Towards Using Simulation to Evaluate Safety Policy for Systems of Systems

Robert Alexander, Martin Hall-May, Georgios Despotou, and Tim Kelly

Department of Computer Science
University of York, York, YO10 5DD, UK.
{robert.alexander, martin.hall-may, georgios.despotou,
tim.kelly}@cs.york.ac.uk

Abstract. The increasing role of Systems of Systems (SoS) in safety-critical applications establishes the need for methods to ensure their safe behaviour. One approach to ensuring this is by means of safety policy — a set of rules that all the system entities must abide by. This paper proposes simulation as a means to evaluate the effectiveness of such a policy. The requirements for simulation models are identified, and a means for decomposing high-level policy goals into machine-interpretable policy rules is described. It is then shown how the enforcement of policy could be integrated into a simple agent architecture based around a blackboard. Finally, an approach to evaluating the safety of a system based on simulation runs is outlined.

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 — attempts 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.

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. Evaluating the effectiveness of such a decentralised policy is not straightforward in a complex system, since the overall safe behaviour only emerges from the behaviour of the entities themselves.

An 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 formal analysis methods and conventional system safety techniques are not adequate for evaluating the safety of an SoS. In this paper, we propose simulation as a viable alternative.

Once the 'baseline' model of an SoS has been evaluated, the analysis can be repeated for a range of candidate safety policies. Based on the results of this analysis, candidate policies can be modified or discarded, and the process repeated until a satisfactory safety policy is found. We believe that simulation is a valuable tool for the development of SoS safety policy.

1.1 Structure of this Paper

The following section describes the problems faced in analysing and ensuring the safety of SoS. Section 3 introduces the concept of safety policy. Section 4 describes what is required from a simulation engine and simulation model. Section 5 outlines an approach to implementation of an SoS as a multi-agent simulation that satisfies the identified requirements. Section 6 highlights how safety cannot be considered in isolation from other dependability attributes. Section 7 describes some issues and challenges that need to be tackled, and section 8 presents a summary.

2 The Problem of SoS Safety Analysis

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 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 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 various failures. Conventional techniques such as fault tree analysis are effective in this task.

In an 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 an 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.

An 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 an SoS safety analysis approach will need to be able to capture the effects of interactions between multiple simultaneous failures and normal agent behaviour.

3 What is Safety Policy?

3.1 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 [7]: "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 [8]:

"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* [9]; 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 [9]. 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.

3.2 Systems of Systems and Safety Policy

According to Bodeau [10], 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[11], 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 [12] talk of "abandoning the myth of total control", while Clough [13] describes it as creating a system that is "deterministic at the levels that count", i.e. at the 'black-box' level, and Edwards [14] 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 [11].

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 [15], KAOS [16] and TROPOS [17], provide a basis for the decomposition of high-level goals. Specifically, the Goal Structuring Notation (described by Kelly in [15]) 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 an SoS. The effect of this policy is to moderate the behaviour of the agents such that no accidents occur in the simulated SoS.

4 Requirements on the Simulation Engine and Models

Multi-agent Simulation has previously been used in a safety context, for example to evaluate the safety of proposed changes to the US National Airspace System [18] and to study the relationship between road intersection layout and automobile accidents [19]. As noted by Ferber in [20], such simulations *"make it possible to model complex situations whose overall structures emerge from interactions between individuals"*.

However, not all multi-agent simulations are suitable for safety analysis. In order to perform safety analysis using simulation, there are two key requirements that must be satisfied by the simulation environment and the models that it contains. Firstly, the simulation must be able to generate the types of hazards and accidents that are of concern, without the emergent system behaviour being described in advance. Secondly, it must be possible to detect these situations when they occur.

For example, consider a system to be analysed that involves flocking Unmanned Air Vehicles (UAVs). Given a description of how the entities behave, in terms of flight control, attempting to achieve mission goals, and collision avoidance, it must be possible to run a simulation of a typical mission scenario and see what flight paths the entity behaviour would generate. It must also be possible to detect whether these flight paths would lead to collisions, or hazardous loss of separation.

From the general requirements above, and by looking at the nature of the accidents we are concerned with, a number of more detailed requirements can be derived. These requirements are discussed in the following sections.

