Resilient Critical Infrastructure Management using Service Oriented Architecture

Martin Hall-May, Mike Surridge
IT Innovation Centre
2 Venture Road
Southampton, SO16 7NP, UK
{mhm, ms}@it-innovation.soton.ac.uk

***Abstract*—The SERSCIS project aims to support the use of interconnected systems of services in Critical Infrastructure (CI) applications. The problem of system interconnectedness is aptly demonstrated by 'Airport Collaborative Decision Making' (A-CDM). Failure or underperformance of any of the interlinked ICT systems may compromise the ability of airports to plan their use of resources to sustain high levels of air traffic, or to provide accurate aircraft movement forecasts to the wider European air traffic management systems. The proposed solution is to introduce further SERSCIS ICT components to manage dependability and interdependency. These use semantic models of the critical infrastructure, including its ICT services, to identify faults and potential risks and to increase human awareness of them. Semantics allows information and services to be described in such a way that makes them understandable to computers. Thus when a failure (or a threat of failure) is detected, SERSCIS components can take action to manage the consequences, including changing the interdependency relationships between services. In some cases, the components will be able to take action autonomously — e.g. to manage 'local' issues such as the allocation of CPU time to maintain service performance, or the selection of services where there are redundant sources available. In other cases the components will alert human operators so they can take action instead. The goal of this paper is to describe a Service Oriented Architecture (SOA) that can be used to address the management of ICT components and interdependencies in critical infrastructure systems.**

***Index Terms*—resilience; QoS; SOA; critical infrastructure, SLA;**

I. Introduction

The SERSCIS ("Semantically Enhanced, Resilient and Secure Critical Infrastructure Services") project aims to support the use of interconnected ICT systems used to plan and manage operations in critical infrastructure such as airports. Failure or underperformance of any of the interlinked ICT systems owing to faults, mismanagement or (cyber-)attack compromise the ability of any or all the interconnected businesses to plan their use of resources, to maintain high levels of efficiency, and to continue providing information needed by others. The SERSCIS approach is to develop service-oriented technologies for creating, monitoring and managing ICT systems, allowing dynamic adaptation to manage changing situations, and to counter the risk amplification effect of interconnectedness. This can be evaluated by investigating failure scenarios caused by or impacting ICT systems, to show how SERSCIS provides more accurate risk assessments and thereby allows the impact of such failures to be minimised.

The technical approach used in SERSCIS is to treat each ICT component as a service, which has a specification of its dependability properties incorporated into a Service Level Agreement (SLA) with its users (humans or other services). This allows interconnections between services to be managed to maintain the dependability of the overall system in which they are used. The overall system is then able to specify its own dependability characteristics for the services it in turn provides to its users.

The key objectives of SERSCIS are the following:

- to devise ways to encode dependability commitments in a machine readable way, so they can be included in SLAs;
- to develop service governance mechanisms to ensure these commitments will be met, using autonomous monitoring and management of available resources (which may themselves be services), and adaptive workflow technology to orchestrate and utilise these resources;
- to verify the approach and its impact on best practice in an application scenario in air transportation.

A certain amount of 'machine intelligence' will be needed to understand SLA commitments and to govern and orchestrate services according to this approach. The project therefore makes extensive use of semantic modelling and reasoning methods to underpin machine understandable dependability specifications and decision-making. These models are also used to analyse ICT dependencies and threats and to provide decision support for human end-users during system design, deployment and operation.

II. Related Work

In its aims, SERSCIS builds on the successes of a number of projects. Research in project DIRC and its successor InDeED [1] raised the need to tackle dependability throughout the lifecycle of a system, including its interactions with humans. CRUTIAL [2] used Petri net and state modelling to understand the operation, behaviour and failure modes of CI in systems of systems arrangements. The IRRIIS project [3] is also improving CI dependability through Middleware Improved Technology and simulation in synthetic environments. SERSCIS will work with partners from IRRIIS, using and extending their work on the Taxonomy of Interdependencies for modelling cascading and escalating failures. There has been work on fault and sabotage tolerance using SLA for Grids [4], [5], which SERSCIS aims to build on to create resilient service-oriented infrastructures.

