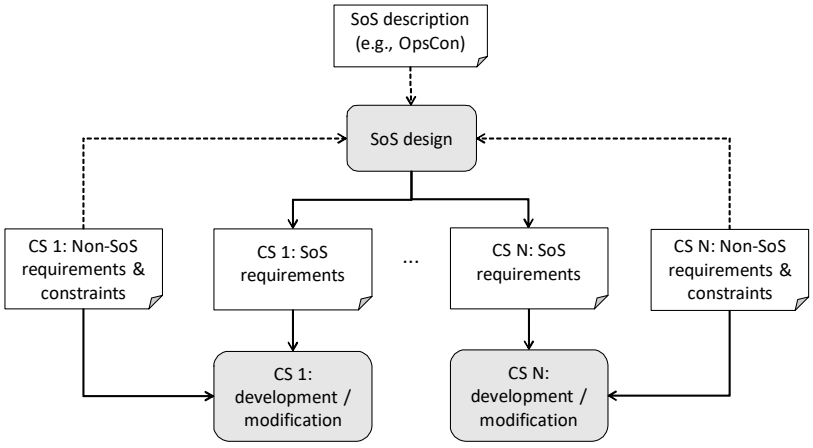

Figure 1. Overview of SoS design process.

SoS risk analysis based on systems thinking

Operational risk analysis methods have been developed over many decades, especially to address safety risks. However, many of the methods proposed are reductionist and focus on the identification of faults (such as a single component breaking down) and their implications on hazards and losses. In an SoS setting, this can be compared to focusing on single system risks while ignoring risks that emerge from CS interactions.

Nowadays, it is clear that many losses do not occur due to faults but are rather the result of normal interactions of different actors in a complex system. Also, the role of management is now better understood. In many organizations there is a continuous pressure to save costs, which can lead to the reduction of safety margins on the normal functional behavior. Over time, this can result in situations where a small variation in normal behavior that was initially within the safety margin suddenly leads to a severe loss. Management is thus a key actor in the system, and the organizations owning both the SoS and the individual CS must be considered.

The insight that many risks are systemic has led to a new view on risk analysis, and one of the most influential researchers in this area was Rasmussen (1997). His systems thinking based approach models the system-of-interest (SoI) using functional abstractions which makes it plausible to include in the analysis not only hardware components, but also software, human operators, management and organizations, which is a must in today's complex, socio-technical SoS.

Risk management is in his approach seen as a cybernetic control task. The different actors are striving to adapt to changes in a dynamic environment, and the functional models of the actors need to consider their individual goals; priorities; functions; processes; and configurations. The control measures can include both feed forward control (such as using plans and routines to drive behavior) and feedback control (where measurements and observation is used to determine adequate actions). As is clear from the law of requisite variety (Ashby, 1956), risks can occur as a result of actors lacking proper feedback from the processes they are trying to control, but it can also be a result of missing information about control objectives and constraints.

An approach to reducing risks is thus to analyze the information flows in a model, and if there are actors that are missing adequate information, more flows can be added. Once hazardous states have been identified, the system can be controlled so that those states are not entered, thereby avoiding the release of a hazard. Also, the consequences of a hazard that nevertheless gets released can be limited by trying to terminate the flow of events that leads from the hazard to a loss. Both these strategies

involve adding controllers in the system. In an SoS setting, this can be done either by allocating additional responsibilities to existing CS or adding mediating systems to perform such control tasks.

Apart from additional controllers and information flows, "wrong" decisions by the actors can be a cause of hazards. In an SoS, the CS retain their own independent objectives. This can lead to a situation where the CS ignore their commitment to SoS goals and prioritize their own benefit. SoS risk management thus needs to include incentives to stimulate the desired behavior (Axelsson, 2019b).

In this paper, we will use the approach of Rasmussen (1997) and investigate further how it can be applied in SoS as a generic method of risk analysis. The SoI will be modelled using functional control diagrams, and these are used to reason about safety, security, and privacy in an SoS setting.

Safety

Rasmussen's work lays a fundament for risk analysis, and especially safety, but he does not provide a process and method for conducting safety analysis. Therefore, other researchers have continued his work in order to better operationalize the theories, such as Leveson (2011) who introduced the System-Theoretic Accident Model and Processes (STAMP) method. This method provides more hands-on guidance on how to perform safety analysis and it consists of three main steps, namely (1) Loss and hazard analysis; (2) Modeling; and (3) Causal analysis. These activities will now be described in more detail with respect to an SoS context following Axelsson and Kobetski (2018).

