SERSCIS: Semantic Modelling of Dynamic, Multi-Stakeholder Systems

Mike Surridge, Ajay Chakravarthy, Martin Hall-May, Xiaoyu Chen, Bassem Nasser
IT Innovation Centre
University of Southampton
Southampton, UK
{ms,ajc,mhm,wxc,bmn}@it-innovation.soton.ac.uk

Roman Nossal
Austro Control
Österreichische Gesellschaft für Zivilluftfahrt mbH
Vienna, Austria
roman.nossal@austrocontrol.at

***Abstract*—This paper describes a novel approach to semantic system and security modelling developed in the SERSCIS project. The approach is designed to address dynamic multi-stakeholder systems that are composed from services at run-time. This presents several challenges for security risk modelling and management that are not well addressed by previous work. The biggest challenge is the fact that at design-time one only knows the structure but not the composition of the system, forcing an abstract modelling approach to be used. The SERSCIS approach deals with this by defining a set of OWL classes describing generic system assets, threats and security controls and the relationships between them. This dependability model captures security expertise concerning the types of threats that can arise in general and the controls that can be used to address them. An abstract system model can then be created using OWL subclasses, to capture the types of assets and their relationships in a specific system, but still without specifying how many assets, where they are deployed or what security controls they have. The resulting models can be used as inputs to run-time semantic monitoring tools, where the knowledge encoded in the abstract system model is used to automatically determine system threat activity and system vulnerabilities. The approach was validated in an Airport Collaborative Decision-Making scenario.**

Keywords-component; semantics; modelling; security

I. INTRODUCTION

The SERSCIS project [13] aims to address the growing need to manage risks in Critical Infrastructure arising from or amplified by information system interconnections between different stakeholders. Increased connectivity is being driven by the need for efficiency. By sharing data, the organizations involved in running critical infrastructure can predict events and manage resources more effectively. Wider access to information also encourages competition, e.g. by requesting a service at some future time, so different suppliers can bid to provide it.

But information sharing also increases vulnerabilities due to three effects:

- attacks can be made against the information systems or communication channels, disrupting the flow or the integrity of exchanged data;
- local disruption in one organization can lead to more problems elsewhere due to the dependency of others on data from the disrupted party; and
- the use of exchanged data to increase efficiency usually leads to a reduction in excess capacity, and so reduces resilience against any type of disruption.

For example, under the Single European Skies initiative, the use of airspace in Europe is managed by Eurocontrol, an international agency that seeks to optimize the flow of air traffic by assigning take-off and landing slots at airports in real time. To do this, Eurocontrol defines a collaborative decision-making process (CDM) in which national air traffic control services share information via Eurocontrol. The CDM approach is now being extended to airport operations (A-CDM), allowing aircraft ground handling operators to provide predictions of when aircraft will be ready to take off. The chain of interconnections means air traffic control in Europe could be disrupted by attacks on or problems of airport service providers that handle tasks like aircraft refuelling, baggage handling, etc.

The SERSCIS project is addressing these issues by treating the information systems in a critical infrastructure as a dynamically composed, multi-stakeholder, service-oriented system. Dynamic service oriented architectures can be used to reconfigure information flows in the event of a problem, providing a dynamic response to mitigate its effects locally or on other parts of the network. The problem is that the risk management methods used in critical infrastructure are based on design-time analysis of information security threats [3,7,10], as typified by the procedures used in the Single European Skies research initiative SESAR [15]. These methods cannot cope with dynamic, multi-stakeholder systems. To overcome this limitation, SERSCIS has developed a novel approach to system modelling, designed to support threat modelling and analysis in dynamic, multi-stakeholder systems. The approach is based on the use of semantic modelling, so machine reasoning can be used to automate the analysis of threats when the system is composed at run-time. In contrast to other approaches, the design-time models are abstract, describing the structure but not the composition of the system (which is not known until run-time). The models are also stakeholder-centric, ensuring

The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement n° 225336, SERSCIS

that they can be used to interpret system behaviour and threats at run-time from the point of view of an individual stakeholder, using only system configuration and status information available to them. This paper focuses on the modelling approach developed in the SERSCIS project, leaving other aspects such as policy modelling and run-time monitoring and risk management to be detailed in other publications [11].

The rest of the paper is organized as follows: In section II we compare our modelling approach to the state of the art in risk and security modelling approaches. In section III we present the layered SERSCIS models (Core, Dependability, Abstract and Concrete). In section IV, we present a detailed validation scenario used to evaluate our approach and finally in section V, we conclude with a description of future work.

