Ensuring Dependable Systems of Systems

Robert Alexander, Martin Hall-May and Tim Kelly
University of York

Abstract

The emerging class of systems known as Systems of Systems (SoS) are composed of many distributed, heterogeneous and autonomous components. Such systems are typically separately designed and manufactured and evolve throughout their lifetime. However, they are expected to work together, often in safety-critical areas of operation such as civil transportation, military operations and space exploration. As such, the systems are required to interact in ways that do not result in accidents. However, hazards that can lead to accidents can arise from the interaction of any of the behaviours of the system , not just from explicit failures.

Given the complexity and dynamism of interactions within a network-enabled SoS it can be extremely hard to analyse the potential for interactions and communications between components to lead to unsafe behaviour. Performing hazard analysis on such systems is therefore challenging, in part because it is difficult to know in advance which of the many observable or measurable features of the system are important for maintaining system safety. Having discovered the potential for hazardous behaviour, it is then necessary to mitigate its effects. However, the nature of SoS configurations means that interactions between system components cannot be restricted by designed-in safety features used in many other systems. Therefore, system behaviour must often be managed and constrained operationally.

Describing the results of research performed within the High Integrity Real Time Systems (HIRTS) Defence and Aerospace Partnership (DARP) at York, this paper describes how a simulation and machine-learning based approach can facilitate hazard analyses of SoS. It also explains how a safety policy can be systematically derived and expressed in a structured fashion in order to constrain system behaviour within the boundaries of that which is deemed acceptably safe. Examples derived from battlefield scenarios and SoS configurations will be used to illustrate the approach presented in the paper.

1 Introduction

Large-scale military and transport Systems of Systems (SoS) present many challenges for safety. Attempts to define the term ‘SoS’ have been controversial - examples can be found in [1] and [2]. It is easy, however, to identify uncontroversial examples, Air Traffic Control and Network Centric Warfare being the most prominent. These examples feature

mobile components distributed over a large area, such as a region, country or entire continent. Their components frequently interact with each other in an ad-hoc fashion, and have the potential to cause large-scale destruction and injury. It follows that for SoS that are being designed and procured now, safety has a high priority.

A SoS is a complex multi-agent system (MAS) in which many entities have a mobile physical presence. The agents within this MAS are, in themselves, very complex. This complexity means that conventional system safety techniques are not adequate for identifying safety hazards in a SoS. In this paper, we propose multiagent simulation augmented by machine learning and agent tracing techniques as a viable alternative.

In order to ensure the safe behaviour of SoS, the behaviour of the individual system entities must be controlled, as must the overall behaviour that *emerges* from their individual actions and interactions. One way to achieve this is to impose a system-wide *safety policy*, which describes the rules of behaviour which agents in the system must obey. Due to the geographically distributed nature of many entities, the policy typically cannot be directly enforced by some external controller (as in security policy); rather, the entities must comply with it individually.

Section 2 of this paper elaborates on the problems for SoS safety. Section 3 shows how multiagent simulation can be used for SoS hazard analysis. Section 4 describes how an appropriate safety policy can be derived from the results of such hazard analysis. Section 5 presents an example of the technique in application, and Section 6 presents a summary and conclusions.

2 The Problem of SoS Safety

The Oxford English Dictionary [3] defines safety as *"The state of being safe; exemption from hurt or injury; freedom from danger."* In system safety engineering, it is common to restrict the definition of 'hurt or injury' to the physical injury or death of humans. For the purposes of this paper, we will restrict ourselves to this definition. It can be noted, however, that the approach presented can easily be expanded to cover alternative conceptions of safety, such as those including avoidance of material loss.

The problems faced by safety analysts when attempting to analyse SoS fall into three categories: the difficulty of performing hazard analysis, the restricted means by which safety features can be introduced, and the problem of 'System Accidents'. In their discussion of functional hazard analysis, Wilkinson and Kelly [4] note that these problems are present in conventional systems. The characteristics of SoS, however, exacerbate them.

2.1 Hazard Analysis

In a conventional system, such as a single vehicle or a chemical plant, the system boundary is typically well-defined and the components within that boundary can be enumerated. Once hazard analysis has been performed to identify events that may cause injury or death, safety measures can be

introduced and the risk of accidents computed from the probabilities of failure. Conventional techniques such as fault tree analysis are effective in this task.

In a SoS, the necessary hazard analysis is itself very difficult. When hazard analysis postulates some failure of a component, the effect of that failure must be propagated through the system to reveal whether or not the failure results in a hazard. The system boundary is not well defined, and the set of entities within that boundary can vary over time, either as part of normal operation (a new aircraft enters a controlled airspace region) or as part of evolutionary development (a military unit receives a new air-defence system). Conventional tactics to minimise interactions may be ineffective, because the system consists of component entities that are individually mobile. In some cases, particularly military systems, the entities may be designed (for performance purposes) to form ad-hoc groupings amongst themselves. Conventional techniques may be inadequate for determining whether or not some failure in some entity is hazardous in the context of the SoS as a whole.

It follows from this that a hazard analysis approach is needed which can reveal hazards caused by failure propagation through complex systems and that can consider the effect of multiple simultaneous failures.

2.2 Ensuring Safety

A purely functional design with no safety features is unlikely to be adequately safe. Therefore, design changes need to be made in order to reduce safety risk to acceptable levels. In a conventional monolithic system, there are many features that can be introduced to prevent or mitigate hazards; examples include blast doors, interlocks, and pressure release valves.

The SoS that are considered here contain many mobile agents with a high degree of autonomy. Such `hard' safety features are therefore not available. Consider, for example, air traffic control. If a controller wants to prevent a given aircraft from entering an airspace region (say, one reserved for an airshow) then he or she can instruct the aircraft to fly around it. The controller cannot, however, physically prevent the aircraft from flying into the region. (In a military scenario there are more drastic measures for dealing with aberrant agents, particularly if they are unmanned.)