DeDiSys [6] aims to manage the trade-off between availability and meeting service constraints in loosely coupled Grid or P2P systems. SERSCIS aims to manage a similar dependability trade-off through a framework of automated service governance, including SLA negotiation, dynamic resourcing and run-time QoS and Quality of Experience (QoE) monitoring, as well as adaptive workflow composition, decision support and risk-aware system modelling techniques.

III. SERSCIS Components

The SERSCIS project proposes a service-oriented architecture with SLA-based management that builds on semantic models. This architecture comprises the following four main areas of component technology:

A. System Modelling

System modelling covers the development of (semantic) models of critical infrastructure requirements and behaviour, including the ICT components. These models are used throughout the SERSCIS framework as computer-understandable descriptions that provide the basis for automated, dependable ICT operation and feedback. The Web Ontology Language, OWL-DL, is used to model ontologies describing critical infrastructure aspects. Models contain valuable system knowledge as well as performance and pricing models [7].

B. System Governance

System governance covers the development of monitoring mechanisms and management actions and policies to maintain the dependability (including security and trust relationships) exhibited by SERSCIS-enabled services. The service-oriented infrastructure middleware GRIA [8] is used as a basis for the governance technological framework. The emphasis is on controlling available resources such that dependability requirements can be met, and where the system provides services, describing their non-functional characteristics in (semantically tractable) Service Dependability Agreements (SDAs).

C. System Composition

System composition covers the development of automated composition and orchestration of services to implement workflows at the system level to meet dependability requirements, using Service Dependability Agreement terms to control the selection of services from those made available by the system governance mechanisms. The emphasis is on development of dynamically adaptive workflow orchestration mechanisms based on semantic descriptions of workflows, dependability requirements, and (via SDA) available resources.

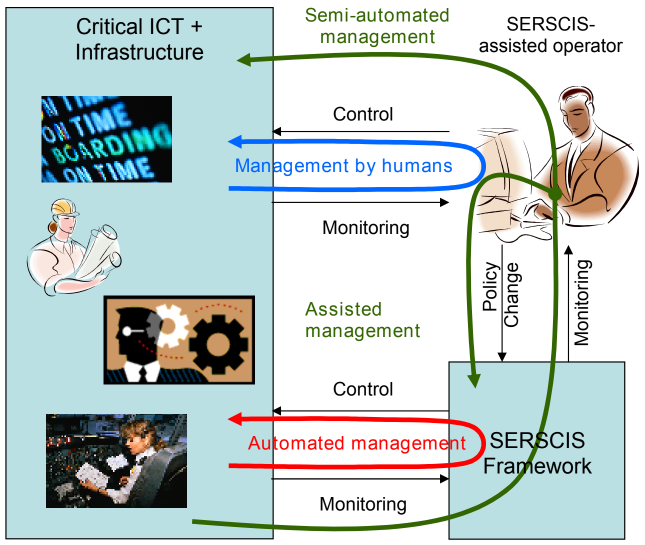

Fig. 1. SERSCIS Interactions with Critical Infrastructure and Operators

D. Decision Support

Decision support covers the provision of tools to present information to human operators of critical infrastructure and its associated ICT. SERSCIS will make use of autonomic service management models driven by high-level policies supplied by the operators, who may also be involved in initiating or carrying out management actions. It is therefore necessary to provide tools to help ICT implementers understand how a SERSCIS-enabled network will behave. Decision support tools aid operators that decide how to deploy applications in defining high-level policies in a SERSCIS-enabled framework, and in understanding the resulting (dynamically adaptive) behaviour that changing these policies has in an operating critical infrastructure.

IV. Critical Infrastructure Management

The key to SERSCIS is that it helps to manage risks and interdependency in the use of ICT systems within critical infrastructure, by adapting the ICT composition in response to events. Management in this context is concerned with sending controlling signals to the critical infrastructure ICT components when monitoring data indicates a need to do so.