Loss and hazard analysis

The goal of this activity is to identify and characterize the negative effects that can be caused by the SoS. The activity consists of three sub-activities that all deal with the SoS at a high level:

- *Loss analysis*: Uses the SoS description (e.g., OpsCon) to identify a set of losses (typically from the five generic categories introduced above) that the SoS should try to avoid.
- *Hazard analysis*: Based on the losses, hazards are identified. The process for identifying hazards is basically to start with a loss and consider (using domain knowledge) what SoS states or conditions that could lead to that loss.
- *Constraints analysis:* Based on the hazards, constraints are defined. These are high-level requirements that the SoS should fulfil, and the subsequent analysis aims at breaking down these constraints in a structured way to concrete requirements that can be allocated to the CS.

Modeling

The modeling activity aims at deriving a functional control model of the SoS, that is suitable for risk analysis. It consists of two sub-activities:

- *SoS modeling*: A textual analysis of the SoS OpsCon to identify concepts that should appear as elements in the model and the information flows between them. This requires domain knowledge that is often partly proprietary to each CS owner, and the activity thus requires active involvement from them.
- *Model analysis*: Takes the SoS control model and verifies it by looking for lacking information, such as missing flows in the control loops, and inconsistencies.

Modeling is often carried out in parallel to the loss and hazard analysis. The modeling also results in the identification of missing or ambiguous information in the SoS OpsCon that require clarifications.

Causal analysis

Starting from hazards and the SoS control model, the causal analysis identifies how the system could cause losses to occur. The activity is divided into three parts:

- *Control action analysis*: The hazards could occur because some part of the system performs an inappropriate action, and this can be analyzed by looking at the output control actions of all controllers in the model. STAMP provides a set of guidewords to aid the analyst in identifying how an inappropriate control action can lead to a hazard in a specific context (i.e., a state of the system and its environment). Note that it is only the context that is added in this step, whereas the other elements are already given from the hazard analysis and control model.

- *Causal analysis*: The list of inadequate control actions is further analyzed to understand how they cause the hazard by identifying scenarios why and when the issue could occur. This is done by tracing the information flows backwards through the controller, and forward to analyze the consequences if an appropriate control action is not followed. The same cause may very well show up in several scenarios, and thus relate to several inadequate control actions (which relate to different hazards or different keywords).

- *Requirements elicitation*: The final step is to elicit requirements that remove the hazards from the system, or at least reduce their likelihood of occurring or their severity. This is done by looking at the scenarios from the causal analysis and defining ways to prevent them from happening. This can either include adding a new controller, giving extended tasks to existing CS or mediating systems, or changing incentives, as discussed previously.

STAMP in the SoSE process

In our previous work, we have applied STAMP to the risk analysis of safety critical SoS in the transportation domain (Axelsson and Kobetski, 2018). From that experience, the method serves as a useful basis for static analysis. It is a clear strength that it is suitable for various kinds of risks, and not just safety, because this makes conflicts between different losses explicit. It is often the case that safety precautions limit the performance and restricts functionality, which can be seen as a mission loss, and in those situations suitable trade-offs are needed.

Although the method was useful, we have also identified a number of areas where further methodological development is necessary to enhance its applicability in SoS. The intention of STAMP is that it should be applied in early systems engineering phases, but it still requires a clear and unambiguous description of the SoI and its functionality. This is often lacking in an SoS context, due to the multiplicity of independent stakeholders, such as CS owners, who have to agree on functionality and interfaces, and who may often be reluctant to share details about their systems for commercial reasons. This leads to a situation where either risk analysis ends up being done based on guesses and assumptions about the functionality and architecture, or is pushed to later phases of the SE process where the possibility to influence the design is reduced.

The functional control models employed in STAMP also lack rigor and flexibility. It is not described in the method how to model the relevant states of different controllers, which is fundamental to the analysis. Also, it assumes a static structure of the control diagram, but in an SoS the structure is dynamic, with different CS constellations forming and dissolving over time. There is thus a need to handle different configurations in an efficient way, avoiding the need to redo the entire analysis for each configuration. This requires tool support, and linking those to the existing tool chain in a model-based systems engineering process.

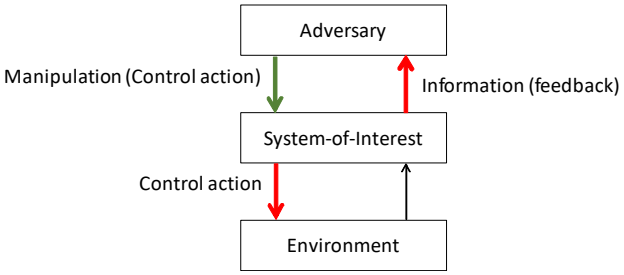

Figure 2. Modeling security as a control system.