II. Relationship to the State of the Art

A. *Risk Management*

As noted above, state of the art risk management methodologies such as COBIT [2], and ISO 27005 [9] are based on analysis of information security risks for a given system design and configuration. The analysis allows system vulnerabilities to be detected, and associated risks quantified. If the risk is too great to be accepted, strategies are defined to reduce the risk (using security controls), avoid the risk (by not using the vulnerable system feature) or transfer the risk to someone else (by outsourcing system functions, or insuring against potential losses). In most methodologies, the effectiveness of control strategies must be monitored, and if necessary the analysis is revisited, e.g. if new vulnerabilities are discovered, or threats prove more likely than expected. However, the approach is essentially a design-time approach, in which changes are effected by amending the system design (e.g. adding new controls), having security experts analyse the new design to check the changes will have the desired effect, and then adding controls to the system. In a dynamically composed service-oriented architecture this approach is not sufficient for two reasons. There is no conventional system design including controls: the system composition changes during run time as services are added removed or replaced, each service having its own controls that may differ from others of its type. And there is no time for a conventional security expert analysis of system changes that occur dynamically.

Matulevicius et al [3] present work on a graphical approach to identify, explain and document security threats and risk scenarios. A graphical notation was developed to perform the five phases needed for security analysis 1. Context establishment, 2. Risk identification, 3. Risk estimation, 4. Risk evaluation and 5. Treatment identification. Diagrams are created during each of these steps (similar to UML models) under the guidance of a domain expert. We have followed a similar approach for the identification of threats and their mitigation strategies. However, the work done by [3] does not go beyond the modelling phase (diagrammatic modelling). The novelty of our SERSCIS system modelling approach is that we use an abstract modelling approach (OWL ontology based) to address the challenge faced in adaptive systems (where the composition of the system is not known in advance). Further, we go beyond the modelling stage and integrate our ontologies into the runtime dynamic stakeholder system. The concrete model (instance information) is automatically generated at runtime depending upon the current composition of the system.

Work by [8] uses an extension of the Secure Tropos language to support the modelling of security risks. The domain model is mainly structured around three groups of concepts: asset-related concepts, risk-related concepts and risk-treatment related concepts. Further security criteria for each of these assets are identified in terms of confidentiality, integrity and availability. This work is an extension of the work on Secure Tropos, and includes the development of syntactic, semantic and methodological extensions that would support security risks and their counter measures. This representation is in a diagrammatic in nature and is used to present abstract syntax elements for risk modelling and the rules on how these can be combined together. The domain model in [8] is similar to the core ontology model we use in SERSCIS (consisting of Threat, Asset and Control). The risk modelling approach we used during the brain storming session uses features from both [3] and [8], however, in SERSCIS we use OWL ontologies to model Core, Dependability and Abstract system models in addition to the diagrammatic representations. This allows our models to be much more expressive (due to expressive nature of OWL syntax) and encoded in a way such that existing rule based languages (e.g. SWRL) and DL reasoners (e.g. Hermit) can be used with these OWL ontologies for the automatic identification and mitigation of threats.

B. *Security Modelling and Machine Reasoning*

One approach that could address the second of these problems (unknown composition of dynamic multi-stakeholder systems) is to use machine reasoning to analyse risks, so this can be done rapidly whenever the system composition changes. There is an existing body of research into how one might create semantic models of a system to support such an automated analysis, though with the motive of capturing security standards and expertise so tools can be developed to support non-experts. A useful overview is provided by [1]. For example, the NRL Security Ontology [10] provides a way to describe the security properties of Web Services, which was later used as a starting point for a Web Service vulnerabilities ontology [16]. The Ontology of Information Security [7] by Herzog et al describes a system in terms of assets, vulnerabilities and threats, so making the link to system risks (via threat models). However, this ontology uses N-ary relationships making it hard to deduce security properties via machine reasoning.

The Security Ontology [5] from Secure Business Austria (SBA) uses only conventional RDF relationships, and captures security threats and controls from the German IT Grundschutz Manual [6], so providing a way to model systems with common threats and control strategies. The SBA approach goes a long way towards the goal of capturing security expertise in a form that can be reused (with supporting tools) by non-experts.

However, this ontology describes only deployed systems and security controls, and cannot be used to create an abstract system model for a dynamically composed system whose concrete composition is not known at design time. The SBA approach also makes extensive use of Web Ontology Language (OWL) instances, which makes it hard to cater for multi-stakeholder systems where it is often necessary to attach different properties to the same threat depending on which stakeholder or sub-domain is targeted.

III. SERSCIS System Models

A. *SERSCIS Approach and Core Ontology*

The SERSCIS approach follows the SBA approach, in that it uses only conventional semantic relationships (no N-ary relationships), so conventional semantic reasoning can be used to make deductions about security from the model. To use this approach for dynamic multi-stakeholder systems, the SERSCIS ontology[1] was constructed such that:

- only OWL classes are used for design-time modelling, allowing an abstract system structure to be captured in terms of the relationships between types of services, and associated threats and controls;
- OWL instances are used to model the run-time system composition, in terms of concrete services with specific controls and subject to targeted threats.

