The Need for SoS Safety Cases

Rob Alexander, Tim Kelly, George Despotou; University of York, York, UK

Keywords: System of Systems, safety cases

Abstract

When you create a System of Systems (SoS), you are doing wilful design. It follows that you need a safety case: a justification of why that system will be safe. All safety cases must have certain common properties: they must focus on risk, they must provide appropriate confidence in their claims, and they must have a clear relationship to a causal model of the system's safety behaviour. None of those are particularly easy for SoS, and there are several areas where SoS are particularly problematic, such as what exactly "the system" comprises, and what on Earth its lifecycle actually is. On the other hand, not everything about SoS safety is necessarily hard, and not every problem faced in SoS safety is an "SoS problem".

Motivation: have we got a case?

A System of Systems (SoS) is a system composed of components that are themselves systems, and that have their own goals and some degree of autonomy, yet still remain part of a whole with some shared goals and management. When you create an SoS, you are doing wilful design. If you define a configuration of assets to achieve certain aims, and agree that they will communicate and coordinate in certain specific ways, then you are doing explicit, conscious design. When you allow personnel within an SoS to adopt a pattern of using a certain configuration of assets to achieve certain aims, and habitually coordinate their actions by specific patterns of communication, then you are doing implicit, passive design. Either way, you are responsible for the consequences of those design decisions, and could be held accountable if they lead to an accident which causes loss of life.

SoS accidents do happen. There have been several accidents described in the literature that fit the title “Systems of Systems accident”. Examples include the Uberlingen mid-air collision (ref. 1), the Black Hawk fratricide (ref. 2, 3), and several fratricide incidents related to GPS use. At the same time, it seems that there is an ongoing trend towards creating more, larger and more coupled SoS, and to opening architectures so as to create multi-vendor SoS in place of traditional single-vendor systems. This should be a concern for everyone within the SoS and safety communities, and for anyone who is legally responsible for the actions of an SoS.

A safety case is an argument that some system is safe enough to operate, supported by evidence. There is much written about *explicit* safety cases, and some standards require them (e.g. Def Stan 00-56 (ref. 4)). However, if an operator of a system is convinced that some system is safe enough to operate, then they have an *implicit* safety case which contains their beliefs about the system's safety. It constitutes an account of why the system is safe that they can give to others, and indeed to themselves. Currently, few SoS have explicit safety cases, so it is worth giving some attention to the implicit cases that are being used. As researchers and consultants involved with industry and with military practitioners, we have encountered several forms of implicit case, which we will describe in the following paragraphs.

The first implicit case is that the system is safe because no serious accidents have occurred so far, and therefore no accidents will occur in the future. We may have been coordinating our fire control over some sensor fusion system for months and never yet called down fire on our own troops. This is some evidence of safety, but it is limited. It is very unlikely that we have sufficient operating hours to make a statistical claim of safety at the 1×10^{-6} per hour that typical standards would ask for. Even if we have, we may be vulnerable to small changes in circumstance, or to (reasonably rare) equipment failure, which will lead quickly to accident. For fifty years, system safety has made great strides by probing systems for hazards in every conceivable circumstance. Allowing safety to be assumed from track record alone discards these gains.

A second implicit case is to assert that every part of the system has its own, local, safety case, and that the overall SoS is therefore safe. Here, we are admitting that there are risks in the system, but argue that the existing cases take care of it; after all, if no part can have an accident then the SoS cannot have an accident. The main weakness here is that those local cases do not, and cannot, take the entire SoS context into account. Most of the creators of the local cases will have been unaware of the specific SoS context of use, and will have had to make assumptions in their local cases. It is likely that the SoS context violates at least some of those assumptions. In other words, the SoS context may erode the adequacy of the local cases, and the only way to assess the result is to check all of these assumptions and how they interact – in effect, to create an explicit SoS case. In any case, some of the risks in an SoS may not be visible at the level of the component systems – they may be *emergent*.