Therefore, achieving safety in a SoS will rely to a large extent on responsible behaviour from the individual agents. In order to achieve this, agents need to know what behaviour is acceptable in any given circumstance. It follows from this that system designers and operators need to know how the agents in the system can safely interact.

2.3 System Accidents

Perrow, in [5], discusses what he calls ‘normal accidents' in the context of complex systems. His ‘Normal Accident Theory' holds that any complex, tightly-coupled system has the potential for catastrophic failure stemming

from simultaneous minor failures. Similarly, Leveson in [6] notes that many accidents have multiple necessary causes; in such cases it follows an investigation of any one cause *prior to the accident* (i.e. without the benefit of hindsight) would not have shown the accident to be plausible.

A SoS can certainly be described as a ‘complex, tightly-coupled system', and as such is likely to experience such accidents. It can also be noted that a ‘normal accident' could result from the combination of apparently safe, normal behaviours which are safe in isolation but hazardous in combination. Imagine, for example, a UAV that aggressively uses airspace and bandwidth under some circumstances. This may be safe when the UAV is operating on its own, but not when it is part of larger SoS.

It follows from this that a SoS safety analysis approach will need to be able to capture the effects of interactions between multiple simultaneous failures and normal agent behaviour.

3 Hazard Analysis Using Multiagent Simulation

Ferber, in [7] provides the following definition of multi-agent simulation: *“Multi-agent simulation is based on the idea that it is possible to represent in computerised form the behaviour of entities which are active in the world, and that it is possible to represent a phenomenon as the fruit of the interactions of an assembly of agents with their own operational autonomy.”*

Similarly, Ilachinski, in [8] offers *“[Multi-agent simulations] consist of a discrete heterogenous set of spatially distributed individual agents, each of which has its own characteristic properties and rules of behaviour.”*

Typically, the value of multi-agent simulation is asserted in comparison to the mathematical models that have traditionally been used in biology, economics and military analysis. Ferber notes that agent-based models allow the integration of quantitative variables, differential equations and symbolic rules into agent behaviour, thereby providing a means to exploit qualitative observations as well as quantitative information [7]. He also notes that such ‘micro-worlds' allow analysts to experiment by modifying agent behaviour and adding new agent types, which is not possible with high-level mathematical models. Most significantly for our purposes, Ferber comments that such simulations *“make it possible to model complex situations whose overall structures emerge from interactions between individuals”*.

Ilachinski, in [8] makes a similar point: in a multi-agent simulation, different levels of behaviour can be observed. Analysts can examine both the top-level emergent behaviour and the low-level interactions between individual agents. That is, the simulations can both predict overall behaviour and explain *why* it occurs. It can be seen that this relates to the concerns raised in section *problem* about the hazard analysis of SoS. On a more general level, Ilachinski also notes that working with multi-agent models gives a researcher an insight into the dynamics of the modelled

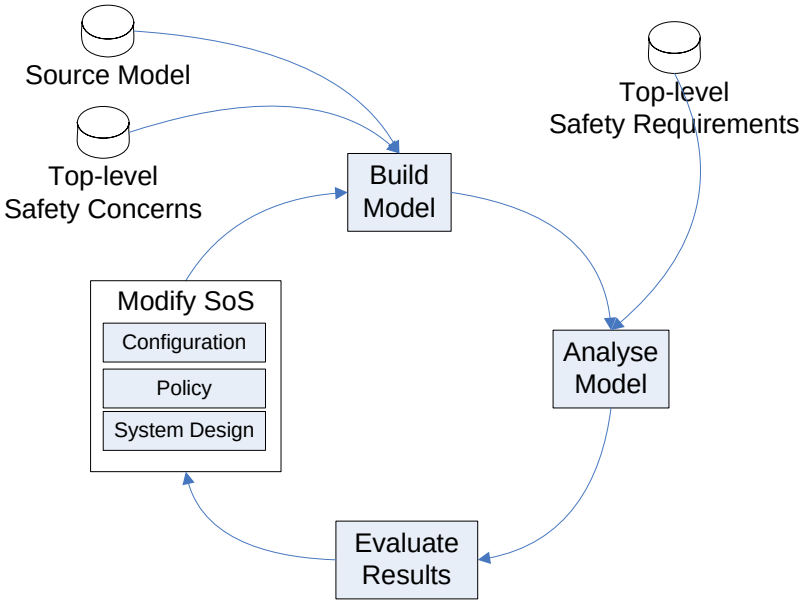

Figure 1 – Overview of SoS Hazard Analysis Process

system that is not provided by high-level mathematical models. For emerging classes of system, this kind of insight is extremely valuable.

The authors have developed a process for using multiagent simulation to perform hazard analysis, which will be summarised here. Further details can be found in [9].

3.1 Overall Process

In this process, the SoS safety team must develop a multiagent model of the SoS. They do this by taking an appropriate source model (such as a MODAF description of the system) and identifying specific safety concerns that they need to model (such as collisions between aircraft). They must also identify (a) the vignettes that the SoS will be expected to participate in and (b) a set of reasonable deviations that may occur in practice, such as a system suffering a particular kind of failure. The resulting multiagent model must be implemented in a multiagent simulation framework, thereby making the model executable.

Once an executable model is available, the 'space' represented by the deviations of that model must be explored. This is performed by running the model with different combinations of deviations and observing the results.

As each run executes, the actions and states of the system components are logged so that they can be studied later. This invariably produces a huge volume of output. To aid comprehension of this data, machine learning techniques can then be used to extract high-level descriptions of hazards, and, once interesting accident runs are identified, causal explanations can be derived using an agent tracing tool.

An outline of this process is shown in figure 1, and key aspects are expanded on in the following sections.

3.2 Modelling

A common concern in simulation modelling is that, in going from a paper model to an implemented simulation, distortions and errors can be introduced. It must therefore be possible to show that the aspects of the system that are of concern to the modeller are represented in the model in all its various representations and artefacts, including the final executable simulation and the output that it produces. These concerns must be expressed in terms of aspects of the source model, and updated and checked at each modelling stage or iteration of the model.