The other design criteria for the SERSCIS approach are:

- security expertise must be added at design time (i.e. in the OWL classes), so run-time models can be created without expert intervention, yet still access their expertise (through machine reasoning);
- the ratio of asserted facts to inferable facts in the run-time model must be as low as possible, since every assertion has to be constructed automatically based on monitoring of the running system.

These requirements led SERSCIS to simplify the high-level class structure used by SBA. The SBA vulnerability class was dropped as it was noticed that it is most often used only to represent a lack of security controls. The resulting SERSCIS core ontology structure is shown in Fig. 1.

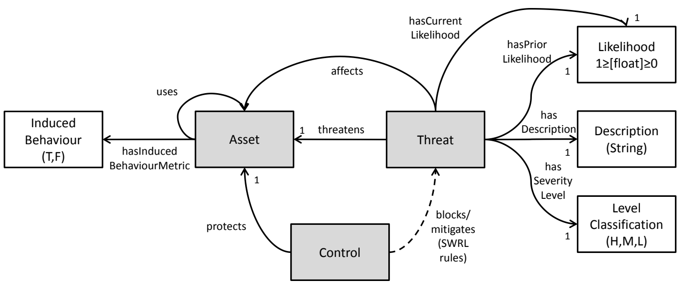

Figure 1. Core SERSCIS ontology

[1] SERSCIS ontologies downloadable at: http://www.serscis.eu/?page_id=317

B. *SERSCIS Dependability Model*

The core structure from Fig. 1 provides the basis for the development of more specific models and analysis tools. The next step is to capture security expertise related to the type of system one is dealing with. In SERSCIS, this is known as a *dependability model*, as it includes the types of assets found in the system, and also the attributes that make these assets dependable or not (e.g. whether they have threat-induced behaviours).

In SERSCIS we are concerned with multi-stakeholder applications in which the relationships between stakeholders are determined at run-time via dynamic composition. Thus the SERSCIS dependability model describes systems in terms of services (and clients), taking a stakeholder-centric view (i.e. the model represents the system as seen by one of its stakeholders). Thus different asset classes are used depending on whether the asset is a service provided by the model stakeholder, a service used by that stakeholder (i.e. a resource), or a client using a service from that stakeholder. It is also important that the dependability model takes account of the fact that in dynamic systems, delegation is often used. Two types of delegation are commonly found:

- a user of a service shares access to the service with another user; or
- a user of two services has one service interact directly with the other.

The first case leads us to define a special type of Client, the Customer, who has a relationship with the stakeholder under which they can use a service. Clients who are not Customers have access to the service only if access is shared with them by the Customer. Each Customer normally has a contextualized endpoint through which they access a service, so each service is really a collection of services (one per Customer). The corresponding asset class is therefore called a Service Group, rather than simply a Service. The second case leads us to distinguish two types of resources. A Client Specified Resource is one specified by a client so a service can interact directly with it. A Provider Specified Resource is not specified by a client, but chosen by the primary stakeholder. In a dynamic system this is often done automatically at run time, by having the service select the resource from a pool of available resources of the required type. This pool is also an asset, represented as a Resource Group.

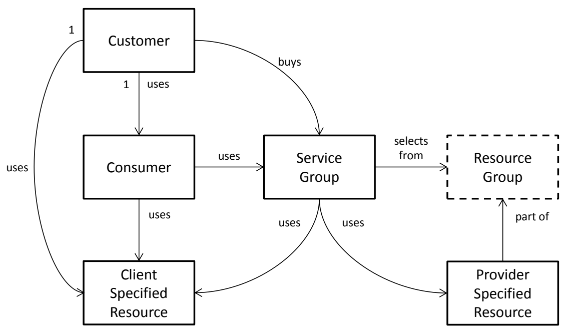

Figure 2. Dependability model logical asset types

The SERSCIS dependability model defines a set of threat induced behaviour states that are symptomatic of different types of asset compromise. A loss of confidentiality arises if the asset behaviour is ‘indiscreet’ while a loss of integrity is linked to ‘unreliable’ or ‘inaccurate’ behaviour (one signifies a process integrity failure, the other data integrity failure). Other behaviours include ‘overloaded’ and ‘underperforming’ associated with availability failures, and ‘unauthentic’ or ‘unaccountable’ indicating a loss of trustworthiness, etc. Threats that can give rise to such behaviour are modelled by their relationships to asset types, and cover the usual range of accidental or malicious disruption including unauthorized access, impersonation (e.g. of a Customer but also potentially delegate impersonation), traffic snooping or corruption, data tampering (by unauthorized or erroneous access), resource shortages (e.g. in a Resource Group), deliberate or accidental service overload, and software bugs. The SERSCIS dependability model covers different ways these threats can arise in the interactions between the asset types from Fig. 2, using different threat classes depending on which asset is threatened and which are affected. Thus impersonation of a service to fool a Customer is considered as a different threat from impersonation of a service to fool a Client Specified Resource, etc.