4.1 Sharing of Physical Space

Safety-critical accidents must, by their nature, occur in physical space. At the point of an accident, it is through physical interaction that humans are injured or killed. It follows that a safety-related simulation must have a clearly-defined model of space and time interactions. Models that abstract away such details (e.g. by maintaining only relative time ordering, or by dividing geography into large, arbitrarily shaped regions) will not be able to capture the necessary interactions.

It can be noted that although physical space is needed to actually effect an accident, many accidents have causes which can be traced back to events and interactions at the control system or communication levels.

4.2 Autonomous Entity Behaviour

The SoS that are of concern to us involve entities with a large degree of autonomy. Many SoS that are being developed now feature unmanned vehicles, and their autonomous behaviour is an important issue for safety analysis. Negative emergent behaviour, resulting from the interaction of many such vehicles, is a particular concern. It is therefore important to model autonomous behaviour. Autonomous agents are also needed in order to simulate deviation from expected scenario courses; entities must be able to make plausible decisions and actions once the situation has departed from the expected course of events.

The simulation cannot, therefore, rely on a single centralised plan of action. The entity models must be capable of some degree of planning and decision-making so as to achieve their goals in the face of unexpected obstacles.

4.3 Local and Shared Entity World Views

A common cause of accidents in many systems is a discrepancy between the mental model of one agent (be it a UAV or a chemical plant worker) and the actual state of the world. Each agent has a local world model based on the information that they have perceived directly, that they have received in communication from others, and that they have inferred from the information from the other two sources. For example, an airline pilot can observe other aircraft in their immediate area, can receive notification from an air traffic controller of upcoming weather obstacles, and can infer from the ATC's instructions that the course they have been placed on is free from either.

Increasingly, automated systems are used to share data between agents in a system. Examples include air traffic control centres exchanging data on aircraft that are moving from one region to another, and a group of fighter aircraft having access to the combined vision cones of all their radars (this is sometimes referred to as 'data fusion'). This exchange provides many benefits (potentially including safety benefits, as agent knowledge is increased), but also raises new kinds of hazards. For example, if an agent misidentifies a friendly aircraft as hostile, a data fusion system may propagate that 'knowledge' to many other agents, some of whom may be in position to threaten the friendly aircraft.

4.4 Communication Between Entities

As mentioned above, entities can supplement their world model through communication with other agents. Communication also incorporates command and control relationships, which affect the behaviour of subordinate agents. Errors in communication may, consequently, cause accidents either by modifying an agent's world model or by instructing the agent to perform an unsafe action.

4.5 Proxy Measures of Safety

Although a simulation model may generate explicit accidents, a safety analyst cannot rely on this. As in the real world, accidents in a well-modelled simulated SoS will be rare; they will be avoided due to subtleties of time and distance. For example, a collision between a UAV and a manned aircraft may be repeatedly avoided in a series of different runs, with the two aircraft coming close to collision but never actually colliding. For the case of policy, the number and severity of accidents is therefore too crude a measure for the safety of a given policy. There is therefore a need for surrogate measures (e.g. counting near misses rather than just collisions), offering greater resolution than a simple casualty count.

4.6 Introducing Expected Variation

In a model that is solely concerned with performance, for example, it may be sufficient to capture only average performance over time. For a safety model, this is not sufficient; specific, high-cost events must be captured. Therefore, simulations must not only be performed with idealised models of expected entity behaviour; they must also cover all anticipated failure modes. For example, it must be possible to specify that a UAV included in a simulated system has no functioning IFF (Identify Friend-or-Foe) capability, or that one of its engines is operating at reduced maximum thrust.

Going beyond simple failures, it is also desirable to be able to implement different behaviours. Each entity has a set of default behaviours, and the developer of an entity model may provide a set of optional or alternative behaviours. By swapping behaviours in and out from this larger set, variations in overall entity behaviour can be introduced that may affect the results of the simulation run. An example would be swapping a cautious target identification behaviour for a more aggressive one that made fewer checks before deciding that an entity was hostile.