A third possibility is to admit that the overall SoS safety arrangements are inadequate. We have no SoS-wide analysis, we have no SoS-wide risk mitigations, so we have no adequate SoS safety cases. However, we can then claim that the system contains humans, and humans are intelligent beings. We can argue that our personnel know about the risks and take regular intelligent action to control them: they check electronic orders verbally, they recognise suspicious sensor readings and compare them with others, they hold off on the final fire action to give time for a check-fire to reach them. We can argue that humans have few hard upper bounds on what safety actions they can spontaneously take.

This is third implicit argument is stronger that the others, but it is not sufficient. Human prevention of accidents is often possible, but not necessarily reliable. Decades of human factors research has shown us the many factors that shape human performance, and therefore shape human management of accidents. We have a broad understanding of human behaviour in accident prevention, and it is irresponsible (an inadequate response to the accountability of an SoS operator) to ignore this information. Without it, we will frequently place personnel in situations where they are responsible for safety, but do not have the authority (or information) to make it safe.

We have now discussed three possible implicit arguments, none of which is adequate. We therefore propose that a safe SoS requires explicit SoS safety engineering and a corresponding explicit SoS safety case. In this paper, we review the requirements on an adequate SoS safety case.

The (explicit) safety case concept has a long history in the UK and achieved embodiment for military systems in the core Ministry of Defence equipment safety standard Def Stan00-56 (ref. 4). More recently, it has begun to make inroads internationally, under the guise of the assurance case, where it is applied beyond safety, to any number of non-functional properties – see, for example, the forthcoming software assurance standard ISO 15026, and the articles on security cases by Lipson, Goodenough and others (ref. 5, 6). The value of the safety case approach is that it challenges us to define all aspects of safety engineering and their outputs, and to look closely at the analysis we can do and the evidence we can generate.

Thus far, explicit safety cases have been mostly used for individual equipment items (e.g. this is UK MOD policy). There have been no attempts to require safety cases for large-scale SoS. Consequently, there is a lack of method and a lack of prior art to copy from. We can, however, define some requirements for SoS safety cases, drawing on two sources. First, SoS safety cases should meet the requirements and quality criteria identified for safety cases in other domains. Second, they should take account of the unique properties of SoS.

What makes a good safety case?

Kelly (ref. 7) identifies seven key ways in which a safety case can fail. Here, we draw out a few implications of that which are particularly relevant for SoS cases.

Must Focus on Risk

All safety cases are concerned with claiming that harm and loss will not occur. In order to do this we must manage specific risks; specific ways by which accidents could occur. Safety cases should, therefore, be structured around risk. This idea is widely accepted, and is a central part of Def Stan 00-56.

It is easy to identify the top-level risks in any SoS; we can quickly generate an exhaustive set of accidents from possible interaction between SoS elements and their peers in the environment. See Aitken et al (ref. 8) for some discussion of this, and a basic method for deriving it from models expressed in the Ministry of Defence Architecture Framework (MODAF).

Some clarity is needed here. For example, Bibby et al (ref. 9) provide the following set of "SoS safety accidents" for deployed military SoS:

- *"Fratricide: When allied forces fire at each other ('friendly fire') having mistaken them as the enemy due to poor combat identification.* [...]
- *Collision: When a vehicle, aircraft or boat crashes into any other object, vehicle, land or sea[,] unintentionally causing damage to itself and/or its occupants.*
- *Unintentional entry into hostile zone: When troops accidentally find themselves in an enemy occupied or dangerous area.*
- *Logistics and supply errors/Inadequate protection: Death, injury or damage resulting from errors from inefficient logistic supply or lack of equipment available.*
- *Collateral damage: Includes all unintentional damage to locations and people which are not the target, such as civilians."*

We can note that the above is a mixture of accident types (1 and 2), hazards (3), events that have causal relationship to loss but are some way from being legitimately called a hazard (4) and generic loss types (5). This lack of clarity is common in discussion of SoS, and may lead to expansion of the scope of "SoS safety engineering"; if this expansion is unchecked, such engineering may become impractical and ultimately ineffective.