Although STAMP claims to be system-theoretic, this really only applies to the use of control models as a basis. Other concepts from systems theory, such as the emphasis on analyzing enforcing and balancing feedback loops, is missing. The loops are important due to the risks they can cause over time, such as the growing cost pressure that gradually reduces safety margins as mentioned earlier. Also, when the number of CS grows, a communication resource which was initially ample can after some time become a bottleneck that suddenly leads to a resource hazard.

STAMP claims to be superior to other methods regarding the analysis of multiple causes leading to a hazard. This is not a false claim, but the support in the method for this is not very explicit and it is to a large extent up to the analyst to identify those connections between causes. This can be difficult to achieve manually in a complex scenario, and proper tool support is lacking.

STAMP explicitly discourages the use of quantifications of probabilities and effects, and as discussed in the introduction of this paper, there are good reasons for doing so also in an SoS context. However, in a practical situation there is often an overwhelming number of potential risks to consider, and the analysts have to prioritize which of these should be given sufficient attention to avoid stalling the project. Without quantitative support, this prioritization becomes ad hoc.

Security

Security is concerned with adversary attempts to manipulate a system in order to gain benefits. This is modelled as a control system in Figure 2, showing how an adversary actor tries to manipulate the SoI through control actions to either make the SoI feedback valuable information, or make it change its behavior by producing other control actions towards its environment than it otherwise would. If the manipulation and its consequences lead to some kind of loss for the SoI stakeholder, defenses should be put in place. Note that the figure shows that the losses may relate not only to traditional security problems such as loss of information, but also safety or loss of mission.

Adversary motives

One way of understanding what security related risks exist is to reason about potential adversaries and their motives. If no adversaries exist, or their motives are weak, the risk is lower that they will consider the effort worthwhile. Ablon (2018) identified four types of adversaries: *cybercriminals*, motivated by financial gain achieved by stealing valuable data; *state-sponsored actors* motivated by advancing the interest of their nation-state or support a political agenda through espionage or sabotage; *hacktivists* driven by ideological activism to steal and leak sensitive information or interrupt the service of a system; and *cyberterrorists* who try to gain support for a cause or ideology through physical damage or interruption of critical infrastructure.

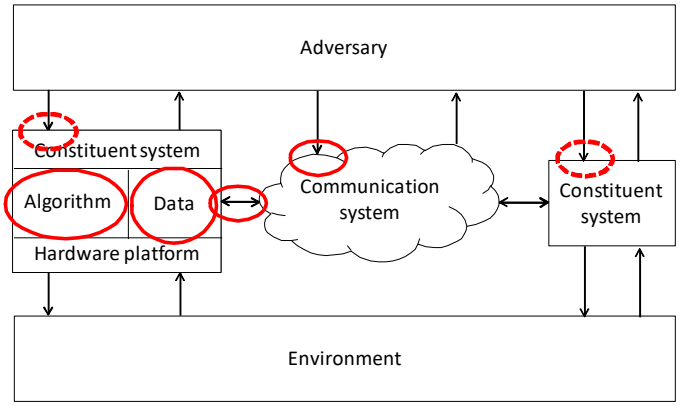

Figure 3. SoS attack surfaces.

The construction of an SoS can produce new or additional motives to attack, based on the SoS purpose which is to provide emergent behavior. The SoS requires new information to be produced which does not exist in the individual CS, and if this emergent information is valuable to an adversary, it could be interesting to attack the SoS even though the individual CS are not interesting.

Attack surfaces

The SoS context can also open new attack surfaces that did not exist in the CS before, and this is illustrated in Figure 3. The CS keep their original single system hazards that they had prior to joining the SoS (the dashed circles in the figure). The added entry point is primarily the communication system which can be disrupted, making collaboration in the SoS impossible. Additionally, the adversary can penetrate it to send data requests from within the SoS, so there is a secondary attack surface between the communication system and each of the CS. Note that these interfaces need to be protected in both directions, to shield a CS from manipulation and to protect the other CS from a rogue CS that has been captured by the adversary. If the adversary manages to get into a CS, they can change the algorithms and data inside it to affect its behavior and cause it to take undesired actions.

To further guide the analysis of SoS security risks in a specific scenario, a threat model such as STRIDE (Kohnfelder and Praerit, 1999) could be used. The categories included are presented in Table 1 together with examples of what they can mean in an SoS context.