5 Implementing a Multi-Agent Simulation of an SoS

5.1 Describing Policy in a Machine-Interpretable Form

Policy has to be implemented by individual agents — even if there is a central 'master controller' for the whole system, in the systems we are dealing with it will not be able to enforce policy by fiat. Therefore policy has to be decomposed into rules that are expressed in terms of individual agent behaviour. It follows that for any given entity type, policy must be expressed such that it involves:

– Responding to states or events that the agent is capable of observing
– Making decisions that are within the scope of the agent's intelligence and world model (this is particularly important for non-human agents)
– Taking actions that the agent is capable of performing

To this end, policy is decomposed in the context of an SoS model. This model embodies the contextual assumptions of other agents' behaviour, knowledge and capabilities. Policy decomposition proceeds with increasing specificity, working top-down from a high-level goal to policy statements on individual agents or sets of agents. Goal Structure Notation (see section 3.2) allows the explicit capture of the strategies by which the decomposition is achieved and the context necessary for such a decomposition.

Figure 1 illustrates an excerpt from a possible policy hierarchy for the UK civil aerospace Rules of the Air [21]. The policy decomposition starts from an obvious high-level goal, 'No collisions shall occur in the civil aviation SoS', which is at the top of the diagram. This goal is then decomposed hierarchically, leading eventually to a number of low-level goals (leaf nodes on the diagram; due to space limitations, only two of these are shown). These lowest-level goals correspond to policy rules that can be implemented directly by agents; in the diagram, the goal 'Below1000' has been annotated by a machine-interpretable version of the corresponding policy rule.

The low-level policy statements are expressed as one of three types:

Permit Describes the actions that an agent is permitted to perform or conditions that it is permitted to satisfy.

Forbid Describes the actions that an agent is forbidden to perform or conditions that it is forbidden to satisfy.

Oblige Describes the actions that an agent is obliged to perform.

In contrast to policy definition languages such as Ponder [22], we do not attempt to define those actions which an agent is forbidden to perform as well as those which it is obliged not to perform. As mentioned in section 2.2 it is not always possible to prevent agents from performing actions contrary to policy. Unlike security policies, which often assume the presence of an access control monitor, safety policy cannot assume that aberrant agent behaviour can be blocked by an external controller. For example, in air traffic control, there is no external way to stop a wayward aircraft from straying into a forbidden region of airspace. In a sense, the system operator must rely on agents to police themselves.

There must also be a design decision about the overall permissiveness of the SoS. A policy model can either be open or closed: the former allowing all actions which are not expressly forbidden, while the latter forbids all those actions that are not explicitly permitted. The presence of both permit and forbid in this policy model would therefore appear redundant. This is not so, however, given that exceptions to rules can be expressed in the opposite modality. For instance, in an open policy model, a policy rule may forbid the low flying of an aircraft; exceptions to this rule (e.g. for take-off and landing) can be expressed

as permissions. The more specific permissions must then take precedence over the more general blanket policy to forbid low flying.

5.2 Implementation in an Agent Architecture

The implementation of policy at the agent level is tied closely to the details of the agent architecture used. In the current work, an architecture is proposed based on the 'C4' architecture developed by the Synthetic Characters group at the MIT Media Lab. This is a blackboard architecture, in that it contains a series of subsystems that communicate only through a structured shared memory space. C4 is described by Isla et al in [23]. The blackboard architecture is valuable in that it allows discrete behaviours to be loosely coupled, and hence allows variant behaviours to be easily swapped in and out as described in section 4.6.

Our proposed architecture is depicted in figure 2. The core of the system is the blackboard, which is divided into several sub-boards. Of particular note is the outgoing action board, which determines what the agent actually does after each decision cycle. Each agent has several behaviours, which act by making changes to the blackboard (including the action space). The arbitration strategy is simple — on each time 'tick', all behaviours are processed in turn (from top to bottom in the diagram).

The arbiter also has a role in enforcing adherence to safety policy. It can be seen that one of the first behaviours to be processed is the Policy Processor, which compares the current policy to the current state and 'fires' all the policy rules that apply. This generates a set of permitted and forbidden actions, which is written to the policy sub-board. This sub-board is hereafter read-only — the other behaviours can observe it, in order that they might propose only permitted actions, but they cannot change it. The policy sub-board is regenerated on each tick, as changes in the environment may change which policy rules now apply.