In this approach, the source model is a MODAF description of the SOS. The MODAF Executive Summary [10] describes MODAF thus:

"The MOD Architectural Framework (MODAF) is a framework for developing architectures that provide a means to model, understand, analyse and specify Capabilities, Systems, Systems of Systems (SoS) and Business Processes."

Once a set of concerns have been identified, the modeller must relate them to the MODAF model, identifying which concerns have particular implications for particular MODAF products or operational nodes. For example, for the concern 'unreliable radio communications' the modeller could note that deviations of radio communication impact needlines in OV-2 (operational node connectivity description) that would be served by radio communication and interactions in OV-5 (operational activity model) that correspond to radio messages.

The process of hazard analysis is inevitably based on identifying what deviations of expected system behaviour could lead to hazards. Simple 'brainstorming' techniques can be effective here, but explicit, systematic techniques can provoke modellers to consider deviations that they otherwise would have missed. Such systematic techniques can capture accumulated expertise and knowledge from past experience with similar systems.

In the current approach, agent deviation is performed using an approach based on Failure Modes and Effects Analysis (FMEA) as described in [11]. In FMEA, a safety analyst works systematically over a set of components of a system, asking at each step "how could the failure modes of this component affect the subsystem it is part of, and thereby affect the wider system".

In the approach described here, the analyst works over distinct parts of agents and services provided by them (analogous to FMEA 'components'), using a set of generic deviation forms to derive deviations that are specific to the agent in question. In effect, the analyst proceeds across an identified `atomic unit' of the system and derives the deviations that could reasonably be expected to occur.

Unlike (manual) FMEA, no particular attempt is made at this stage to derive the effects of the deviations --- that is a matter for the simulation

to achieve. Of course, the modeller will need to bear in mind the identified concerns, to ensure that the deviations adequately explore them.

The generic deviation forms are derived by combining the set of generic agent parts and services with a set of guide words.

Table 1 Generic Deviations

Entity Part/Service	***Guide Word***	***Generic Deviation***
Sensor	OMISSION	Total loss of sensing
		Reduction of sensor range
	COMMISSION	Duplication of contacts
		Wholly 'imaginary' contacts
		Increase of sensor range
	EARLY	n/a
	LATE	Delay in registering sensor contacts
	INCORRECT	Incorrect identification of contact side/force
		Incorrect identification of contact entity type
		Incorrect determination of contact location
Actuator	OMISSION	Total loss of function
		Reduction in magnitude of function (e.g. damage, speed, range)
		Partial loss of applicability of function
	COMMISSION	n/a
		Increase in magnitude of function
	EARLY	n/a
	LATE	Delay in performing function
	INCORRECT	Function applied to wrong target (e.g. entity, location, direction)
		General loss of precision/control (e.g. wide area, extra entities)
Plan	OMISSION	Plan step omitted
		Trigger condition not implemented
	COMMISSION	Plan step duplicated
		Extra trigger condition
	EARLY	Plan step moved earlier in sequence
	LATE	Plan step moved later in sequence
		Plan takes more thinking/processing time
	INCORRECT	Substitute entire plan with another
Communications	OMISSION	Loss of transmission
		Loss of receiving
		Entity excluded from network
	COMMISSION	Duplicate messages sent
		Duplicate messages sent later
		Entity added to network
		Additional bandwidth used
	EARLY	n/a
	LATE	Delay in sending messages
		Delay in receiving/processing messages
	INCORRECT	Value error in message
Situational Awareness (SA)	OMISSION	SA does not persist
	COMMISSION	Duplication of entity traces
	EARLY	Trace prematurely removed from SA
	LATE	Trace persists in SA after known to be moved/destroyed
	INCORRECT	SA coordinate system mismatched with peer entities
Computation/thinking	OMISSION	Processing tasks dropped
	COMMISSION	n/a
	EARLY	Accelerated processing
	LATE	Delay in processing (reduced processing capability)
	INCORRECT	n/a

3.3 Analysis

Once the model has been built, including specification of the deviations that can be applied, it must be analysed. The analysis technique must select simulation runs to be performed so as to achieve adequate coverage of the parameter space of the simulation while spending the minimum of computation time. Exhaustive exploration of the parameter space demarcated by all agent deviations would be highly desirable, in that all behaviour paths that were implemented by the simulation model would then be revealed. In practice, this is not going to be possible for anything more than a toy example.

The solution adopted in the work described here is to specify a probability for the occurrence of each deviation in any given simulation run, and then perform runs only for those combinations where the combined probability is above a certain threshold. The plausibility of this combination can be determined by setting a threshold for 'incredibility of failure'. This concept originally stems from the nuclear industry, and situations that appear to be more improbable than this threshold are not studied further in hazard analysis. A value for this is given in [12] as 10^{-7} per year of operation (equivalent to 10^{-11} per hour), and this value is adopted here.

Once those runs have been performed, the accidents that occurred need to be identified and their causes found. The former task is relatively easy, since the set of possible accidents is small. The latter, however, is harder, and machine learning techniques have been adopted to make it tractable.

The task of machine learning can be viewed as one of function approximation from a set of *training instances* expressed as input-output pairs; given a function specification (a set of named input parameters (the 'features' used for learning) and a particular form of output value), the algorithm learns the relationship between combinations of parameter values and the output of the target function for those values.

For our purposes, the features represent parameters of the simulation and the output values are the consequences within the simulation. All the features used in the current work are deviations that are applied to the model, and the target function is the set of accidents that occurs during the simulation run.

The output of the learning algorithm is a set of rules that describes the relationship between deviations and accidents. For example, a rule might be "Aircraft 1 lost_radio_comms causes aircraft 1 to collide with aircraft 2".

Such rules, however, only explain how accidents occur in very broad terms. In order to choose appropriate definitions of our hazards, or to take action to prevent or mitigate them, more detailed information about causation is required.