Without SERSCIS, the ICT operators can of course monitor information supplied by ICT components used within the critical infrastructure, and take action when they consider this information indicates a need to do so. This provides a ‘humanised’ or ‘slow’ management loop between the operators and the infrastructure, indicated in the upper half of Figure 1. However, without SERSCIS, the interconnectedness of ICT systems used makes human decision-making very difficult, as problems (or actions taken) elsewhere may affect the quality of information available with which to make decisions. Moreover, any action a human operator takes may have an adverse impact elsewhere. Thus the risk of incorrect responses and the damage this might do are both increased. One of the key goals of

SERSCIS is to address this 'risk amplification' effect from ICT interconnectedness.

The SERSCIS framework monitors the critical infrastructure and uses a common Web service management interface to manage the dynamic composition of services and underlying resources. This process represents an automated management loop (marked as 'automated management' in Figure 1), in which the SERSCIS framework takes action as and when required by management policy. The management policy is defined by a SERSCIS-assisted operator and may be dynamically updated.

In addition to the above, SERSCIS governance components may conclude that some action is required that cannot be implemented autonomously. In such cases a signal is sent to the human operators. These signals may simply advise the operator that management action may be needed, leaving the human to decide what action to take (if any). In some cases, the signal may also propose the action, but leave the human to decide whether or how to carry out this action. This provides an 'assisted' management loop, which is also shown in Figure 1.

The operators can also provide control inputs to the SERSCIS framework, e.g. to change the models it uses to analyse the critical infrastructure, or to change the range of monitoring inputs or automatic actions available to it. These control inputs do not directly affect the critical infrastructure itself, but do change the way SERSCIS uses agile service composition models to support its future management. It is important to recognise that this facility to define SERSCIS models and policies is relevant even before the critical infrastructure and associated ICT is deployed, as well as during its operation. In the pre-deployment phase, this interaction can be used to support the ICT implementers, helping them to design and configure interconnected ICT systems in a way that allows risks to be managed by design as well as through subsequent adaptation.

V. High-level Service Architecture

The management of critical infrastructure ICT components and interconnections is considered an integral part of the overall SERSCIS approach. This is addressed by treating all ICT components as services, whose dependability can be specified via machine-understandable SDAs, allowing automatic and semi-automatic management of ICT dependability and interdependence, along with the rest of the critical infrastructure. Thus the SERSCIS Framework from Figure 1 contains the following:

- services whose purpose is to establish and keep track of the SLA that specify how critical infrastructure ICT services should interact;
- services to monitor each critical infrastructure ICT component and to ensure (through automated or assisted control actions) that it behaves in accordance with its SLA;
- services or other components to support dynamic adaptation of critical infrastructure ICT, especially its access control policies, interconnectedness and resourcing levels.

The high-level architecture provides a preliminary decomposition of these SERSCIS Framework facilities into software components, and explains how they interact with the critical infrastructure services they monitor and manage. To derive this architecture, we first consider how the lifecycle of SLA (which are new entities introduced by SERSCIS) should relate to the critical infrastructure ICT services (the application to which SERSCIS is being applied).

This will lead to a decomposition of the SERSCIS Framework needed by each service provider into components for managing the SLA, the critical infrastructure services, the resources available to those services, and the way services are composed and interact with each other.

VI. SLA Lifecycle

The most important architectural issue in developing for resilience is to have a common understanding regarding the lifecycle of SLAs, services and resources, a lesson learned previously in the GRIA project [8]. Unfortunately, these lifecycles are only loosely coupled, which leads to a large variety of possible scenarios as shown in Figure 2. SERSCIS builds on the SLA definition and lifecycle developed under the NextGRID project [9] as implemented in the GRIA middleware [10], [11].