Policy rule firings generate tuples of the form (operator, action, list of parameters). For example:

- (FORBID, change-speed, $<$ 180 knots)
- (FORBID, enter-region, 2000m radius of [15000,5100])
- (PERMIT, attack-target, entity#127)

The rule-firings and the behaviours use the same ontology. As noted above, behaviours can see which PERMIT and FORBID policy rules are active at the current time, and modify their behaviour accordingly. As a supplement to this, or an alternative, the arbiter may check proposed actions (against the policy board) and reject those that are against the rules. This could seem redundant, since the behaviours are part of the agent, and hence as much a trusted source as the arbiter itself. It is easier, however, to build a reliable policy enforcer than it is to build behaviours that always conform to policy. Likewise, it is easier to build a behaviour that chooses its action taking into account what policy currently permits, rather than build one that tries whatever action seems best, then tries to respond when the action is forbidden.

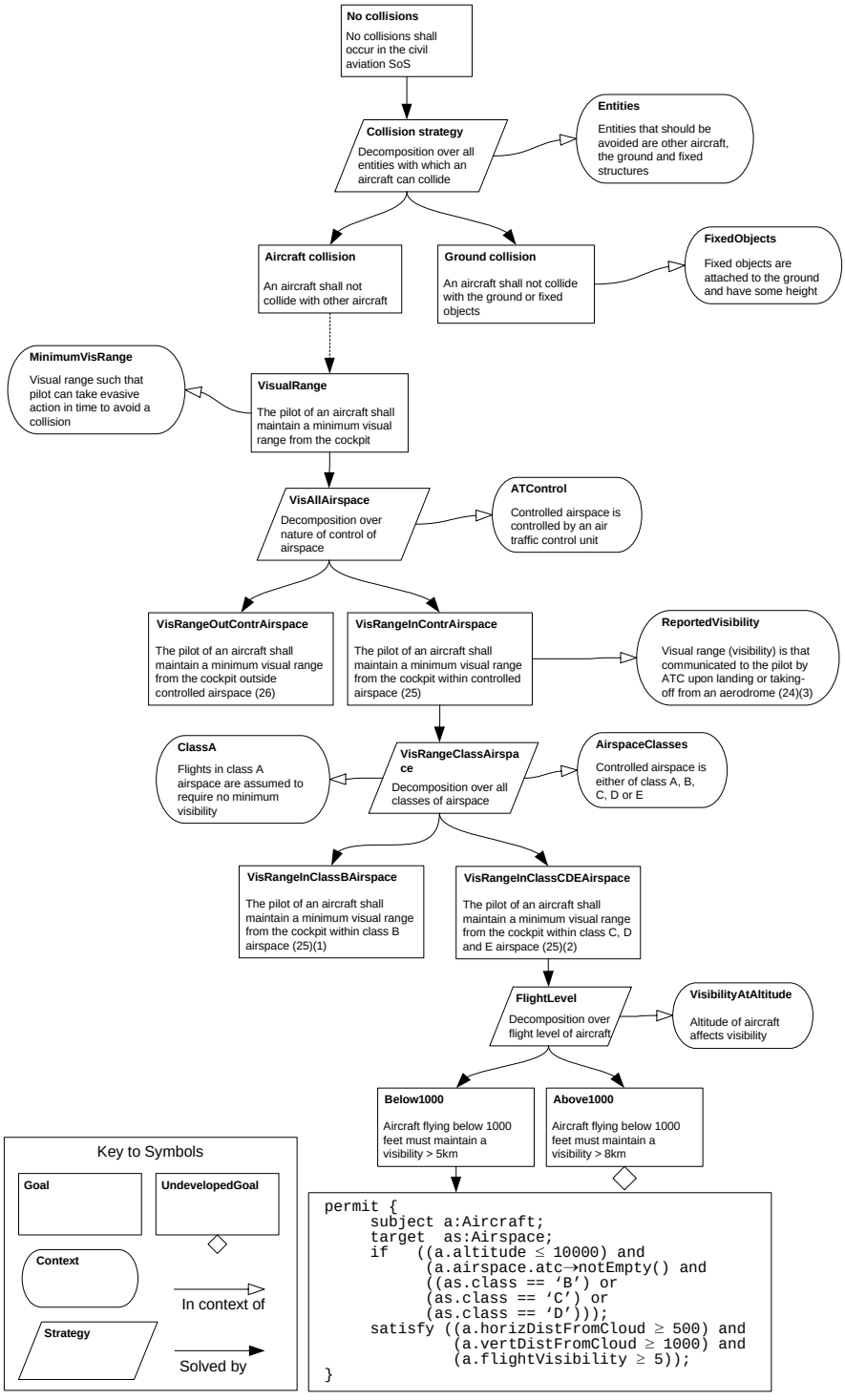

Fig. 1. Example Policy Decomposition for Rules of the Air

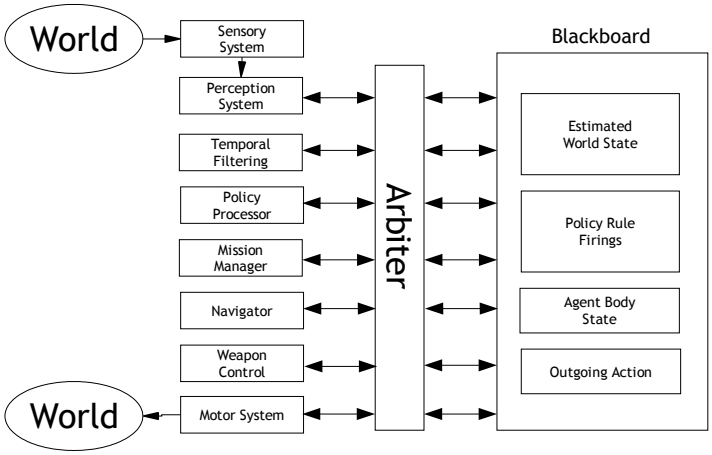

Fig. 2. The Agent Architecture

An alternative to the policy enforcement role of the arbiter is that of a monitor, which notes policy violations but does not prevent them. This is particularly valuable during development of an agent.

An advantage of the blackboard model is that the behaviours are loosely coupled to each other; they interact only through the blackboard. This means that behaviours can be added, removed or changed at any time, without changing the other behaviours. This relates to the requirements identified in section 4.6.

5.3 Evaluating the Safety Achieved by the Policy

In order to evaluate the level of safety that has been achieved, the SoS agents must be configured to use the policy, then simulation runs must be performed for a variety of representative scenarios. Further variation can be introduced through failures and variant behaviours applied to agents. The level of safety achieved by the system operating with the policy can be evaluated by measuring the number of accidents and incidents, and the worst near-incidents, that occurred across all runs with that policy.