SoS responsibilities and countermeasures

As always, it is important to have a clear division of responsibility between the SoS level and the CS. The SoS should provide a secure SoS architecture, in particular a secure communication system. It should also provide fundamental security requirements on the CS, and protect the rest of the SoS against a compromised CS. The countermeasures that can be employed include:

- *Identity management*: This ensures that every CS is who it claims to be, and the solution can be either centralized, where the SoS implements its own mechanism for handing out privileges, or it can be distributed, where the CS owners agree to trust each other, so if one organization has handed out privileges to a CS, it holds for the SoS as a whole.

- *Communication architecture*: This can also be centralized or distributed. In a centralized architecture, there is a specific server that communication passes through, such as a message broker, and efforts can then be concentrated on securing this. The CS only connect to this

Table 1. The STRIDE threat model.

Threat	Security objective	SoS example
Spoofing of user identity	Authenticity	Adversary gains access to user login information, e.g. to an SoS communication system, and pretends to be a CS.
Tampering	Integrity	Adversary injects incorrect data into an SoS, e.g. by sending messages in the SoS communication system.
Repudiation	Non-repudiability	Adversary intrudes the SoS without being traced.
Information disclosure	Confidentiality	Adversary gets access to critical information without spoofing, e.g. by CS sending data unencrypted.
Denial of service	Availability	Adversary makes the system unusable, e.g. by flooding the SoS communication system with messages.
Elevation of privilige	Authorization	Adversary poses as a central entity in a directed or acknowledged SoS, sending orders to other CS's.

server, and not directly to each other. In a distributed architecture, the CS establish direct connections to each other.

- *Firewalls*: The communication should be protected with firewalls for the external interfaces, but also for input from CS since they could have been compromised.
- *Message encryption*: Communication channels should use encryption to make it more difficult for intruders to get access to useful information.
- *Role based privileges*: As an example, a CS that is part of a certain constellation within the SoS should be able to exchange messages with others within that constellation, but it is not necessarily the case that other CS in the larger SoS should have access to that communication.

Constituent system responsibilities and countermeasures

The CS owners need to protect its non-SoS related functionality, implement the SoS security requirements, and ensure that changes made in preparing it for an SoS are secured. The available countermeasures include:

- *Firewalls*: Just as for the SoS, the CS may use firewalls for self-protection to limit input from the communication system.
- *Input validation*: The adversary could try to manipulate the behavior of a CS by sending fake data, and if possible, all inputs should therefore be validated against secondary sources, such as own sensors. If this is not possible, CS need to use received data cautiously.
- *Encryption*: Internal data of a CS can be encrypted, to make it more difficult for an intruder to extract data if it manages to get into the system. Also, software algorithms can be protected in similar ways, or at least have checksums so that manipulation of the software can be detected.
- *Rapid updates*: Whatever measurements are taken, there will always be a risk that vulnerabilities remain. The ability to update software rapidly is thus a key to sustaining a secure SoS.

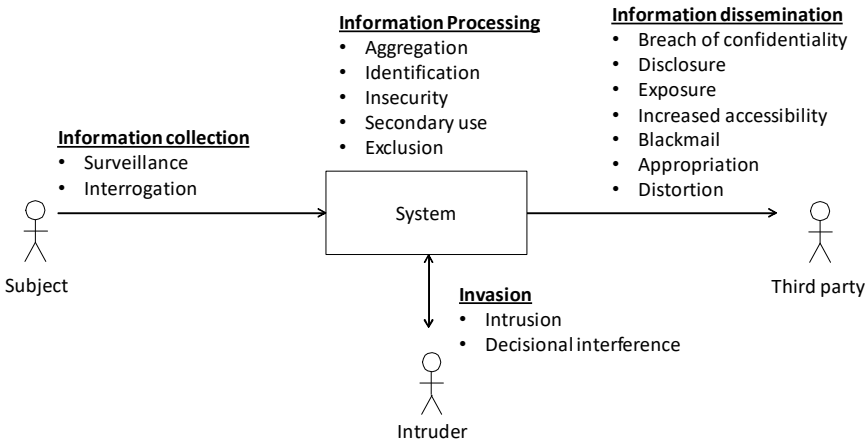

Figure 4. Solove´s taxonomy of privacy in information systems.

Privacy

Information privacy is, broadly speaking, the right to have control over how one's personal information is collected and used. Personal information is any information that can be connected, directly or indirectly, to a living person. A few examples (out of many more) are: Name, address, telephone number; photos, sound recordings; health, economic, and social status; and, depending on circumstances, a car registration number, or an IP address. Privacy is a socially created need and without society, there would be no need for privacy. It is an ethical and legal issue and is highly subjective.