The SERSCIS dependability model also includes several control subclasses representing the types of controls typically found in a multi-stakeholder service-oriented system. These include access control, identification (meaning an entity has a way to prove its identity), authentication (a means to verify another entity’s identity or other attributes), message encryption and integrity verification, and a range of resource and load management mechanisms such as resource redundancy and failover (to prevent resource shortages), and SLA enforcement (to prevent overloading of services by their clients). Finally, the SERSCIS dependability model includes a set of Semantic Web Rule Language SWRL rules [14] that specify what combination of controls is sufficient to block or mitigate each type of threat depending upon the nature of the control (i.e. proactive or reactive). For example, suppose a client tells a service to interact with another service, i.e. with a client specified resource (CSR). If someone else could access the CSR without authorization, they could interfere with the data exchanged between the CSR and the service. To prevent this threat, the CSR must deploy access control, and have a way to verify a user’s identity or other attributes the access

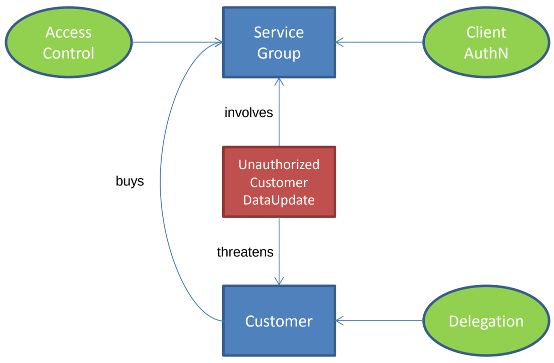

Figure 3. Threat and control modelling

policy requires the user to have. This in turn means the client must have a way to delegate access rights to other parties, and the service (group) needs a way to identify itself so the client can avoid delegating these rights to the wrong person.

Fig. 3 shows how this threat is modelled in terms of its relationships with asset types and controls. The diagram corresponds to a threat class and the associated control rule in the SERSCIS dependability model (actually several threat classes depending on whether the unauthorized access is used to steal or corrupt data at the CSR). If one finds concrete instances of service group, client and client-specified resource assets related to each other as shown, one can use the control rule to deduce whether the system is vulnerable to this threat against those assets. The SERSCIS dependability model contains about 50 such threat classes, so a wide range of types of vulnerabilities can be detected in the running system, using expertise encoded in the dependability model classes.

C. *Abstract System Model*

The SERSCIS dependability model provides the starting point for development of an abstract system model. This describes a particular system in terms of the types of assets it contains and the relationships between them. The abstract system model is a design-time model of a system that will be composed dynamically at run-time. It does not include the concrete system deployment and configuration such as who owns the assets, where they are deployed or with what security controls in place. The idea is that the abstract system model is created at design time by a system expert (not a security expert), using security expertise encoded in the dependability model. The resulting abstract system model is then used (along with the underlying dependability model) as input to a set of fully automated, run-time model generation and analysis tools connected to the system monitoring infrastructure.

Starting from the SERSCIS dependability model, the system modeller proceeds as follows:

- Define sub-classes of the dependability asset classes from Fig. 2, corresponding to the different types of services, clients and resources in their system.
- Define usage relationships between the client, service and resource asset sub-classes.
- Automatically generate threat sub-classes against each relevant combination of asset sub-classes.

The last step can be automated because threats are defined in terms of their relationships to dependability model asset classes. The threat in Fig. 3 attacks a client specified resource by exploiting its relationship to client and service group assets. It is easy to find all combinations of client, service group and client specified resource sub-classes that have the same set of relationships. For each combination, a corresponding threat class is generated. Thus the system designer (not a security expert), does not need to know all the arcane ways in which an attacker might seek to compromise a multi-stakeholder system through these interactions. If they understand the system assets and relationships, SERSCIS tools can be used to fill in a complete set of potential threats.

Once the threat classes have been generated, the system designer can optionally insert a human-readable explanation of the threat, information about the impact of the threat if carried out (high, medium or low), and an estimate of how likely it is that the threat will be active (a probability between 0 and 1). All these can be taken from the underlying dependability model threat classes, but it is advisable to modify the explanation (at least) as otherwise it will not refer to any of the system-specific asset types.

D. *Concrete Model Generation*

Once the system is deployed and running, a SERSCIS concrete model generator is connected to the monitoring subsystem, and used to create a model of the running system. This reflects the current system composition, i.e. what assets of each type are involved, what security controls are in place to protect each asset, and which threat induced behaviours are exhibited by each asset. Once the assets are known, instances of the corresponding threat classes are added. The threat instances represent potential threats to the system which may or may not be active, and to which it may or may not be vulnerable.