Once the SoS model has been configured and the set of runs decided on, measures must be put in place to measure the level of safety achieved by the system. From the definition of safety presented in section 2, it can be seen that two types of event need to be counted.

The first type is accidents, which corresponds to *"exemption from hurt or injury"* in the definition. Examples of such accidents include collisions between vehicles and military units firing on friendly forces. The set of possibilities is quite small, and they can be easily detected.

From *"freedom from danger"* we can derive another class of event, the incident or 'near miss'. Examples include activations of an aircraft's collision-

avoidance system, separation between two aircraft falling below some safe level, and queries to a superior of the form "Is X hostile?" when X is in fact friendly. Unlike actual accidents, a great many types of such incidents can be described. It can be noted that many incidents correspond to *hazards*; when an incident occurs, it may be the case that an accident could happen without any other deviation from normal behaviour.

The great value of counting incidents as opposed to accidents is that accidents are extremely rare — in the real world, accidents are often avoided by (seemingly) sheer chance. Measures that track incidents can be given variable sensitivity, so that they can be adapted to the level of risk that is actually exhibited in the system. For example, it is desirable to calculate actual collisions accurately, so that their effects on the unfolding scenario can be modelled realistically. This is especially true if a multi-criteria analysis is being performed, for example with performance as well as safety being analysed. By comparison, an incident measure based on aircraft proximity can be as sensitive (or insensitive) as is required since triggering it only affects the statistics gathered, not the events that follow it.