Lam and Barber, in [13] present a tool-supported approach to the comprehension of agent systems. The core of the approach is that, given

a log of the events that occurred in a single simulation run, and an identified event of interest within that run, the system attempts to explain *why* that event happened in terms of its immediate causes. Those causes can each then be explained in the same way, and the process repeated until the final explanation is in terms of the initial state of the simulation run or 'external' events that occurred. This explanation, complete or partial, can be expressed as a causal graph leading to the event that we asked the tool to explain.

A simple example of such an explanation would be of the form *"UAV 1 received a percept indicating the location of an enemy unit. This caused it to form a goal of destroying that enemy unit, which it selected the 'air strike' plan to resolve, and as a consequence of that plan the UAV conducted the 'attack' action using a laser-guided bomb"*.

The tool achieves this by storing what Lam and Barber call 'background knowledge'. This takes the form of a set of possible causal relationships between events of different types and different properties. As the tool tries to explain each event, it reviews these rules to find which previous events could have caused it.

3.4 Responding to the Results

Once the analysis is complete, the analyst must evaluate the significance of the results, in consultation with the rest of the project safety team and other stakeholders. It is likely, particularly early on in development, that the analysis will reveal problems with the SoS that need to be resolved. As indicated in figure 1, this may involve changes to the configuration of the SoS, to the design of the individual elements, or to the policy under which the SoS operates. The role of design changes in safety is well understood, but that of policy is less so, and is discussed in the following section.

4 Policy

4.1 Background on Policy

The belief that numerous independently designed and constructed autonomous systems can work together synergistically and without accident is naïve, unless they are operating to a shared set of rules which is informed by a high level view of the system. In existing systems of systems such rules already exist, to a degree, because otherwise such systems would be nothing more than an uncoordinated collection of parts. Burns, in [14]: "The proper functioning of the network as a whole is a result of the *coordinated* configuration of multiple network elements whose interaction gives rise to the desired behaviours."

The problems that we face, however, are that often these rules or procedures are either not explicitly expressed, not well understood or are inconsistent. Similarly, they typically do not consider the inter-operating systems as a whole SoS, or simply do not address the safety aspects arising from this inter-operation. A term that can be used to encompass such rules and procedures is policy. Whilst some existing work covers

security policy, no work yet deals with a policy for the safe operation of a system of systems.

The Oxford English Dictionary [3] defines ‘policy’ as:

“A course of action or principle adopted by a government, party, individual, etc.; any course of action adopted as advantageous or expedient.”

Intuitively, therefore, a policy guides the action of an individual or group according to some criteria. Foreign policy, for example, is a familiar concept from everyday language and sets out ground rules for guiding a nation’s diplomatic interactions with other nations. Similarly, common law attempts to curtail undesirable—and hence illegal—behaviour and promote desirable behaviour amongst the populace.

Much of government policy, however, confuses policy with ‘goal-setting’. Although some definitions of policy mention goals, they are in the context of policy goals, or high-level actions, such as “the system is to operate safely at all times” or “no University applicant should be discriminated against based on his/her ability to pay tuition fees”, as distinct from targets, e.g. “to ensure 50% of school-leavers continue to higher education”. Policy can therefore be thought of as being *orthogonal*, but complementary, to plans and goals.

Policy is defined in the literature in various ways, but the most generally applicable system-oriented definition is given in [15]:

“A policy is a rule that defines a choice in behaviour of a system.”

This definition is distinct from that used in, for example, reinforcement learning, where a prescriptive policy maps from perceived internal state to a set of actions. Indeed, it can be seen that policy is persistent [16]; policy is not a single action which is immediately taken, because a policy should remain relatively stable over a period of time. Any policy containing one-off actions is brittle, in that it cannot be reused in a different context and quickly becomes out-of-date and invalid.

Most organisations issue policy statements, intended to guide their members in particular circumstances [17]. Some provide positive guidance, while others set out constraints on behaviour. To take a simple example as an illustration, consider a mother who asks her child to go to the corner shop to buy a pint of milk. She may lay down two rules with which the child must comply on this trip:

1. The child must not talk to strangers.
2. The child must use the pedestrian crossing when crossing the road.

The first of these rules defines what the child is allowed to do, specifically it proscribes conversation with people with whom the child is not previously acquainted. The second statement expresses the obligation that the child should take a safe route across the road, namely by using

the pedestrian crossing. Together these rules form a policy that guides the behaviour of the child on his journey to the corner shop. The rules are invariant to the child's 'mission'; they still hold whether the child is going to buy a loaf of bread or a dozen eggs, or not going to the corner shop at all.

4.2 Systems of Systems and Safety Policy

According to Bodeau [17], the goal of SoS engineering is "to ensure the system of systems can function as a single integrated system to support its mission (or set of missions)." Among the principle concerns of SoS engineering that Bodeau identifies are interoperability, end-to-end performance, maintainability, reliability and security. Unfortunately, he neglects to mention safety.

Wies, in [18], describes policy as defining the desired behaviour of a system, in that it is a restriction on the possible behaviour. Leveson extends this sentiment to say that the limits of what is possible with today's (software-based) systems are very different to the limits of what can be accomplished *safely* [6]. In terms of collaborative groups of systems, SoS, whose behaviour has been observed to be non-deterministic, a policy is a mechanism to create order or (relative) simplicity in the face of complexity. Sage and Cuppan [19] talk of "abandoning the myth of total control", while Clough [20] describes it as creating a system that is "deterministic at the levels that count", i.e. at the 'black-box' level, and Edwards [21] observes the need to "selectively rein in the destructive unpredictability present in collaborative systems".

In discussing policy many different terms are employed, such as rule, procedure, convention, law and code of conduct. The presence of so many terms would seem to suggest a lack of clarity about what policy is, but these terms can be viewed as policy at *different levels of abstraction*. Often policy specifications cause confusion by combining statements at high and low levels of abstraction [18].