A further aspect of risk is that it must be prioritised – given limited resources for analysis, understanding and mitigation, safety engineers should target the most serious risks first and with the greater part of their effort. Failure to prioritise was major criticism of the developers of the Nimrod safety case by Haddon-Cave (ref. 10) – they approached the safety case in systematic but un-prioritised fashion, without giving special attention to those parts that they knew were more dangerous. Haddon-Cave criticises the UK Ministry of Defence for accepting the (unfinished) Nimrod safety case, but in the presence of an active, aggressive regulator a prioritised approach to risk also makes strong business sense – it minimises the chance of an unexpected failure to certify.

Must be Product-based

We can create a safety case that argues that suitable processes have been followed, and that therefore a safe system will have been produced. This is not a good strategy – the resulting argument is very indirect, and therefore gives us little confidence that our claim of safety is true.

The core evidence that is used in a safety case should be direct – it should provide evidence about the specific system (e.g. software test results, design review, past incident rates) rather than indirect evidence about the processes used to create and manage it. See Hawkins in (ref. 11) for an expansion of this point. Direct evidence supports claims that some specific hazardous phenomena will not occur.

Safety engineering and management processes are important, but they have only a supporting role – their use provides confidence that particular direct evidence is valid (for example, good software testing processes will increase your confidence that your software test results are trustworthy).

Following the *assured safety argument* approach described in (ref. 12) forces this issue, as the main *safety argument* may only refer to a causal model of the system itself; only the separate *confidence argument* may refer to the engineering and management processes used.

Must Provide Appropriate Confidence

When creating a safety case, we can make any claim we wish, but without supporting evidence that claim is worthless. As well as a convincing structure for our safety argument, we need adequate confidence that each claim holds.

One corollary of that is that we need adequate research into the confidence provided by sources of evidence; if we use a testing technique, a trials regime, or a particular notation for encoding our system design, then we need research into how effective those techniques are. This is of interest for SoS because there is limited research into methods and techniques.

Must be Related to a Causal Model of the System

If we are going to create a safety case, it must be closely related to a causal model of the system (see Hawkins et al (ref. 12)). This allows claims about causation (e.g. a claim that a certain safety function will prevent a given hazard) to be checked. There are two types of methods that matter here – ways of generating causal models from system models, and ways of deriving risk assessments (e.g. hazard probabilities) from the causal model itself. We should only have confidence in our safety case if both of these are sound.

Must be Tractable, Understandable and Reusable

A safety case must be tractable, understandable and reusable. It is easy for a safety case to become a huge write-only artefact, created to meet a regulatory need and then abandoned. Kelly (ref. 7) calls this *"safety case shelf-ware"*, and Haddon-Cave (ref. 10) noted that the Nimrod safety case had such properties. This is inefficient, and indeed actively dangerous because it can provide a false sense of security that everything "has been made safe". An effective safety case must be a *living* part of an effective safety management system.

The Goal Structure Notation (GSN) was originally developed to make prose safety cases more tractable and manageable (ref. 13). Merely using a graphical notation, however, is not sufficient to make a case comprehensible, and having a complex graphical case with many nodes in the diagram proves nothing. This is *"the illusion of pictures"* in Kelly (ref. 7).

What makes a good *SoS* safety case?

The trick in understanding the *specific* requirements of SoS safety cases is to look at the characteristics that distinguish SoS from "mere systems". For each requirement, we need to draw out its implications and suggest compensatory measures. The adequacy of these compensatory measures will, to a large extent, determine the quality of our safety case.

The following subsections discuss the characteristics that matter the most for SoS safety cases. Section 0 then reviews those characteristics which are not unique to SoS and so should not be treated as such.