For both accidents and incidents, it is possible to weight events by a measure of their severity. Consider one policy that generated a number of minor accidents against another that caused a single accident with massive loss of life. A simple approach is to count the human casualties (or potential casualties, in the case of an incident) that could result from an event. The safest policy is then the one that caused the smallest loss of simulated lives over all the scenarios that were considered. Weighing accidents against incidents is more difficult, however; there is the question here of model fidelity, and the consequent fear that an incident in the simulation might have been an accident in the real system.

The means of detecting accidents during a simulation run are well understood as they are an essential part of many non-safety simulations. Providing a large range of incident detectors is less straightforward, and some of these will raise performance challenges. This is, however, beyond the scope of this paper.

If two policies cannot be compared because their accident and incident counts are zero or very low, a third technique is possible. For a variety of measures, perhaps the same measures as those used for incidents, the worst magnitude achieved could be tracked. An obvious example is violation of aircraft separation; rather than just counting the number of occasions on which this occurred, the minimum separation achieved can be recorded. The minimum for the policy is then the minimum over all runs. An example of this can be seen in Benson [24].

6 Dependability Conflicts in Systems of Systems

Safety is an important system attribute, but it is not the only consideration when developing an SoS. There are other important attributes such as availability, performance and security. The term *dependability* is commonly used to encompass all such system attributes [25]. Attempting to address all these different attributes can result in competing objectives; consequently there are conflicts

that need to be resolved and trade-offs that need to be made in order to achieve the optimum characteristics for a system.

In SoS, conflicting objectives (and hence trade-offs) are inevitable; probably the most obvious are conflicts between performance and safety. An example is the reduction of minimum aircraft vertical separation (RVSM), within controlled airspace. In RVSM airspace, aircraft fly closer to each other, greatly increasing the number of aircraft that can be served by an ATC centre within a certain period of time. This has obvious performance benefits (reduction of delays, more flights during peak hours), but it raises some serious safety concerns. Special safeguards (changes to either sub-system design or operational policies) are therefore required.

If an SoS is developed with safety as the highest priority, it will be possible to devise policies that constrain the interactions of system agents to the safest that are possible. However, such an approach might unacceptably decrease the overall performance of the system. For example, there is no point in introducing an extremely safe air traffic policy if doing so reduces the throughput of the system to uneconomic levels. In order that safety is not achieved at the unacceptable detriment of other attributes, it is important to model the effect on all attributes of safety-related changes.

Performance acceptability criteria differ depending on the particular system mission. Therefore, the required performance level and its criticality (based on which we determine our willingness to compromise safety in favour of performance) are defined with consideration of the system's context and operational objectives. Simulation provides a way to evaluate the different dependability attributes of the system in different contexts, by running a set of representative operational scenarios. This provides a basis for achieving a satisfactory trade-off between the different attributes.

7 Issues and Challenges

7.1 Model Fidelity and Statistical Significance

No novel and untried systems of systems will enter operation with only simulated evidence of its safety. Simulation, however, gives a guide to the behaviour of the system which informs, and is supplemented by, further analysis. It is particularly valuable in that it can reveal the emergent behaviour of a complex system in a variety of contexts; it is difficult if not impossible to acquire knowledge of this by other means.

Even when the fidelity of a given simulation is considered inadequate to assess the safety of a system, it can provide confidence that a given policy is viable, and help judge relative superiority to other candidates. (For an example of this, see Benson in [24]). Perhaps most importantly, the simulation analysis can reveal flaws in a policy that would not have been apparent in manual analysis.

The problem of model fidelity, and of the validity of any results that are gained through simulation, is a serious one and affects all applications of simulation analysis, not just safety. This is a longstanding controversy in the field of

robotics; discussion can be found in Brooks [26] and Jakobi [27]. In the current context, one key requirement for usefulness is that the simulation be able to exhibit emergent behaviour.

7.2 Volume of Processing

As noted above in section 5.3, evaluating the safety of the system requires a large number of scenarios to be simulated. For each of those simulations, a large range of failures and variant behaviours need to be considered. Combinations of failures and behaviours are also important.

It follows that the possible set of simulation runs is extremely large. A naïve approach would be to run all possible combinations of scenario, failures and behaviours. However, as discussed by Hoeber in [28], such exhaustive exploration is intractable even for simple simulations and modest numbers of inputs.