Policy statements or goals can be organised into a hierarchy, with the most abstract at the top. There is a need to refine from these abstract policies down to implementable, atomic procedures. Existing goal-oriented techniques and notations, such as GSN [22], KAOS [23] and TROPOS [24], provide a basis for the decomposition of high-level goals. Specifically, the Goal Structuring Notation (described by Kelly in [22]) allows the explicit capture of contextual assumptions, for example assumptions made about other agents' behaviour, and of strategies followed to perform the decomposition.

At the lowest level of abstraction policies can be expressed in terms of the permissions, obligations and prohibitions of individual and groups of agents. In this paper, an approach is suggested for decomposing and implementing policy goals motivated by safety concerns in a simulation of a SoS. The effect of this policy is to moderate the behaviour of the agents such that no accidents occur in the simulated SoS.

4.3 Deriving Safety Policy from Hazard Analysis

A safety policy can be thought of as a set of operational rules that guides the behaviour of individual systems, collaborating as part of a SoS, such that the emergent SoS-level behaviour does not result in accident. Such a safety policy aims to stop hazards that may arise from inadequate control over the interaction between component systems. Given that the component systems exhibit some degree of autonomy, and that this autonomy is considered beneficial, policy rules serve to circumscribe the limits of safe behaviour, not to unduly restrict the inherent flexibility of the SoS. In terms of the safety concepts already discussed, a policy is derived from hazards and ensures the operational satisfaction of safety requirements. In order to arrive at rules that govern system operations a policy decomposition is needed.

Policy decomposition refers to the process of transformation of high-level policy goals into more specific policies that are defined in terms of lower-level entities and operations of the system. Figure 1 shows the top-level structure for the scenario that is described in section 5. At this level the hazards are quite generic, and hence the safety policies contain very little SoS-specific detail. Figure 2 shows an excerpt from the policy hierarchy concerning a decomposition of the goal “avoid enemy fire”. The policy decomposition hierarchy is represented using the Goal Structuring Notation (GSN) [22], typically used to describe safety arguments for use in single-system safety cases. Policy goals are depicted as rectangles. A parallelogram indicates a strategy, which describes the type of goal decomposition approach taken. For example, a strategy might be to decompose over the nature of an obstacle to be avoided, since the policy to avoid an oncoming aircraft (which has a pilot who can reason about your behaviour and manoeuvre to avoid you too) is different to that for avoiding a stationary object.

Similar assumptions (captured by context models) can be used to generate different rules for systems with varying capabilities, or properties of the environment (e.g. two policies for day and night time, or for systems equipped with radar vs. those without). What at first appears to be simply an operational decision is in fact a design decision for autonomous systems, but is masked by the flexibility of human operators to adapt to changing requirements. Consider, for example, the policy for a group of aircraft either to flock or to follow diverse routes to a common destination. For manned aircraft, this would be a simple enough exercise (provided the pilots were suitably trained); however for a group of UAVs it would entail extra flight control algorithms and the exchange of flight path information. This, in turn, impacts the design of the flight control system of the UAVs. In a manner similar to that shown in Figure 2, it is possible to generate further safety policy rules for the other system/hazard combinations given in Table 1. For instance, safety policy rules for the UAV to avoid collision must involve the use of sense-and-avoid, however they can also reduce the probability of collisions between UAVs by obliging them to patrol discrete (non-overlapping) areas or at different altitudes.