Must Appropriately Bound the SoS

There is a longstanding complaint that the boundary used for systems safety modelling is often too narrow (e.g. see Leveson (ref. 14), Rasmussen (ref. 15), Hollnagel and Woods (ref. 16)). Obviously, when we decide where to place the model boundary, we implicitly decide what model dynamics we will get. Yet it is not clear where we should draw the line – the boundary of SoS can be amorphous and there are a great many lines that could be drawn. Leveson has included governments agencies in some of her STAMP models (e.g. in (ref. 17)).

It is important that SoS safety cases concern themselves specifically with hazards that result from interactions within the SoS, and relegate hazards stemming solely from a single system's behaviour to the safety case of that individual system. Similarly, it is important to bound the SoS being analysed such that it is tractable – the call to model SoS is not a call to model the whole universe. By deciding to do "SoS safety cases" we of course don't magically gain more resources – most likely, we'll have to retarget existing safety resources. We need to make our SoS analysis count.

What is the "product" that an SoS case should focus on? There is no convenient metal boundary as there is for an aircraft or ship. Perhaps the best option is to analyse a well-defined *capability* – a capability that we know

can cause harm (e.g. to a specific example of networked-supported kill chain capability, as discussed by Caseley et al in (ref. 18)). This will lead us to a network of interacting nodes or agents – but only to the aspects of those nodes or agents that relate to the capability in question. This may reduce our effort significantly.

In the UK military, safety case creation is part of equipment acquisition. This is sensible in many cases (when major vehicles or weapon systems are acquired) but is of little value for systems that are only safety-significant as part of a larger SoS (such as communications equipment or information-related software). It is impossible in those latter cases to understand enough of the context of use (all possible contexts of use) to reason adequately about the safety of the system. An SoS approach provides an opportunity to produce safety cases at a level where they can be effective, i.e. at a level where enough context is known to connect supporting communications and information systems to the systems that can directly cause harm.

Must Take Account of the Role of Humans in the Systems

It is recognised that most SoS are sociotechnical systems and include the humans that manage it and participate in it. SoS cannot be adequately analysed or engineered from a purely technical perspective – the human dimension must be thoroughly managed. This is a contrast with some traditional safety engineering, which has tended to enforce a system/operator dichotomy, but it is neither entirely new nor specific to SoS: the need for a sociotechnical view is recognised by Qureshi (ref. 19), Leveson (ref. 14), Hollnagel and Woods (ref. 16), none of whom are explicitly writing about "SoS". Keating et al. suggest in (ref. 20), however, suggest that current SoS engineering work overemphasises the technical at the expense of these other factors.

For military SoS, and some other domains (such as the healthcare and the emergency services), we have a problem in that prevention of harm cannot be fully proceduralized – there are many cases where "unsafe" decisions are required in order to prevent unwanted losses. The classic examples involve taking safety risk order to prevent near-certain enemy risk, e.g. by calling down artillery fire without adequate knowledge of friendly movements, justified by your certainty of strong enemy forces in the area who are very likely to cause harm if you do not fire.

Thus, during SoS safety case development, we have to somehow 'bracket' around the human – we have to create a safety case that remains valid even when the goals and intentions of the humans involved change. One approach to this is to evaluate the safety case based not on its absolute achievement of safety but on its ability to support safe behaviour if the humans involved want that. In other words, we need to show that operators can maintain control of key safety constraints, in the manner recommended by Rasmussen in (ref. 15).

If we are going to achieve this, we need to work *with* human participants rather than against them – we need to provide the information and control that they need in order to make safety-related decisions and safety-related trade-offs. Much of an SoS safety case will thus involve claims that operators will be able to intelligently and safely manage the system. In other words, they will need to argue that they won't introduce anything that they won't be able to control.

One caveat here is that doing "SoS safety" does not necessarily mean that we need to consider government, culture or other large-scale sociological systems. Although SoS require a sociotechnical approach on one level, it is not necessarily true that they are more affected by these wider than smaller systems are. The decision of what higher-level containing systems to consider is an engineering decision that needs to be made regardless of the level or scale of the core system being analysed.