There is therefore a need for more targeted exploration of the state space. In [29], Dewar et al discuss some experimental designs that can be used for reducing the number of combinations that need to be run. Many such designs, however, deal poorly with systems in which the interesting phenomena result from combinations of inputs.

One other approach would be to concentrate on and prioritise those combinations of failures that were statistically most likely. A useful selection criteria can be based on the potential for certain types of SoS failures to occur together, as discussed by the authors in [30].

8 Summary

In this paper, we have presented the case for using safety policy to ensure the safe behaviour of complex systems of systems, and suggested multi-agent simulation as a means of evaluating the effectiveness of such policies. It is clear that analysing SoS is difficult, particularly when they are highly decentralised. Simulation offers an approach to dealing with some of these difficulties.

An approach to policy evaluation has been proposed, whereby an SoS is exercised through a variety of simulations for each candidate policy, with a range of failures and behaviour modifications being introduced. The level of safety provided by each policy can be assessed by measuring the values of various safety-related parameters. This concept can be extended further, using simulation to consider the trade-off between safety and performance.

A number of challenges remain, such as limitations in the fidelity of models and the number of runs needed to get statistically valid results. The authors are currently working on tools and examples to demonstrate the concepts described in this paper.

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. 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
8. 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)
9. 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
10. 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
11. 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
12. 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
13. 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
14. 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
15. Kelly, T.P.: Arguing Safety—A Systematic Approach to Managing Safety Cases. Dphil thesis, University of York, Heslington, York, YO10 5DD, UK (1998)
16. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Science of Computer Programming **20** (1993) 3–50
17. 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
18. Lee, S., Pritchett, A., Goldsman, D.: Hybrid agent-based simulation for analyzing the national airspace system. In Peters, B.A., Smith, J.S., Madeiros, D.J., Rohrer, M.W., eds.: Proceedings of the 2001 Winter Simulation Conference. (2001) 1029–1037

19. Archer, J.: Developing the potential of micro-simulation modelling for traffic safety assessment. In: Proceedings of the 13th ICTCT Workshop. (2000) 233–246
20. Ferber, J.: Multi-Agent Systems: an Introduction to Distributed Artificial Intelligence. Addison-Wesley (1999)
21. Allan, R., ed.: Air Navigation: The Order and the Regulations. third edn. Civil Aviation Authority (2003)
22. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: Ponder: A language for specifying security and management policies for distributed systems. Research Report DoC 2000/1, Imperial College, London (2000) http://www.doc.ic.ac.uk/deptechrep/DTR00-1.pdf.
23. Isla, D., Burke, R., Downie, M., Blumberg, B.: A layered brain architecture for synthetic creatures. In: Proceedings of the International Joint Conference on Artificial Intelligence, Seattle, WA (2001)
24. Benson, K.C., Goldsman, D., Pritchett, A.R.: Applying statistical control techniques to air traffic simulations. In Ingalis, R.G., Rosetti, M.D., Smith, J.S., Peters, B.A., eds.: Proceedings of the 2004 Winter Simulation Conference. (2004) 1330–1338
25. Avizienis, A., Laprie, J., Randell, B.: Dependability of computer systems: Fundanmental concepts, terminlogy and examples. In: Proceedings of the IARP/IEEE-RAS Workshop on Robot Dependability, Seoul (2001)
26. Brooks, R.A.: Intelligence without representation. Artifical Intelligence **47** (1991) 139–159
27. Jakobi, N., Husbands, P., Harvey, I.: Noise and the reality gap: The use of simulation in evolutionary robotics. Lecture Notes in Computer Science **929** (1995)
28. Hoeber, F.P.: Military Applications of Modeling: Selected Case Studies. Gordon & Breach Science Publishers (1981)
29. Dewar, J.A., Bankes, S.C., Hodges, J.S., Lucas, T., Saunders-Newton, D.K., Vye, P.: Credible uses of the distributed interactive simulation (dis) system. Technical Report MR-607-A, RAND (1996)
30. Alexander, R., Hall-May, M., Kelly, T.: Characterisation of systems of systems failures. In: Proceedings of the 22nd Internation Systems Safety Conference (ISSC), System Safety Society (2004) 499–508