The concrete model is then used as input for two further SERSCIS tools:

- a threat classification tool that decides to which threats the system is vulnerable, based on the control rules from the underlying SERSCIS dependability model;
- a threat activity estimation tool that decides how likely it is that each threat is active, based on the types of induced behaviour detected in the assets.

The output from these tools provides the following information at run-time:

- a list of potential threats to the system, classified according to whether the system is vulnerable to them;
- estimates of how likely it is that each threat is active;
- a description of each threat, including how severe its impact on the system would be;
- a list of controls that could be introduced to block or mitigate a threat, and whether these are available in the system.

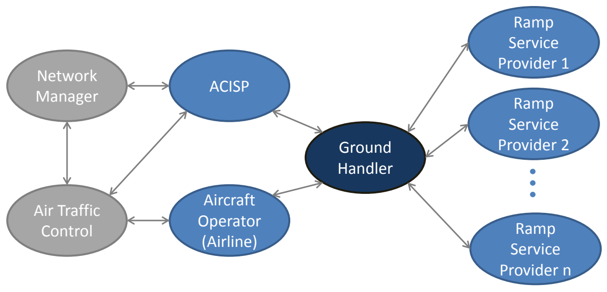

Figure 5. A-CDM Data Exchanges from a Ground Handler Perspective

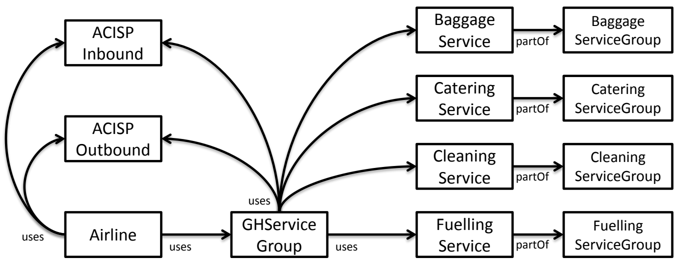

Figure 4. Abstract System Model of A-CDM

SERSCIS has developed a simple prototype decision support tool to display this information to a run-time system operator. Such a tool could be integrated into an existing user interface, and linked to the means to deploy a control (e.g. to block access for a misbehaving client, or blacklist a resource that is implicated in an attack). Details of the concrete model generation and processing and the decision support tool are beyond the scope of this paper and will be addressed in a future publication.

IV. VALIDATION

The SERSCIS semantic modelling approach was validated using an Airport Collaborative Decision Making scenario as outlined in Section I. The approach is stakeholder-centric, so the validation case study viewed the system from a Ground Handler perspective. A Ground Handler has the job of orchestrating services needed to get an aircraft ready for its next flight (e.g. refuelling, baggage handling, etc.). The data flows seen by a Ground Handler are summarized in Fig. 4. The ACISP is the airport information service supporting data exchange between the Ground Handler, Airlines and Air Traffic Control. The Ground Handler uses the ACISP to find out when incoming flights will arrive, and to pass on its predictions of when outbound flights will be ready to leave (i.e. the Target Off Block Time, TOBT). These are based on the inbound flight data, and planning information from ramp service providers at the airport. Eurocontrol's Network Manager (NM) uses the predictions to allocate slots in European airspace. The Ground Handler does not interact directly with NM or Air Traffic Control – this is typical in a multi-stakeholder system where each actor may interact with (or even be aware of) only a subset of the others. An attacker who wished to discredit air travel by disrupting it without causing any injuries might try to tamper with the data flowing to and from the ACISP.

The SERSCIS project could not inject such attacks into a real airport, so validation was carried out using a simple simulator based loosely on Vienna Airport. This simulated only airside operations (i.e. it did not include passenger check in and landside handling), but it did include all the actors shown in Fig. 4, so the effect of simulated disruptions on them could be detected.

The first step was to create an abstract system model from the Ground Handler perspective. The asset classes for this are shown in Fig. 5. Note that the NM and Air Traffic Control services are not included – this is because the model captures the system from the Ground Handler perspective. Since they have no direct contact with these services, they cannot detect instances of them in the monitoring data, so there is no point including them in the model. Note also that in this case, the Airlines not the Ground Handler are assumed to be Customers of the ACISP, so that the Ground Handler accesses ACISP data only when invited to do so by an Airline when they ask the Ground Handler to turn around an aircraft. The Ground Handler knows they will be given endpoints where they can read inbound flight data, or write outbound flight ready-time predictions, and these client specified resources are modelled as ACISP_Inbound and ACISP_Outbound sub-classes. From Fig. 5, it was possible to complete the abstract system model by automatically generating threat sub-classes to represent generic attack patterns applied to all the relevant combinations of asset types in the A-CDM network. This ran to over 100 system-specific threat classes. The generated threat classes were then refined by adding human readable description elements, severity levels and activity likelihood estimates. In addition, estimates of the probability that each threat would induce specific behaviours in the threatened and affected assets were compiled so detected behaviours could be used to infer threat activity in the run-time system.