Must Take Account of Negative Emergence

SoS are complex, in that their overall behaviour will emerge from diverse interactions between their components. If not explicitly managed, some of those interactions will cause unwanted behaviours – negative emergence. There is a commonality here with common-cause failures in simpler systems, and there will be gains from applying similar techniques at the SoS level, but we cannot expect them to find all such cases.

The complication here is that we want to achieve safety affordably, and that means subdividing SoS components in a modular way as much as possible. We need to decompose SoS into small, tractable parts, but we need to be

aware of the aspects that are not actually decomposable in that way. There is a strong human factor here – a human can learn to rely on a system that you didn't think they needed on (indeed, that you didn't even think they had access to).

Where techniques exist that have the potential to reveal negative emergence, they should be used. Simulation-based techniques have significant potential here (e.g. see (ref. 21). In practice, however, the solution will rely heavily on online incident reporting systems, and on monitoring of safety proximity measures (literal ones like air proximity, or more abstract ones such as the time between designating an invalid target and a check fire being called). Incident reporting and investigation must take on greater importance as the system we analyse increase in complexity, and when we move to an SoS view we are increasing the complexity of "the system" that we are analysing.

It is important to maintain a sense of perspective here – we cannot provide perfect analysis, particularly in military environments, so we should not seek to do so. What can, and should, do much of our safety activity at a level and perspective where we can effectively take account of SoS interactions and therefore improve aspects of safety that can only be understood at that levels. In the UK, the ALARP principle may make this legally possible, although it is more difficult to justify limiting analysis (e.g. by not performing certain analyses at the SoS level) than it is to justify limiting ensurance (e.g. by not implementing some expensive safety function). There is work on the *ACARP* principle ("As *Confident* As Reasonably Practicable") that may be useful here (see Hawkins in (ref. 11) for more on ACARP, albeit in the context of software only).

Must Allow for a Non-equipment Lifecycle

Keating et al state in (ref. 20) that for SoS "*it is naïve to assume that 'one size fits all' and a singular n-step process can be successfully applied to any complex system problem context with an equivalent probability of success"*. They propose, instead, that the focus be on engineering 'methodologies' as defined by Checkland in (ref. 22): *"… not* a method *but a set of* principles of method *which in any particular situation have to be reduced to a method uniquely suitable to that particular situation.*" This has implications for the value of any explicit SoS lifecycle or engineering process – inevitably, steps or phases will be performed in an order, or with repetitions, that cannot be easily foreseen in advance. It may be better to abandon the idea of a "lifecycle" for SoS entirely, and instead treat SoS engineering activities as *responses to triggers for action* – an example of such a trigger would be the introduction of a new entity type into the SoS.

Must Deal with Tradeoffs

Trade-offs between desirable attributes are a fact of life in any engineered system. SoS, however, can make them particularly difficult to do. This is largely a side-effect of complexity – of having a great many component systems which are coupled in diverse ways. As a result of this coupling, changes in one place may cause changes in another place. As Raheja and Moriarty observe in (ref. 23) *"Systems today are connected directly and indirectly with many other systems. A typical weapon system is linked to other systems, vehicles, soldiers and satellites in almost unlimited interactions. Tweaking in one place is bound to create some change in another place."*

Brooker's paper on Uberlingen (ref. 1) discusses a form of this problem – he emphasises the role of the TCAS collision avoidance system in preventing accidents as well as causing them (and rightly criticises those safety professionals who don't acknowledge this). A nastier extreme is the tradeoff of life against life in the military, emergency services and healthcare cases – as we noted earlier, failure to perform can cost in the same currency as accidents do (if the system does not perform, them people will die). For healthcare, this issue is tackled in Brennan et al (ref. 24) who call for less emphasis on accidents and more emphasis on quantitative measures of overall care effectiveness.