In Figure 2, the following four main state models are illustrated:

- the service itself: its definition including its implementation as a piece of software and description via models of its behaviour, configuration (by specifying management models) for use at a service provider, and deployment to make it executable using allocated resources;
- resources: their acquisition by the service provider, and their allocation (or deallocation) for use by a particular service according to the provider's management policies;
- the SLA offer (or template): the specification by the service provider of terms (including dependability commitments) under which they can make the service accessible, and the publication of these terms to potential consumers;
- the interaction between a service provider and a service consumer: this includes the initial request for an SLA based on a published template/offer, the granting (or otherwise) of the request leading to an SLA being made between the two, and enabling access to the service under this SLA.

The solid lines in Figure 2 represent state transitions in each of these processes. The loose coupling between the processes is indicated by the dashed arrows linking different processes, which signify that one process must be in a given state for the other process to enter a given state. For example, consumer interactions have a state where the service is accessible to the consumer. This state can only be reached if the service is deployed at the service provider, which is denoted by the dashed arrow from the 'Service Deployed' state in the service lifecycle to the 'Service Accessible' state in the consumer interaction lifecycle. To summarise these couplings:

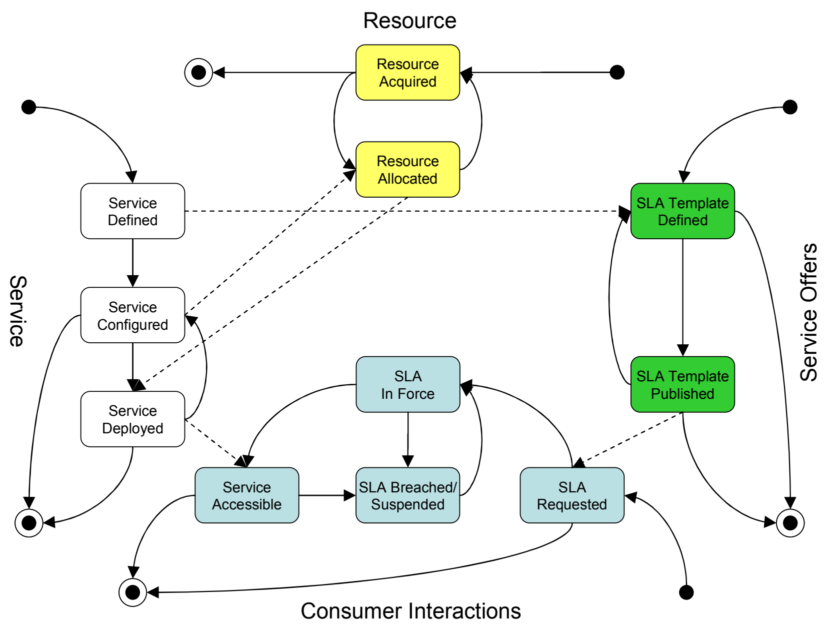

Fig. 2. Lifecycle Models

- the service must be defined before an SLA Template (specification of offered dependability commitments) can be formulated;
- the SLA Template must be published before potential consumers can request an SLA embodying its commitments;
- the service must be configured with a management policy before resources can be allocated to it;
- at least some resources must be allocated before a service can be deployed; and
- a service must be deployed before it can be made accessible to consumers.

The main purpose of the SERSCIS Framework is to support the use of SLA in consumer interactions with a service, including the introduction of Service Offers and SLA creation, and to facilitate coupling between consumer interactions, service management and resource management, keeping them all consistent with the dependability commitments made in the SLA [12], [13].

The loose couplings represented by dashed arrows in Figure 2 allow for many possible scenarios. For example, at one extreme the provider may procure and allocate resources and deploy the service long before they start to define SLA templates to specify their dependability commitments. This may be a common scenario where a service is already operating and the provider decides to introduce SERSCIS technology to manage its dependability.

At the other extreme, the provider may acquire the service implementation (i.e. the software), then create and publish SLA templates, and wait until they have agreed an SLA with a consumer before even configuring the service. This scenario must be followed where service providers are willing to negotiate over the exact service levels and other specifications with consumers. In such cases, the SLA template includes service metrics but not agreed levels, and the policies to regulate services have to be configured after the SLA is negotiated. In practice, it is difficult and/or costly for a service provider to manage resources and services against a large number of potentially different commitments [14]. In SERSCIS we currently consider only the case where the management policy is defined first, and a limited (discrete) set of SLA templates are derived from it, which encode non-negotiable dependability commitments that the management policy is able to meet.