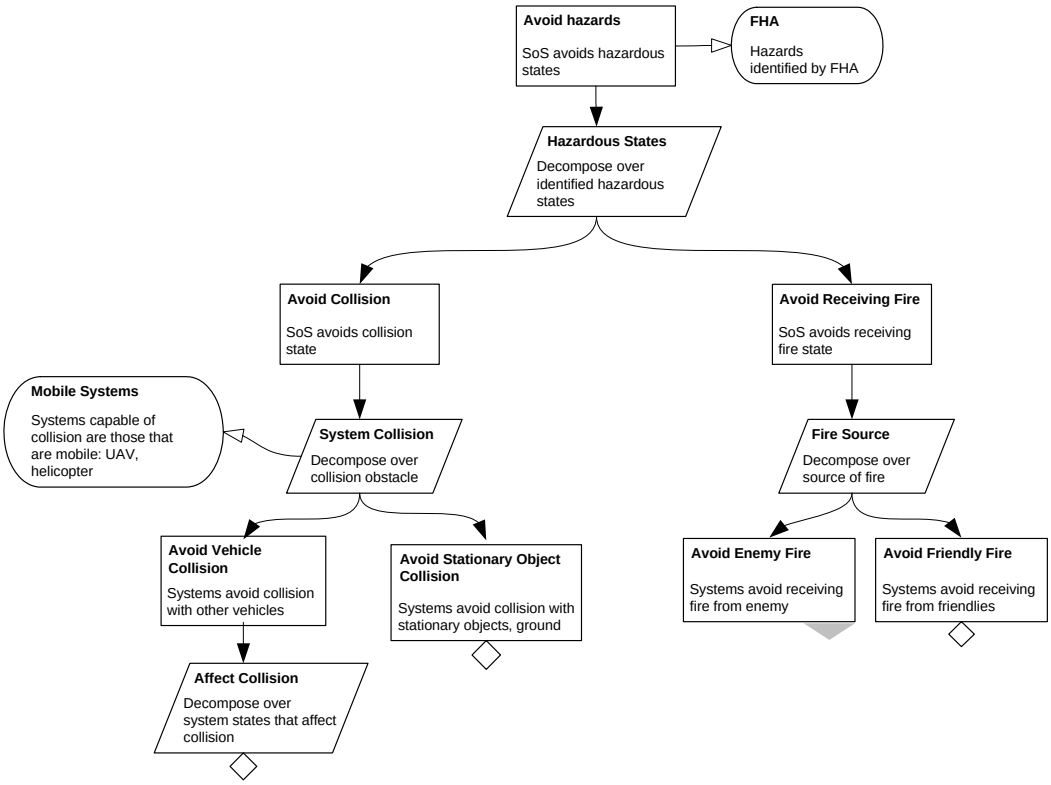

Figure 2 - Top-level Policy Decomposition

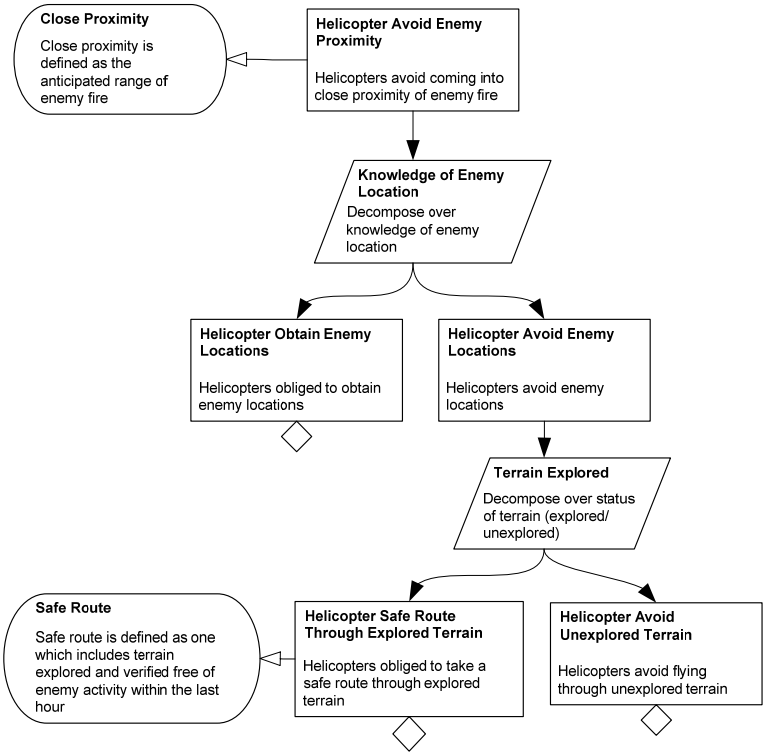

Figure 3 - Policy Rule Specification

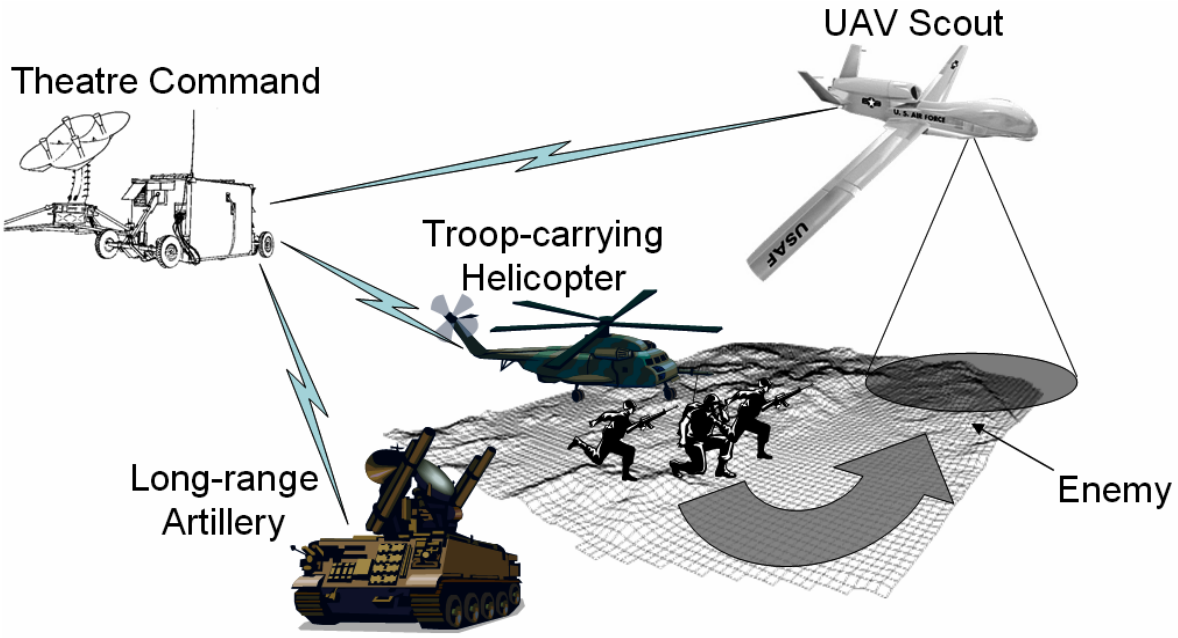

Figure 4 - Anti-Guerrilla Operations

5 Case Study

This case study deals with a hypothetical military system operating in an anti-guerrilla role. It uses network-centric technology to coordinate (unmanned) mobile artillery and airborne special forces in action against an enemy who are hard to detect and spread across a wide area.

An overview of the system's operation in shown in figure 4. There are key parts to the unit: unmanned air vehicles, unmanned self-propelled guns, transport helicopters and special forces sections. The standard operating procedure is that UAVs locate targets, the guns provide artillery bombardment to suppress, disorder and weaken the target, then the special forces move in (carried by the helicopters) to deal with any remaining enemy and secure the area.

Only a single vignette is described for this scenario. The UAVs patrol the area, using a variety of sensors to observe any activity in the terrain below them. When they detect signs of enemy movement, they communicate this to the other entities via some form of data fusion, and this allows the guns to open fire. The UAVs continue to monitor the situation, and feed their observations back to the other entities. When it appears from the shared picture that the enemy have been adequately weakened the guns will cease fire and the helicopters will move in.

A partial MODAF description of the system is given by Despotou in [25]. This was used to derive a simulation model. A small number of deviations were devised and implemented, as shown in table 2.