It is difficult to do this if only safety has a developed case. It may be that this will ultimately require *dependability* cases to be developed for SoS, which argue explicitly about varied attributes and their trade-offs (ref. 25). This of course has its own cost implications, and there is little industry experience with producing dependability cases. Further work is needed – in particular, further practical experience of the dependability case concept is needed before we can advance it as a solution to the trade-off problem.

Must Take Account of Past SoS Accidents

A final, crucial requirement for SoS safety cases is that they must take account of past SoS accidents. This is an indirect "meta-requirement" – SoS must use *thinking and methods* that take account of the insights we have gained from past SoS accidents.

Past accidents provide something real for us to measure ourselves against – if nothing else, we can try to check whether our methods would likely have preventing such accident. If they don't then our behaviour is suspect; in effect, we are denying empirical feedback of a very definite kind. That is foolish (and potentially negligent).

By definition (see Aitken et al in (ref. 8)), SoS accidents emerge from interactions between component systems. This may involve the classic composition of multiple failures (which is a common reason for non-SoS accidents) and the interaction of diverse non-failure behaviour (which again, is often part of accidents in all kinds of systems – see Perrow (ref. 26)).

Beyond investigations into individual accidents, there is great value in summarising work that draws together multiple analyses by multiple authors and comments on their validity and draws overall conclusions. Examples of this include Brooker (ref. 1), which draws some "Macro-Level Lessons for ATM" from the Uberlingen accident, and Caseley et al (ref. 18), which summarises a range of information on SoS-related fratricide accidents.

What is *Not* Unique to SoS?

In contrast to the previous section, it is important that we are clear about what is not unique to SoS safety, that we are clear about where SoS safety is just like the safety of a vehicle or the safety of a weapon. It is very easy to make "SoS safety" subsume all safety, because in a sense it does; making an SoS safe ultimately means that all other aspects of safety must have been practiced on it and its constituent systems. But treating these issues as if they uniquely belong to SoS just muddies the waters and obscures the SoS safety work that really matters.

For example, doing good safety engineering is not unique to SoS. Good SoS safety requires that we set the boundary in an intelligent way, appropriate to the control we have and ways in which accidents can occur; but that is true of non-SoS safety too. Good SoS safety requires that we look wider than "faults and failures" and consider how "normal" or "acceptable" behaviour by users or components could lead to accidents (see Leveson (ref. 14) and Qureshi (ref. 19)); but this is true of non-SoS safety too.

Similarly, good SoS safety requires that we acknowledge the sociotechnical nature of most systems and take steps to support their human operators and participants (see Rasummusen (ref. 15), Hollnagel (ref. 16)). But this is not unique to SoS.

As "SoS engineers" we should be modest, and restrict our claims to those areas where thinking about "the SoS" really makes a difference.

Longer term, we should probably move away from talking about "SoS safety", and instead make a more nuanced mapping of system properties to techniques and guidance. Classifications of SoS such as the (virtual, collaborative, acknowledged, directed) of the DOD SOSE guide (ref. 27) and the (mission-level, platform-level, IT-based) of Dahmann et al (ref. 28) may be important. However, we suspect that it will be more useful to map techniques and guidance to specific properties – identifying techniques, for example, for systems that have an open architecture, for systems that have only just been recognised as systems (and hence where the only safety engineering done so far is at the component level), and for systems that you know have a flexible composition and high component autonomy.

Who Should Produce SoS Safety Cases?

Realistically, SoS safety cases need to be produced and maintained by the system owner and operator. Obviously equipment suppliers and contractors have a role to play, but they cannot know enough detail of how

the system is being operated. A real-world SoS may change quite rapidly, especially in military domain, and this implies a new way of thinking about how safety cases are managed. More than ever, it is imperative that safety cases are live documents and are useful in operation.

Conclusions

We need SoS safety cases. They will need to be good safety cases by standards common to all safety cases: they need to be focussed on risk, they need to be product-based, and they need to provide appropriate, legitimate confidence, they need to be based on a causal model, and they need to be comprehensible.