The SERSCIS run-time[2] tools were then connected to the monitoring interfaces of the airport simulator and several scenarios executed:

- a sunny day scenario in which no attacks are carried out, and the airport functions normally;
- a scenario in which a bug is accidentally introduced in the Ground Handler's turnaround prediction software;
- a scenario in which a ramp service underestimates the number of crews it will need, causing it to turn up late when aircraft need servicing;
- a scenario in which a malicious intruder misdirects a ramp service, causing them to no show when the inbound flight reaches the stand and needs servicing;
- a scenario in which a malicious intruder tampers with the Ground Handler's estimates of aircraft readiness by overwriting them at the ACISP.

Details of the various scenarios and the changes for the different failure cases are given in the following section.

A. *Scenario details*

All scenarios use a schedule of 124 flights to be turned around during a day. All of those are regulated flights, i.e. all require a slot. Apart from the non-failure case, three degraded mode cases are listed and assessed below. The first case, the 'sunny day', provides sufficient resources for all ramp service providers. Hence none of the flights experiences a delay in turn-around. Case 2 is characterized by a reduction in the number of workers of the baggage handler to 23. In this case the baggage handler fails to honour several perform attempts and the flights experience substantial delays. In case 3 the number of workers is further reduced to 18. Hence even fewer perform attempts get honoured. In case 4, a second baggage handling resource is introduced (i.e. the Ground Handler starts with two SLAs for the provision of baggage handling services with different suppliers). If the primary baggage handler fails, an alternative service provider can replace it. If this arises as a one-off problem it can be handled by service orchestration (i.e. failing over to the replacement service), though some delay will still be experienced. If the primary supplier persistently fails, it is better to manage the situation by excluding it from further use. This was done by specifying a policy on the individual baggage handler resources (as seen by the Ground Handler). This policy sets the condition of the service to 'failed' if there is more than one failure, and deregisters the service so preventing it being available for selection. This addresses the immediate problem of a failing supplier, but it reduces the number of available options for baggage handling to one. A further policy is therefore needed that causes the resource manager to procure a new SLA with a replacement baggage service provider. Finally, case 5 implements a simulation of communication delays to demonstrate the effects of a denial of service attack on the ACISP. Due to the slow rate of communications, a number of flights take off outside their slot windows. No mitigation for this was considered in the run-time tests, as the only one that could be handled by the emulated components was to have redundant ACISP endpoints, which duplicates the mechanisms tested in Cases 1-4.

B. *Key Performance Indicators (KPI) used in the evaluation*

To evaluate the performance of a ramp service provider level, two KPI are considered: arrival reliability and service delivery duration. In the proof-of-concept evaluation the second KPI is a constant and disregarded. The first KPI on the other hand is taken into account to show the effect of a reduced number of workers. It is assumed that the reliability decreases if the number of workers available at a service provider is reduced. In the testbed this is measured by the number of "perform attempts" issued to the service provider. If a service provider has sufficient resources, every flight requires exactly one perform attempt that is honoured by the service provider; i.e., the number of perform attempts must be equal to the number of flights. If the provider cannot immediately honour a perform attempt due to a lack of workers, the perform attempts will be repeated. Thus the number increases beyond the number of flights.

The ramp service performance also has an effect on the ground handler's KPI. The ground handler's performance is characterized by two KPI: TOBT (Target Off Block Time) accuracy and TOBT stability. TOBT accuracy is derived from comparing the TOBT with a reference value (Actual Ready Time, ARDT). The mean square deviation between TOBT and ARDT is calculated for all flights departing in a day, where the value for TOBT is taken at TOBT freeze time, i.e. 30 minutes before TOBT. TOBT stability parameter measures how stable the prediction mechanism of the ground handler is. For this

[2] SERSCIS run-time tools downloadable at: http://www.serscis.eu/?page_id=317

purpose the average number of TOBT updates per flight is calculated. Since the actual delivery time of the ramp services is a distribution with a certain variation, e.g. dictated by the actual service requirements or by the service provider's resource trade-offs, TOBT accuracy will decrease with a reduction in the number of workers at the ramp service provider. TOBT stability expresses the number of updates to the TOBT required for each flight. In a sunny day scenario this number should be 1 or close to it, i.e. once a TOBT is issued it will not be changed. In degraded scenarios, however, a ramp service will deliver late due to a lack of workers. In the testbed the ground handler re-issues a TOBT whenever the estimate deviates from the previous value by more than 10 minutes. Hence the longer the ramp service delays its service delivery the more TOBT values need to be issued for a flight.