Taxonomy

Just as for security, there are many aspects of privacy, and a taxonomy has been proposed by Solove (2006), which is shown in Figure 4. It introduces four main categories:

- *Information collection*. Getting data about a person into the system by surveillance where a person's activities are recorded; and interrogation which is about probing for information.

- *Information processing*. Deals with the system's data usage, and includes combining different data points; dealing with identification of the person behind the data; insecure handling of the data; using data for other purposes than for which the user has given consent; and exclusion, where the subject is not informed about the data available about that person.

- *Dissemination*. Allows a third party to get access to the subject's data, and includes spreading data that has been promised to remain secret; disclosing truthful information about a person which can make other persons judge the subject differently; exposure of data in a way that is uncomfortable for that person; making data about the subject easier to get at; making it possible to threat the subject with dissemination, such as blackmailing; appropriation, which is using the subject's personal information for the benefit of others, such as using a portrait of a person in commercials without their consent; and distortion where false or misleading information is disseminated.

- *Invasion*. This category differs from the others in that it does not necessarily include personal information, but it can consist of an intrusion which disturbs one's solitude or tranquility, or an interference by the intruder into the subject's personal decisions.

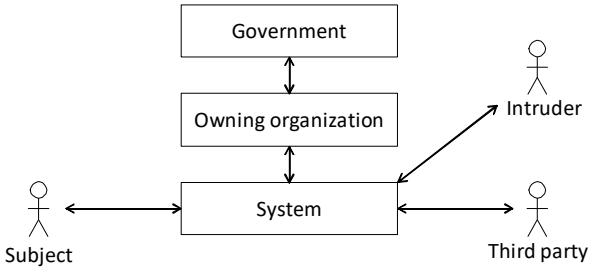

Figure 5. Modeling privacy as a control system.

Legislation

To protect the rights of individuals, governments often impose legislation that controls how an organization can design and operate systems that deal with personal data. In terms of a control diagram, this introduces two additional levels of control, as illustrated in Figure 5:

1. *Owning organization*, which should control the system in such a way that privacy is ensured.
2. *Government* that imposes legislation and sanctions on the owning organization.

A widely known legislation is the European Union's General Data Protection Regulation (GDPR), which was introduced in 2018. It applies to all companies in the world with clients in EU. One of the aspects of this regulation that has caused it to be taken seriously is that it can yield considerable fines to companies if not followed, in amounts to up to 20 M€ or 4% of a company's annual turnover.

A company is required to obtain active consent of the subject for dealing with its personal data. There are however exceptions to this, such as if the information is required by law, or if it is obvious to the client that this data must be handled as part of the service. Further, a subject has the right to access its personal data, and can demand that it is erased which is known as the "right to be forgotten". All data breaches must be reported immediately to authorities. If the organization is dealing with sensitive data, it must appoint a data protection officer with obligations to ensure privacy.

Privacy by design

GDPR requires that privacy is designed into the system from the start. Two commonly used techniques for privacy protection are:

- *Pseudonymization*: Identifiers of a person are substituted by a consistent value. All references to the same person get the same pseudonym, but name and other information is removed.
- *Anonymization*: Identifiers of a person are removed, including indirect references that can allow someone to deduce who the person is from the data.

With pseudonymization, it can still be inferred that a set of data records relate to the same person, since they use the same pseudonym, but with anonymization, no such links can be established.

To this should be added many of the security protection measures mentioned in the previous section, such as encryption of personal data, which can also help in protecting against some privacy breaches.

SoS implications

The above descriptions of privacy have some strong implications on SoS that deal in any way with personal data. The objective of an SoS is to create some emergent property which is achieved in part

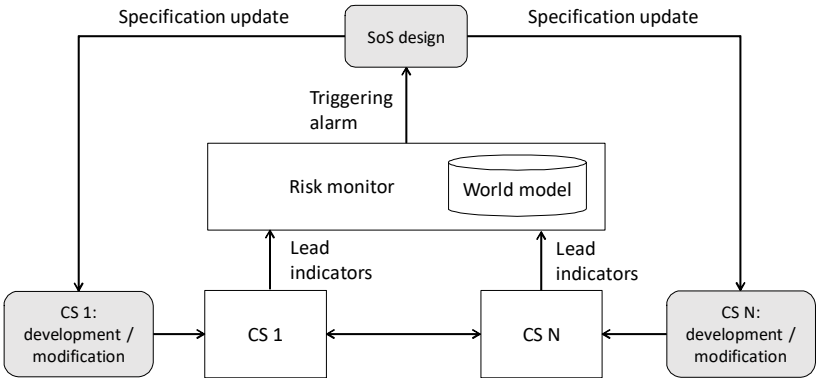

Figure 6. Dynamic risk management.

by combining data and knowledge from the different CS. Separate data items that were previously considered harmless or anonymous when viewed in isolation, can become sensitive and attributable to a person when combined, as shown by e.g. De Montjoye et al. (2015).