SoS safety cases also need to take account of the unique challenges posed by SoS. This creates hard decisions for SoS safety case maintainers, decisions that need to be taken at the (operating) organisational level. Not least this means that SoS safety cases need to be owned and maintained by an operating organisation (not by a system developer, even if they are initially responsible for the delivery of the "the SoS"), and that they must be owned at a level where a wide view of the SoS is possible. In UK military terms, this suggests that the safety case is for a particular capability deployed in a particular theatre. This raises its own problems (not least that there will be overlap between the systems that provide different capabilities), but there are existing methods (e.g. modular safety cases) that may help here.

Creating SoS safety cases may help you do better safety overall; in some cases, the SoS level will be a much more natural level for doing safety work than the equipment level. The measure of success for SoS safety cases (and, indeed, any other safety engineering activity done at the SoS level) should be their costs versus their benefits. In the short term this can be estimated by the acceptance they achieve; in the longer term, this should be judged by their impact on SoS safety management.

References

1. P. Brooker, The Überlingen accident: Macro-level safety lessons, Safety Science, vol. 46, pp. 1483-1508, 2008.
2. N. Leveson, P. Allen, and M.-A. Storey, The Analysis of a Friendly Fire Accident using a Systems Model of Accidents, in Proceedings of the 20th International System Safety Society Conference (ISSC 2002), 2002.
3. S. A. Snook, Friendly Fire: the Accidental Shootdown of U.S. Black Hawks Over Northern Iraq. Princeton, New Jersey: Princeton University Press, 2000.
4. MoD Interim Defence Standard 00-56 Issue 4 - Safety Management Requirements for Defence Systems, Ministry of Defence, 2007.
5. J. Goodenough, H. Lipson, and C. Weinstock, Arguing Security - Creating Security Assurance Cases, 2007.
6. H. Lipson and C. Weinstock, Evidence of Assurance: Laying the Foundation for a Credible Security Case, 2008.
7. T. Kelly, Are 'Safety Cases' Working?, Safety Critical Systems Club Newsletter, vol. 17, pp. 31-33, 2008.
8. J. M. Aitken, R. Alexander, and T. Kelly, A Risk Modelling Approach for a Communicating System of Systems, in Proceedings of IEEE International Systems Conference, 2011.
9. S. K. Bibby, A. German, P. R. Symons, M. Spencer, and G. Wilkinson, Resilient and Dependable Systems of Systems: System of Systems Safety, in Proceedings of INCOSE, 2010.
10. C. Haddon-Cave, The Nimrod Review, 2009.
11. R. Hawkins and T. Kelly, Software Safety Assurance - What Is Sufficient?, in Proceedings of Proceedings of the 4th IET System Safety Conference, London, UK, 2009.
12. R. Hawkins, T. Kelly, J. Knight, and P. Graydon, A New Approach to Creating Clear Safety Arguments, in Proceedings of the Safety-critical Systems Symposium, 2011.
13. T. Kelly, Arguing Safety - A Systematic Approach to Managing Safety Cases, PhD Thesis, University of York, 1998.
14. N. Leveson, A New Accident Model for Engineering Safer Systems, Safety Science, vol. 42, pp. 237-270, 2004.
15. J. Rasmussen, Risk Management in a Dynamic Society: a Modelling Problem, Safety Science, vol. 27, pp. 183-213, 1997.