Finally, although Figure 2 does not explicitly show cardinality, in practice the couplings can be many-to-many. For example, one SLA may cover several services and one service may be accessible to different consumers under many different SLA. Similarly, services may need multiple resources, and multiple services may share resources, and so on. Keeping track of these relationships is one of the key requirements for a dynamic, (semi-)autonomous management architecture.

VII. Service Provider Architecture

Given the above model for the lifecycle, it is possible to define a high-level service provider architecture for SERSCIS Framework components to manage services (and consumer interactions), SLA and resources during each phase of their respective lifecycles, as shown in Figure 3.

The software components include the service interface and workflow orchestrator, along with SERSCIS governance components to support the management of services, resources and SLA. These components are:

- a security Service Access Control Point, able to restrict access to the service according to a security policy that must be dynamically updatable [15];

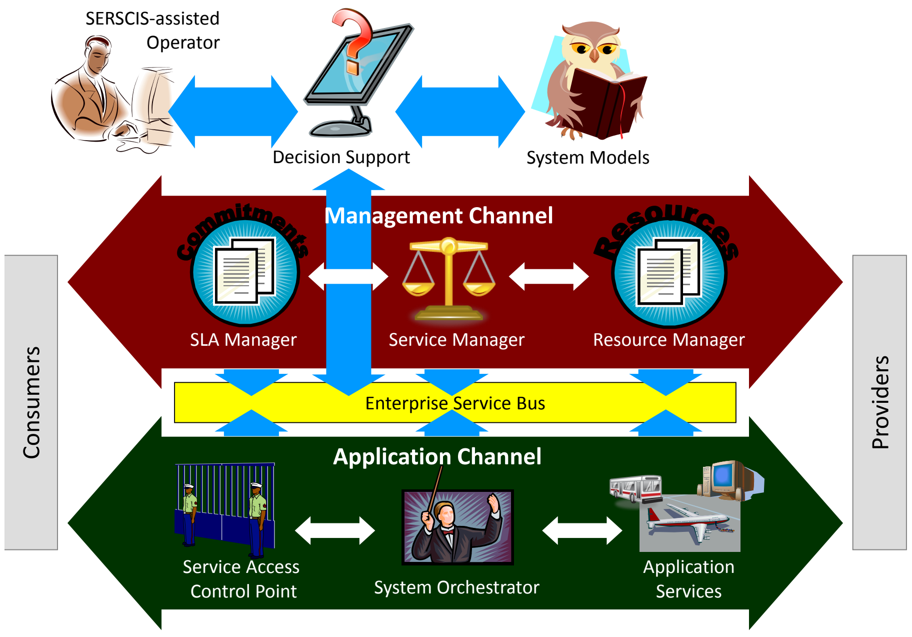

Fig. 3. High-level Service-Provider Architecture

- an SLA Manager that hosts SLA templates and handles requests from clients for SLA based on them: the SLA manager grants new SLAs, provides information to the clients on their status, and may terminate existing SLAs if required;
- a Service Manager that monitors the Quality of Service (QoS) and Quality of Experience (QoE) reports and analyses them using the service model against the provider's service commitments and management policy, and initiates appropriate management actions;
- a Resource Manager that handles the acquisition/allocation and removal of resources, and maintains a registry [16] of these resources in which the orchestrator can discover resources when it has to execute a service workflow.