Both above-mentioned KPIs affect the number of take-offs outside the slot-tolerance windows (STW), which is an overall KPI for the A-CDM network. The value will increase in a ripple on effect of the ramp service provider's inaccuracy. If the ramp service provider fails to show up on the initial perform request, there is a risk that they will delay the turn-around of a flight and cause it to miss its slot. This effect, however, might be countered in part by the available slack in the turn-around process. A policy change that allows the ground handler to replace the service provider with an alternate in case it fails to respond to a perform request must reverse the above effect. Choosing an alternative service provider when the primary provider failed, will replace the overall service delivery reliability and thus result in fewer take-offs outside the STW.

Another KPI is applied to evaluate the overall CDM system's performance: the average number of slots issued per flight. Obviously, in the ideal case one slot is issued for a flight and this one is used subsequently. In less ideal situations delays in the turn-around prevent a flight from meeting its slot. Thus a new slot has to be issued, potentially wasting the previous one if it cannot be claimed by another flight. The longer the delay of a turn-around, e.g. induced by a lack of workers at a ramp service providers, the more slots must be issued for a flight[3].

C. *Experiment results*

The KPI obtained from the above scenarios are listed in Table 1. The results shown here clearly indicate that the chosen KPIs are meaningful for the testbed and the scenario and that verification and validation of the testbed succeeded. The KPI "perform attempts" was expected to increase if a service provider does not have sufficient resources to honour all requests in parallel. In this case some of the requests must be repeated, which means a larger figure. When introducing the possibility to choose an alternative provider in case the first one fails to honour requests, the total number of perform attempts should decrease again. This is exactly the behaviour of the testbed. Case 2 and also case 3 exhibit a significantly larger number of perform attempts than the sunny day case 1. With the introduction of an alternative service provider in case 4, the number of request drops close to the value of the sunny day case again. Note that it is still slightly larger than in the sunny day case, because additional perform requests are issued (and not honoured) while the alternative provider is being set up. Hence the KPI provides meaningful characteristics of the testbed and the testbed shows the expected behaviour.

TABLE I. KPI VALUES

Scenario	**Scenario**				
	Case 1	***Case 2***	***Case 3***	***Case 4***	***Case 5***
Baggage perform attempts	249	556	961	266	249
Average TOBT error	4 min	14 min	49 min	4 min	4 min
Average TOBT updates per flight	1	1.5	2.7	1	1
Average number of slots issued	1	1.5	2.7	1	1
Take-offs outside STW	0%	15%	31%	0%	13%

The TOBT-related KPIs reflect the quality of service delivery by a ramp service provider. With a decreasing number of workers in cases 2 and 3, the TOBT accuracy decreases as well and the required number of updates to this value per flight increases accordingly. When the ground handler has the option to choose an alternative service provider in case 4, the trend reverses and case 4 delivers the same performance as the sunny day case 1. Similarly, the number of slots issued per flight increases with the number of TOBT updates per flight. In case 4, in which TOBT does not get updated, only one slot is required as in case 1.

The last KPI, which was assesses in the testbed evaluation, is the percentage of take-offs outside the slot-tolerance window (STW). In the case of a sufficient number of workers at all service providers (case 1), none of the flights should miss its slot[4]. Hence the KPI must be 0%. With turn-arounds being delayed due to an insufficient number of workers at one of the service providers, flights will miss their slots and take off outside the STW. For this reason the value increases to 19% in case 2. When an alternative service provider steps in to take over the tasks from a failed provider as in case 3, turn-arounds are on time again. The percentage of missed slots falls back to 0% again. In the event of communication delays with the ACISP we see the percentage of take-offs outside the slot tolerance window increase in proportion to the delay. This is

[3] In the testbed the ground handler uses a simple strategy to update the TOBT. A new estimate for TOBT is calculated, and the TOBT is updated if the new estimate is more than 10 minutes after the previous TOBT. Note that in the current simulation every TOBT change automatically results in the issuance of a new slot. For this reason, the number of slots per flight is equal to the number of TOBT updates.

[4] The limited capacity of taxiways and runways might cause flights to miss their slots despite a timely turn-around, but this is not modelled in the testbed.

caused by delays in communications resulting in windows of opportunity to be missed.

D. Validation conclusions

Tests showed that if left untreated, the injected faults all cause the performance of the airport to be degraded: more flights take off late, and Eurocontrol wastes airspace by allocating slots that the Airlines cannot use. Moreover, all the faults lead to inaccurate forecasts to be passed to NM (in the last case the Ground Handler's forecast is accurate, but it is then corrupted at ACISP). However, the SERSCIS monitoring tools were able to detect these different threats and correctly diagnose them based on the monitoring data available. These experiments demonstrate that the SERSCIS approach provides a practical solution for security risk analysis in a dynamically composed, multi-stakeholder system. Security expertise is captured beforehand in the SERSCIS dependability model OWL classes, and these can be used by a system expert at design-time to create an abstract system model. This captures the structure of the system but does not specify how many assets it contains, where they are deployed or what security controls they have. The abstract system model provides inputs to run-time semantic monitoring tools, where the knowledge encoded in the OWL classes is used to automatically determine system threat activity and system vulnerabilities.