16. E. Hollnagel and D. D. Woods, Joint Cognitive Systems: Foundations of Cognitive Systems Engineering: CRC Press, 2005.
17. N. Leveson, M. Daouk, N. Dulac, and K. Marais, Applying STAMP in Accident Analysis, in Proceedings of the 2nd Workshop on Investigating and Reporting of Incidents and Accidents, 2003.
18. P. Caseley, D. Dean, J. Gadsden, and P. Houghton, Concepts of Network Enabled Capability - safety issues and potential solutions, in Proceedings of the 25th International System Safety Society Conference (ISSC 2007), Baltimore, USA, 2007.
19. Z. Qureshi, A Review of Accident Modelling Approaches for Complex Critical Sociotechnical Systems, Defence Science and Technology Organisation, 2008.
20. C. Keating, R. Rogers, R. Unal, D. Dryer, A. Sousa-Poza, R. Safford, W. Peterson, and G. Rabadi, System of Systems Engineering, Engineering Management Journal, vol. 15, pp. 36-45, 2003.
21. R. Alexander, Using Simulation for Systems of Systems Hazard Analysis, PhD Thesis, University of York, 2007.
22. P. Checkland, Systems Thinking, Systems Practice: John Wiley & Sons, 1999.
23. D. Raheja and B. Moriarty, New Paradigms in System Safety, Journal of System Safety, vol. 42, 2006.
24. T. Brennan, A. Gawande, A. Thomas, and D. Studdert, Accidental Deaths, Saved Lives, and Improved Quality, New England Journal of Medicine, vol. 353, pp. 1405-9, 2005.
25. G. Despotou, Managing the Evolution of Dependability Cases for Systems of Systems, PhD Thesis, University of York, 2007.
26. C. Perrow, Normal Accidents: Living with High-Risk Technologies. New York: Basic Books, 1984.
27. Department of Defense, Systems Engineering Guide for Systems of Systems: Department of Defense, 2008.
28. J. Dahmann, G. Rebovich Jr, J. A. Lane, R. Lowry, and J. Palmer, Systems of systems test and evaluation challenges 2010.

Biography

Dr Rob Alexander, Ph.D., Department of Computer Science, University of York, Deramore Lane, York, YO10 5GH, UK, telephone – +44 1904 325474, facsimile – +44 1904 325599, e-mail – robert.alexander@cs.york.ac.uk

Dr Alexander is a Lecturer in the High Integrity Systems Engineering (HISE) group in the Department of Computer Science at the University of York. Since October 2002 he has been working on methods of safety analysis for systems of systems and autonomous systems, with a particular emphasis on simulation and automated analysis. He is currently supervising research projects on certification of autonomous systems, dynamic risk assessment for systems of systems, and automated hazard analysis using simulation and metaheuristic search. Rob graduated from Keele University in 2001 with first class honours in Computer Science, and was awarded his doctorate in 2008 by the University of York.

Dr George Despotou, PhD., Department of Computer Science, University of York, Deramore Lane, York, YO10 5GH, UK, telephone – +44 1904 325466, facsimile – +44 1904 325599, e-mail – george@cs.york.ac.uk

Dr Despotou is a Research Associate in the High Integrity Systems Engineering (HISE) group in the Department of Computer Science at the University of York. His background is Dependability, particularly work on Dependable Systems of Systems (including rationalising tradeoffs amongst conflicting dependability objectives), Hazard Analysis for Enterprise Architectures, and evidence-based assurance and argumentation (including work on modular and incremental certification approaches, and vendor assessments).

Dr Tim Kelly, PhD., Department of Computer Science, University of York, Deramore Lane, York, YO10 5GH, UK, telephone – +44 1904 325477, facsimile – +44 7976 889545, e-mail – tim.kelly@cs.york.ac.uk

Dr Kelly is a Senior Lecturer in the Department of Computer Science at the University of York. His research interests include safety case management, software safety analysis and justification, software architecture safety, certification of adaptive and learning systems, and the dependability of 'Systems of Systems'. He has supervised a number of research projects in these areas with funding and support from Airbus, BAE SYSTEMS, Data

Systems and Solutions, DTI, EPSRC, ERA Technology, Ministry of Defence, QinetiQ and Rolls-Royce. He has published over 140 papers on high integrity systems development and justification in international journals and conferences. He is also Managing Director of Origin Consulting (York) Limited – a consultancy company specialising in safety-critical systems development and assurance.