The legislation also provides difficulties for SoS due to requirements on consent for data usage. In many cases, an individual would not consider himself/herself to be a user of the SoS, but rather of one of the CS. This means that it is difficult for an SoS to ask for consent, but it must be done by a CS. However, this would make other CS within the SoS third parties, and a CS would therefore need to obtain the user's consent for such sharing within the SoS. This consent is required by legislation to be short and understandable to users and explaining the relations within the SoS can be challenging.

Dynamic approaches to risk management

Traditionally, risk analysis is carried out prior to putting a system in operation, and this makes sense since in theory this would allow hazards to be taken out of the system before use. The analysis is based on descriptions and models of the system during its development. However, such approaches have several problems that are getting apparent in today's increasingly complex and dynamic SoS:

- *Scalability*: The procedures used for risk analysis, including STAMP and its predecessors, are primarily manual. As complexity grows, they consume a lot of time and it is also quite hard for humans to grasp the consequences of e.g. multiple events that can lead to a hazard.

- *Evolution*: An SoS does not come fully formed from the beginning but will change during its lifetime. Therefore, risk analysis must be continuous, which adds to the burden, and also increases the possibility that it gets neglected. Nowadays, it is common practice to update systems very rapidly, even with daily or hourly updates of software, and it is hard to see how to incorporate static risk analysis into this way of working.

- *Validity*: Since the analysis is carried out on a model of the system, there is a risk that it only verifies that the system has a correct specification, but it could have been built differently.

These arguments do not necessarily mean that manual, pre-operational analysis practices should be abandoned. Rather, they need to be complemented with other approaches that allow for increasing automation of analysis; that can cope with rapid changes; and that work on the real system.

A potential framework for dynamic risk management of SoS is outlined in Figure 6. The basis is a continuous monitoring of the CS to collect operational data which is relevant to hazards. This data

constitutes a world model (or "digital twin") of the system in operation, and it can be compared to the expected values assumed during a static risk analysis.

By analyzing the world model continuously, situations that fall outside the normal boundaries can be detected and trigger the SoS owners to take action. If the analysis of this alarm indicates that there is an anomaly, the SoS design can be revisited leading to new CS requirements that get implemented. Thereby, a hazardous situation can be identified and removed before it leads to a loss.

This approach resembles the now popular DevOps process used in IT systems. It aims at bringing development (Dev) and operations (Ops) closer together to increase productivity and quality of software systems. However, it requires the possibility to very rapidly update systems, and a similar improvement is also needed in SoS. The loop time from the triggering alarm to an update having been implemented across CS must be much faster than is possible today, and this puts requirements on the architecture of CS to allow rapid software updates (Axelsson and Kobetski 2013; 2014).

The dynamic approach also supports an increased use of quantitative data, and this is to some extent also mentioned by Rasmussen (1997), although his work was carried out long before it became technically feasible to do this on a massive scale. By collecting run-time data, the frequency of different events can be calculated with great accuracy, and it can be detected when those frequencies change, indicating that the system or its environment has evolved in some way. Leveson (2015) also suggest monitoring and uses the term "leading indicators" to refer to the safety key performance measures. However, she does not discuss automated procedures for this, but rely on manual work.

Related work

SoS risk analysis. A number of authors have identified the importance to SoS of risk management, which is defined by Gorod et al. (2008) as to "monitor, identify, assess, analyze, and mitigate risk encountered in the SoS". Pinto et al. (2012) identify the foundations of risk management for SoS to be undesirable consequences; uncertainty; and temporal domain. This leads to seven guiding questions for risk management. Uday and Marais (2015) suggest tactics to improve SoS resilience, and Cavallo and Ireland (2014) suggest systems thinking to deal with complex interdependent risks.

SoS safety analysis. Harvey and Stanton (2014) derive and exemplify ten key challenges to SoS safety. A risk model for SoS, with a focus on safety, is proposed by Aitken et al. (2011). It emphasizes the need to focus on SoS related risks, and not just any risk associated with a CS. The model is described as a fault tree, where four generic regions at different levels of the tree are identified. Alexander et al. (2004) characterizes different SoS failure types related to goals and communication. Lock (2012) proposes a methodology to support non-specialist end users in the identification, organization and discussion of information required to manage SoS evolution and uses a modified form of HAZOPS (Hazard and Operability Study) to analyze the associated risks of evolution.