Agent	Part/Service	Generic Deviation	Specifics	Code
UAV	Comms	Loss of transmission	ok	LOSSOFCOMMS
	Air obstacle sensor	Total loss of sensing	ok	NOAIRSENSORS

	SA/worldview	SA coordinate system mismatched with peer entities	Just do one version	FUSIONGRIDNORTHWEST SKEW
Gun	Artillery fire actuator	General loss of precision / control	Increase area over which fire is distributed	FIRESPREADWIDELY
	SA/worldview	SA coordinate system mismatched with peer entities	Just do one version	FIRESKEWNORTHWEST

Table 2 - AGO System Deviations

5.1 Performing Hazard Analysis on the Case Study

Performing all combinations of deviations requires 262 thousand runs. A decision tree learner (C4.5, described by Quinlan in [26]) was used to derive rules that described hazards. A more detailed discussion of the results can be found in [9], but here we will just show the exploration of one example rule:

```
IF
     UAV 2 has NOT lost communications AND
     UAV 4 has lost communications AND
     Gun 3 NOT generally inaccurate AND
     Gun 3 NOT skewed northwest
THEN
     Some gun destroys helicopter 1
```

A tracing tool was applied to a run that exhibited that accident, and a trace was produced to explain the accident event as shown in figure 5.

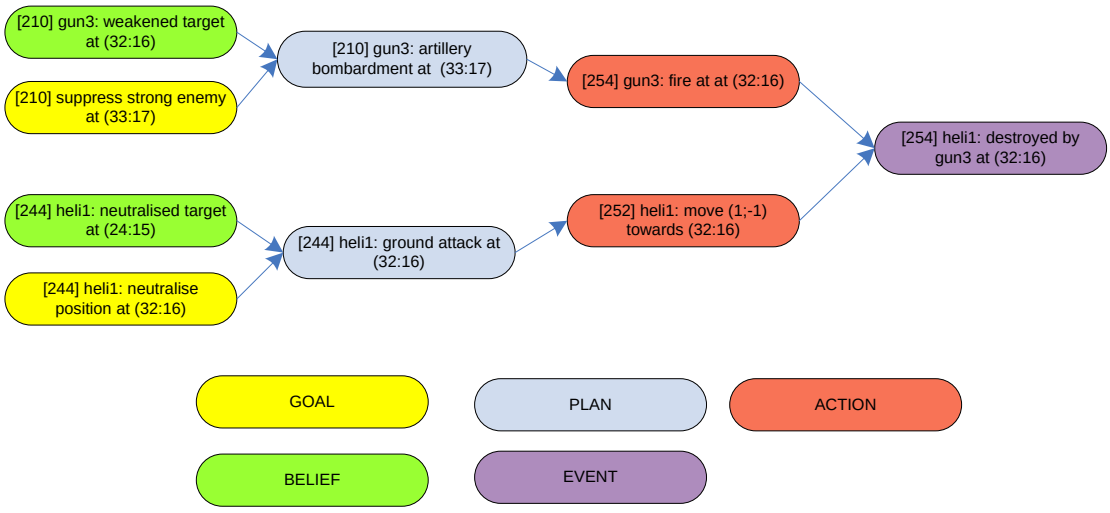

Figure 5 - Tracing result

It can be seen that the helicopter adopts the goal of neutralizing anything at a particular coordinate and so moves into it. The gun, at the same time, adopts the goal of suppressing the enemy in the neighbouring square and fulfils the goal with plan of artillery bombard. But the action actually performed (due to the 'skew' deviation applied to the gun) is to fire at the square that the helicopter has just moved into. The artillery fire therefore hits the helicopter.

5.2 Safety Policy for the Case Study

It has been shown that securing the area in the AGO scenario involves both suppressing and neutralising the enemy. The task of providing suppressing fire has been assigned to the artillery agent, while neutralising the enemy involves transporting special forces by helicopter to the already suppressed enemy for close combat. This means that the helicopters are brought into proximity of the same target as the guns have been firing on.

It should be obvious that, although the helicopter and artillery's efforts are coordinated via the reports of the UAV scouts, there is no direct communication between the helicopter and the artillery. The hazard therefore exists that the infantry may be in the vicinity of the target at the same time as the artillery fires upon it. If the artillery piece is inaccurate in its fire, this can lead to a friendly fire accident. Given that no direct communication is possible between the two agents (nor indeed necessary to achieve the aims of the mission), a safety policy must be derived to make the artillery aware of the helicopter's movements.

The resulting policy decomposition is shown in 6. The left-hand side of the policy structure concentrates on informing the artillery of the helicopter locations so as to avoid accidental attack, whereas the right-hand side focuses on informing the helicopters to avoid an area designated as a target for the artillery.

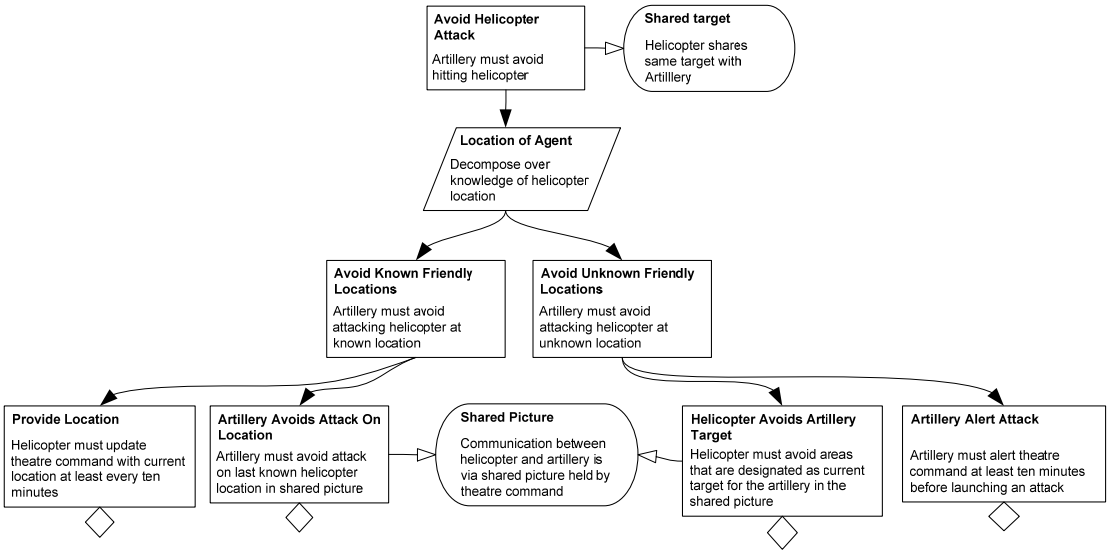

Figure 6 - Policy Structure for Avoiding Friendly Fire

6 Summary and Conclusions

SoS safety engineering is a challenging task, and the existing manual methods that are widely used do not readily extend to the SoS context. There is, however, potential to use multiagent simulation to replace these traditional methods.