Governance components are deployed at each service provider that wishes to manage some or all of their services. In this way, governance remains distributed and light weight, without the need for a monolithic governance 'agent'. Components within each service provider communicate via an Enterprise Service Bus (ESB), which allows them to be loosely coupled. The relationship between these information sources and the lifecycle models from Figure 2 are as follows:

- the system model and management policy are created by software developers and system administrators using SERSCIS ontologies, supported by SERSCIS modelling and decision support tools;
- service commitments are created or removed when an SLA is negotiated or terminated, based on an SLA Template;
- the service access policy is dynamically generated when the service is deployed, and updated when an SLA is negotiated or terminated;
- the QoS event stream is generated by a service interface (or wrapper to a black-box service [17]) based on what is provided by or can be measured by the service;
- the QoE event stream is generated by the orchestrator when using other services based on what is measurable by or meaningful to the consumer of that service. Note that while QoS is a measure of what the service provides to its consumers, QoE is a measure of how its resources (including other services) have performed; and
- resource registry entries are created or removed when resources are procured and allocated or de-allocated.

Actions taken by the Service Manager may induce changes in the available resources, according to the management policy of the provider. For example, it may ask the Resource Manager to negotiate additional SLAs to provide greater resource volume or redundancy, or allocate more in-house resources. Alternatively it may seek to manage demand on the service. For example, it may simply revoke the SLA template so that no new SLAs can be granted by the SLA Manager, preventing any further increase in the level of service commitments. It may go further, by updating the security access policy to restrict access if the SLA allows this. It may even ask the SLA Manager to breach or to terminate the SLA. The service itself may also support some management actions to influence its behaviour, and the Service Manager can use these if the management policy allows it. These actions can in principle be taken automatically, implementing the 'automated' management loop from Figure 1.

SERSCIS components also provide administration interfaces giving operators direct access to the Service Manager, SLA Manager and Resource Manager, and to the associated models, policies and monitoring data. In many situations, the management policy will instruct the Service Manager to inform an operator that action is needed, leaving a human to decide whether and if so how to act. The operator can then implement their action directly on the critical infrastructure, by accessing SERSCIS components or by changing the models and policies used to control them. This provides the 'assisted' management loop in Figure 1.

Finally, SERSCIS provides decision support facilities to help operators understand the behaviour of the system, based on events generated by the components shown in Figure 3. As noted above, this facility will also be useful prior to operation of the SERSCIS-enabled ICT infrastructure, allowing an analysis of risks using the service model during the design phase, and providing insights into the commitments that can be made and the management policies needed to meet them. These tools will also play a role in auditing processes used to analyse and verify/improve the management of the infrastructure.

A number of application services are managed by a single governance component, comprising a service manager, SLA manager and resource manager. The orchestration component coordinates workflows involving 'local' resources as well as resources encapsulated in services from other organisations. A SERSCIS-assisted operator has an overview of all the services within a given organisational/operational domain. The operator's situational awareness is supplemented by a decision support component, which uses a model of the whole system. The 'whole system' may include details of concepts outside of the operator's immediate control, i.e. involving the repercussions that local actions will have on the wider system of systems. These system models may be shared across organisational boundaries.

VIII. Layered Approach

Figure 4 shows three concentric dependability management loops that arise from the SERSCIS architecture. Similar to Kramer and Magee's model for autonomic systems [18], this allows runtime information to propagate up to system decision makers and governance actions to flow down to control service execution. System modelling and decision support components are closely linked and may run synchronously; however, with respect to governance and workflow components, the execution is entirely asynchronous.

The inner runtime loop runs at the speed of the application services, selecting the most appropriate resources and executing the workflow. The governance loop monitors and manages resources against SLA commitments, at a slower rate. This loop has to be asynchronous with respect to application responses because it is often infeasible for the orchestrator to wait while a new SLA is negotiated and approved. Finally, the outer loop runs even more slowly so SERSCIS-assisted human operators can be involved, making changes to autonomic governance policies in response to changes in key performance indicators (KPI).

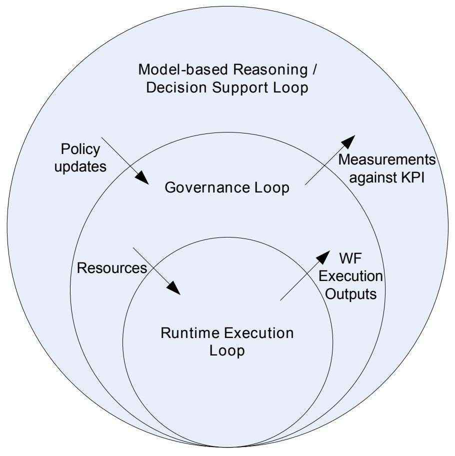

Fig. 4. Concentric Dependability Management Feedback Loops

It is this separation of decisions, which simplifies the work that each set of components must do, that is the guiding principle of SERSCIS. Such a layered approach to the architecture was taken in recognition of real-world business practices and performance.