SoS security. Bodeau (1994) is an early yet insightful account of SoS security engineering principles and processes. Dahmann et al. (2013) also analyze security in SoS from a process perspective. Balasubramaniam et al. (2009) discuss identity management in an SoS. Hafiz et al. (2007) discuss some common security patterns and organize them according to the STRIDE model. Young and Leveson (2014) present the use of STAMP for security analysis.

SoS privacy. Work on SoS and privacy is scarce, but the application of STAMP to privacy is described by Shapiro (2016; 2017).

Dynamic risk management. Several SoS risk quantification methods have been proposed, on a very high level of abstraction. Zio and Ferrario (2013) analyze risks of a plant exposed to external events such as earthquakes. The focus is on establishing probabilities of events, to support Monte Carlo simulations. A model-based approach based on Bayesian Belief Networks with Monte Carlo simu-

lations is suggested by Kinder et al. (2015). Some approaches focus on the dynamic nature of the SoS, such as Guo and Haimes (2016) which suggests the use of precursor analysis, which are early signs of system failure. The need for dynamic risk assessments in SoS is also argued by Aitken et al. (2010). An early idea to improving the efficiency and effectiveness of STAMP analysis by using formal methods has been presented by Yang et al. (2019).

Conclusions

This paper studied the use of system theoretic models as a common framework for risk analysis in SoSE. It applied approaches with their roots in the safety domain and extended them to also deal with security and privacy, while identifying concerns that are specific to SoS. By providing a unified method, it becomes possible to also deal with trade-offs between different types of risks, which is a necessity in practice. The conclusion is that systems theory provides a good basis for dealing with SoS risk, but also that the complex and evolving nature of SoS requires present methods to be complemented by dynamic approaches which our future research will investigate and test further.

References

Ablon, L. 2018, *Data Thieves: The Motivations of Cyber Threat Actors and Their Use and Monetization of Stolen Data*. RAND Corporation report.

Aitken, J. M., Alexander, R. and Kelly, T. 2010, 'A case for dynamic risk assessment in NEC systems of systems', in *5th Intl. Conf. on System of Systems Engineering*. IEEE, pp. 1–6.

Aitken, J. M., Alexander, R. and Kelly, T. 2011, 'A risk modelling approach for a Communicating System of Systems', in *2011 IEEE International Systems Conference*. IEEE, pp. 442–447.

Alexander, R., Hall-May, M., and Kelly, T. 2004, 'Characterisation of Systems of Systems Failures,' in Proc. *22nd Annual System Safety Conference*, Unionville, VA.

Axelsson, J. 2015, 'A Systematic Mapping of the Research Literature on System-of-Systems Engineering', in *IEEE Systems of Systems Engineering Conference*, San Antonio, Texas.

——— 2019a, 'A Refined Terminology on System-of-Systems Substructure and Constituent System States', in *IEEE Systems of Systems Engineering Conference*, Anchorage, Alaska.

——— 2019b, 'Game theory applications in systems-of-systems engineering: A literature review and synthesis', in *17th Conference on Systems Engineering Research (CSER)*, pp. 154-165.

Axelsson, J. and Kobetski, A. 2013, 'On the conceptual design of a dynamic component model for reconfigurable AUTOSAR systems', ACM SIGBED Review, 10(4), pp. 45-48.

——— 2014, 'Architectural Concepts for Federated Embedded Systems', in *European Conference on Software Architecture – Workshops*, Vienna, Austria.

——— 2018, 'Towards a risk analysis method for systems-of-systems based on systems thinking', in *IEEE Systems Conference*, Vancouver, Canada.

Ashby, W. R. 1956, *An Introduction to Cybernetics*. London: Chapman & Hall Ltd.

Balasubramaniam, S. *et al.* 2009, 'Identity management and its impact on federation in a system-of-systems context', in *3rd Annual IEEE Systems Conference*, pp. 179–182.

Bianchi, T., Santos, D. S. and Felizardo, K. R. 2015, 'Quality Attributes of Systems-of-Systems: A Systematic Literature Review', in *3rd Intl. Workshop on Software Engineering for Systems-of-Systems*.

Bodeau, D. J. 1994, 'System-of-systems security engineering', in *Tenth Annual Computer Security Applications Conference*. IEEE Comput. Soc. Press, pp. 228–235.

Cavallo, A. and Ireland, V. 2014, 'Preparing for complex interdependent risks: A System of Systems approach to building disaster resilience', *Intl. Journal of Disaster Risk Reduction*.