V. Future Work

The SERSCIS semantic modelling approach was validated using a subset of the European Air Traffic Management system. As outlined in Section I, it was applied to an Airport Collaborative Decision Making scenario at a major European airport. The authors are aware that A-CDM is a relatively small part of the overall ATM system, but it exhibits characteristics that make it well suited as a test bed:

- involvement of several stakeholders, coming from the core ATM as well as other domains;
- involvement of several technical systems interconnected by various communication infrastructures; and
- closely linked processes, in which the disruption of one sub-process without treatment will lead to major disruptions of the entire system.

These properties also apply to the entire European ATM system. At the same time, the sheer complexity and size of the European ATM system forestalls any security approach that tries to tackle the system as a whole. Rather a hierarchical methodology will be required. Hence future work will be to scale up the approach described here.

The SERCIS project ends in late 2012. In the final year, improvements to the SERSCIS core and dependability model have been made to make the model suitable for future research work. Improvements to the core ontology (Fig. 1) allow us to model physical and electronic attacks on airport connectivity and spaces. These include the ability of actors (including intruders) to move around the airport, the use of private networks to support communications with and in the airport, and the potential for physical or electronic attacks on communication assets as well as services that use them.

A final rationalization of the threat model has been to describe threats according to their target, action and consequence, based on [18]. This provides a basis for evaluating the coverage of the threat model and makes it easily extensible for different types of systems to provide threat-centric, run-time security analysis.

References

[1] Blanco, C., Lasheras, J., Fernandez-Medina, E., Valencia-Garcia, R., and Toval, A.: "Basis for an integrated security ontology according to a systematic review of existing proposals," Comput. Standards & Interfaces, vol. 33, no. 4, pp. 372-388, 2011.

[2] COBIT 4.1, IT Governance Institute, 2007.

[3] Matulevicius, R., Mouratidis, H., Mayer, N., Dubois, E., and Heymans, P.: "Syntactic and Semantic Extensions to Secure Tropos to Support Security Risk Management", Journal of Universal Computer Science (J.UCS), Vol. 18, N°6, pp.816-844, March 2012.

[4] EBIOS Methodology."Expression des Besoins et Identification des Objectifs de Sécurité". Specification white paper. www:http://www.ssi.gouv.fr/IMG/pdf/EBIOS-1-GuideMethodologique-2010-01-25.pdf (last accessed, February 2012)

[5] Fenz, S., and Ekelhart, A.: "Formalizing information security knowledge," in Proceedings of the 4th International Symposium on Information, Computer, and Communications Security (ASIACCS'09), Sydney, Australia, 2009.

[6] German Federal Office for Security in Information Technology (BSI), "IT Grundschutz Manual," 2005.

[7] Herzog, A., Shahmehri, N., and Duma, C.: "An ontology of information security," International Journal of Information Security and Privacy, vol. 1, no. 4, pp. 1-23, 2007.

[8] Hogganvik, I., and Stølen, K.: "A graphical approach to risk identification, motivated by empirical investigations". International conference on Model Driven Engineering Languages and Systems (MoDELS'06). 2006.

[9] ISO/IEC 27005:2011. Information technology – Security techniques – Information security risk management, International Organization for Standardization, 2011.

[10] Kim, A., Luo, J., and Kang, M.: "Security ontology to facilitate web services description and discovery," Journal on Data Semantics, vol. 9, pp. 167-195, 2007.

[11] Leonard, T., Hall-May, M., and Surridge, M.: "Modeling Access Propagation in Dynamic Systems", submitted to ACM Transactions on Information and System Security, March 2012.

[12] MEHARI "Risk management concepts and methods", Specification white paper.

[13] www:http://www.clusif.asso.fr/fr/production/ouvrages/pdf/CLUSIF-risk-management.pdf(last accessed, February 2012).

[14] Semantically Enhanced Resilient and Secure Critical Infrastructure Services, EC FP7 Project 225336, 2008-2012. See also http://www.serscis.eu

[15] SWRL "A Semantic Web Rule Language Combining OWL and RuleML" www: http://www.w3.org/Submission/SWRL/ (last accessed, February 2012).

[16] Touzeau, J., Hamon, E., Krempel, M., Gölz, B., Madarasz, R., and Alemany, J.: "SESAR DEL16.02.01-D03: SESAR ATM Preliminary Security Risk Assessment Method.," 2011.

[17] Vorobiev, A., and Bekmamedova, N.: "An ontology-driven approach applied to information security," Journal of Research and Practice in Information Technology, vol. 42, no. 1, pp. 61-76, 2010.

[18] Shirey, R.: "Internet Security Glossary," IETF RFC 2828, May 2000.