IX. A-CDM Test Case

Airport Collaborative Decision Making (A-CDM) brings together the main airport partners — Air Traffic Control (ATC), Airport Operators, Airlines, Ground Handlers, and Europe's Control Flow Management Unit (CFMU) — to share operational data. Such information sharing is fundamental to achieving a common situational awareness, which improves decision making [19]. The SERSCIS proof of concept test case focuses on the process of 'turning round' an aircraft from moment it arrives 'in block' to when it taxis out for take-off.

Figure 5 shows the services involved in the proof of concept test case and how SERSCIS components are deployed. Ovals represent the organisational/operational boundaries and thus represent a service provider. The CDM owner is a service provider and an Information Sharing Platform (ISP) service. SERSCIS system models, decision support, governance and orchestration components are installed at the CDM owner. SERSCIS components at the CDM owner can only directly access local services and resources. Access to external resources is via service requests.

The Ground Handler is another service provider, which is also SERSCIS-enabled. This allows us to explore the inter-domain aspects of SERSCIS in this case study. Other service providers (Airline, Fuel and Catering companies) may or may not have SERSCIS. Figure 5 does not show all the SERSCIS features at all these providers where they may be used.

The fuel service in the test case is assumed to be one of many available in the service registry. The exact one that is used is chosen at runtime based on availability, price or any other QoS criteria specified in the SLA. The relationship with the catering company is managed by a long-term contract and hence is likely to be the only resource in the service registry from which the workflow composer can choose.