Dahmann, J. *et al.* 2013, 'Security engineering in a system of systems environment', in *2013 IEEE International Systems Conference (SysCon)*. IEEE, pp. 364–369.

De Montjoye, Y., Radaelli, L., and Singh, V. K. 2015, "Unique in the shopping mall: On the reidentifiability of credit card metadata." *Science* 347(6221): 536-539.

Gorod, A., Sauser, B. and Boardman, J. 2008, 'System-of-Systems Engineering Management: A Review of Modern History and a Path Forward', *IEEE SYSTEMS JOURNAL*, 2(4).
Guo, Z. and Haimes, Y. Y. 2016, 'Risk Assessment of Infrastructure System of Systems with Precursor Analysis', *Risk Analysis*, 36(8), pp. 1630–1643.
Hafiz, M., Adamczyk, P. and Johnson, R. E. 2007, 'Organizing Security Patterns', *IEEE Software*, 24(4), pp. 52–60.
Harvey, C. and Stanton, N. A. 2014, 'Safety in System-of-Systems: Ten key challenges', *Safety Science*, 70, pp. 358–366.
ISO 2009, *Risk management – Principles and guidelines,* ISO 31000:2009.
Kinder, A., Henshaw, M. and Siemieniuch, C. 2014, 'System of systems modelling and simulation - an outlook and open issues', *Intl. J. of System of Systems Engineering*, 5(2), pp. 150–192.
Klein, J. and van Vliet, H. 2013, 'A systematic review of system-of-systems architecture research', in *9th Intl. Conf. on Quality of software architectures*. New York.
Kohnfelder, L. and Praerit, G. 1999, 'The threats to our products', Microsoft report.
Leveson, N. 2011, *Engineering a Safer World: Systems Thinking Applied to Safety*. MIT Press.
Leveson, N. 2015, 'A systems approach to risk management through leading safety indicators', *Reliability Engineering & System Safety*, 136, pp. 17–34.
Lock, R. 2012, 'Developing a methodology to support the evolution of System of Systems using risk analysis', *Systems Engineering*, 15(1):62–73.
Pinto, C. A., McShane, M. K. and Bozkurt, I. 2012, 'System of systems perspective on risk: towards a unified concept', *International Journal of System of Systems Engineering*, 3(1), pp. 33–46.
Rasmussen, J. 1997, 'Risk management in a dynamic society: A modelling problem', *Safety Science*. Elsevier Sci B.V., pp. 183–213.
Redmond, P. 2007, 'A System of Systems Interface Hazard Analysis Technique', MSc thesis, Naval Postgraduate School, Monterey, California.
Shapiro, S. S. 2017, 'Addressing Early Life Cycle Privacy Risk: Applying System-Theoretic Early Concept Analysis and Model-Based Systems Engineering to Privacy', in *International Workshop on Privacy Engineering*.
Shapiro, S. S. 2016, 'Privacy Risk Analysis Based on System Control Structures: Adapting System-Theoretic Process Analysis for Privacy Engineering', in *2016 IEEE Security and Privacy Workshops*, pp. 17–24.
Solove, D. J. 2006, 'A Taxonomy of Privacy', University of Pennsylvania Law Review, 154(3).
Uday, P. and Marais, K. 2015, 'Designing Resilient Systems-of-Systems: A Survey of Metrics, Methods, and Challenges', *Systems Engineering*, 18(5), pp. 491–510.
Yang, P., Karashima, R., Okano, K., and Ogata, S. 2019, ' Automated inspection method for an STAMP/STPA - Fallen Barrier Trap at Railroad Crossing', Proc. *23rd Intl. Conf. on Knowledge-Based and Intelligent Information and Engineering Systems*.
Young, W. and Leveson, N. G. 2014, 'An integrated approach to safety and security based on systems theory', *Communications of the ACM*. ACM, 57(2), pp. 31–35.
Zio, E. and Ferrario, E. 2013, 'A framework for the system-of-systems analysis of the risk for a safety-critical plant exposed to external events', *Reliability Engineering & System Safety.*

Biography

Jakob Axelsson received an MSc in computer science in 1993, followed by a PhD in computer systems in 1997, both from Linköping University, Sweden. He was in the automotive industry with Volvo and Volvo Cars 1997-2010. He is now a full professor of computer science at Mälardalen University, Sweden and a senior research leader in systems-of-systems at RISE Research Institutes of Sweden. His research interests are focused on all aspects of systems-of-systems engineering, and in particular system architecture. Prof. Axelsson is a member of INCOSE and has served as chairman of the Swedish chapter.