In this paper, we have presented a promising method and shown how it can be used with a small but plausible example of a SoS. The construction of a multiagent simulation model allowed the system behaviour to be derived under a variety of conditions, while machine learning and agent tracing techniques allowed the overall safety-critical behaviour of the system to be comprehended.

Future work will include application to larger case studies, and to use with more detailed models of agents and their environment. There is also potential to apply similar techniques a range of challenging hazard analysis applications, such as highly autonomous systems. Key research concerns include the credibility of the results derived from such simulation models, and the relative effectiveness of this work compared to key alternatives such as model checking.

7 References

[1] Maier, M.W.: Architecting principles for systems-of-systems. In: 6th Annual Symposium of INCOSE. (1996) 567–574

[2] Periorellis, P., Dobson, J.: Organisational failures in dependable collaborative enterprise systems. Journal of Object Technology 1 (2002) 107–117

[3] Simpson, J., Weiner, E., eds.: Oxford English Dictionary. Second edn. Oxford University Press (1989)

[4] Wilkinson, P.J., Kelly, T.P.: Functional hazard analysis for highly integrated aerospace systems. In: IEE Seminar on Certification of Ground / Air Systems, London, UK (1998)

[5] Perrow, C.: Normal Accidents: Living with High-Risk Technologies. Basic Books, New York (1984)

[6] Leveson, N.G.: A new accident model for engineering safer systems. Safety Science 42 (2004) 237–270

[7] Ferber, J.: Multi-Agent Systems: an Introduction to Distributed Artificial Intelligence. Addison-Wesley (1999)

[8] Ilachinski, A.: Exploring self-organized emergence in an agent-based synthetic warfare lab. Kybernetes: The International Journal of Systems & Cybernetics 32 (2003) 38–76

[9] Alexander, R., Kazakov, D., Kelly, T.: System of systems hazard analysis using simulation and machine learning. In Gorski, J., ed.: Proceedings of the 25th International Conference on Computer Safety, Reliability and Security (SAFECOMP ’06). Volume 4166 of LNCS., Gdansk, Poland, Springer-Verlag (2006) 1–14

[10] MODAF Partners: MODAF Executive Summary. Version 1.0, 31 August 2005. www.modaf.com

[11] Alain Villemeur. Reliability, Availability, Maintainability and Safety Assessment: Volume 1 — Methods and Techniques. John Wiley & Sons, Chicester, England, 1992.

[12] Ammirato, F., Bieth, M., Chapman, O.J.V., Davies, L.M., Engl, G., Faidy, C., Seldis, T., Szabo, D., Trampus, P., Kang, K.S., Zdarek, J.: Improvement of in-service inspection in nuclear power plants. Technical Report IAEA-TECDOC-1400, International Atomic Energy Agency (2004)

[13] D N Lam and K S Barber. Comprehending agent software. In Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005), 2005.

[14] Burns, J., Cheng, A., Gurung, P., Rajagopalan, S., Rao, P., Rosenbluth, D., Surendran, A.V., Martin, Jr, D.M.: Automatic management of network security policy. In: Proceedings of the DARPA Information Survivability Conference and Exposition. Volume 2., Anaheim, California, USA, IEEE Computer Society (2001) 1012–1026

[15] Damianou, N., Dulay, N., Lupu, E., Sloman, M.: Managing security in object based distributed systems using Ponder. In: Proceedings of the 6th Open European Summer School (Eunice 2000), Twente University Press (2000)

[16] Moffett, J.D., Sloman, M.S.: The representation of policies as system objects. In: Proceedings of the Conference on Organizational Computing Systems, Atlanta, Georgia, USA, ACM Press (1991) 171–184

[17] Bodeau, D.J.: System-of-systems security engineering. In: Proceedings of the 10th Annual Computer Security Applications Conference, Orlando, Florida, USA, IEEE Computer Society (1994) 228–235

[18] Wies, R.: Using a classification of management policies for policy specification and policy transformation. In Sethi, A.S., Raynaud, Y., Fure-Vincent, F., eds.: Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management. Volume 4., Santa Barbara, California, USA, Chapman & Hall (1995) 44–56

[19] Sage, A.P., Cuppan, C.D.: On the systems engineering and management of systems of systems and federations of systems. Information, Knowledge, and Systems Management 2 (2001) 325–345

[20] Clough, B.T.: Autonomous UAV control system safety—what should it be, how do we reach it, and what should we call it? In: Proceedings of the National Aerospace and Electronics Conference 2000, Dayton, Ohio, USA, IEEE Computer Society (2000) 807–814

[21] Edwards, W.K.: Policies and roles in collaborative applications. In: Proceedings of the Conference on Computer-Supported Cooperative Work, Cambridge, Massachusets, USA, ACM Press (1996) 11–20

[22] Kelly, T.P.: Arguing Safety—A Systematic Approach to Managing Safety Cases. DPhil thesis, University of York, Heslington, York, YO10 5DD, UK (1998)

[23] Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Science of Computer Programming 20 (1993) 3–50

[24] Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A.: Tropos: An agent-oriented software development methodology. Journal of Autonomous Agents and Multi-Agent Systems 8 (2004) 203–236

[25] Despotou, G: DODAF Model Development Methodology; The AGO Example. Technical Report DARP/TR/2005/16, University of York.

[26] J R Quinlan. C4.5: Programs for Machine Learning. Morgan Kauffman, 1993.