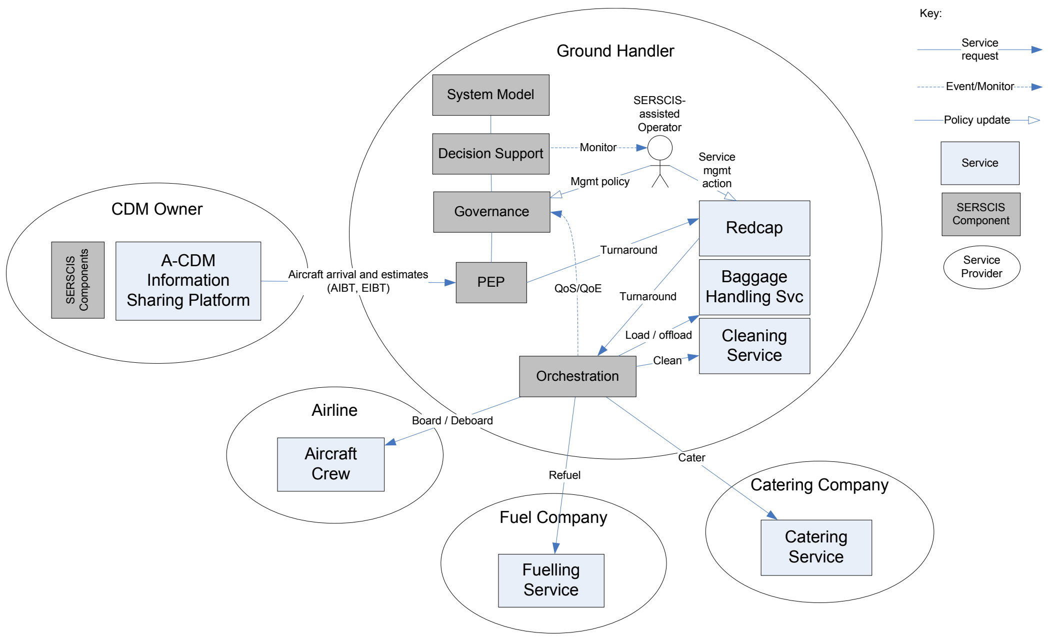

Fig. 5. SERSCIS Components and Services in the A-CDM Test Case

If the other service providers in Figure 5 (Airline, Fuel company, Catering company) are also SERSCIS enabled, their services will have their own governance and orchestration components, which provide automated management, and their own SERSCIS-assisted operator, who provides human and assisted management (the latter supported by SERSCIS decision support tools). Without SERSCIS components, SLA and resource management (including maintaining a repository of available resources) and workflow orchestration are a manual process, which may not even be consistent between organisations. With SERSCIS, not only are these processes computer-assisted, but they can be made mutually consistent between service providers by sharing aspects of the overall system model and ontology.

A. *Turnround Scenario*

The following steps represent one possible sequence of events describing the process of turning round an aircraft:

1) CDM owner defines a service ‘Information Sharing Platform’ with a method ‘turnaround’
2) CDM owner uses workflow editor (orchestration component) to create an abstract workflow that represents the steps to prepare an aircraft for pushback (e.g. deboard passengers, offload bags, clean aircraft, refuel etc.)
3) CDM owner defines (a set of) SLA templates offering specified QoS on the service (e.g. turnaround in less than 90 minutes).
4) SERSCIS-assisted operator sets management policies on governance components for service management (resource and SLA management)
5) Resource manager (governance component) acquires SLAs to use externally provided resources (e.g. refuelling, baggage handling) and stores them in the service registry
6) Local ATC signals the arrival of an aircraft, whereupon the ground handler invokes the ‘turnaround’ service. The service access control point (governance component) determines whether access should be allowed under the provided SLA.
7) If access is allowed, the workflow composer (orchestration component) selects the appropriate abstract workflow and concretises it using resources available in the service registry
8) The workflow executor (orchestration component) executes the concrete workflow, invoking the services (offload, deboard, clean, refuel, etc.) and records QoE
9) Events from the SLA manager, resource manager and orchestration components are monitored by the service manager (governance component) and appropriate management actions are taken
10) Events are also elevated to the decision support component and processed/aggregated to inform the SERSCIS-assisted operator of system performance with respect to key performance indicators
11) The SERSCIS-assisted operator takes assisted management action on the SERSCIS components (e.g. updating management policies so as to prevent further SLA offers) or directly on the service components (e.g. stopping the baggage handling service)

B. *Service Disruption Example*

To elaborate the steps to recover from a disruptive event, consider the event of a passenger ‘no-show’ at the airport. A ‘no-show’ occurs when a passenger has checked in but does not arrive at the gate when boarding commences. Boarding begins but cannot complete. After a certain tolerance, the passenger's baggage may need to be unloaded in order for the aircraft to make its allocated take-off slot.

In the above situation SERSCIS can help in the orchestration of the resources needed to unload the baggage. This may require the workflow to adapt to the situation. The system composer may also have to dynamically select a new baggage handling agent to unload the baggage if the current service is unavailable (e.g. if it needs to service another aircraft). The SERSCIS Framework manages the available ICT in the following sequence of events:

- The disruption begins with a prolonged boarding time, while the aircraft crew wait for the missing passenger
- After a given time, the crew informs the ground handler that the passenger has not boarded
- The system orchestrator executes the exception path in the ground handling workflow and calls the baggage service to unload the passenger's bags
- The original baggage service is unavailable at such short notice, so the system orchestrator uses an alternative service provider with which the resource manager has previously negotiated an SLA. The SLA guarantees responsive service delivery with no pre-scheduling required

SERSCIS will allow designers and operators to analyse how well their (interconnected) ICT systems cope with disruptions such as the above scenario in design, deployment and operation as well as during operational auditing. This will lead to ICT systems that can exploit connections while maintaining an acceptable level of resilience by managing interdependencies.

Prototypes of all components described in this paper have been implemented in order to demonstrate the proof of concept outlined above. A full validation of the concepts based on an enlarged application scenario will be carried out in the next stage of the project.

X. Conclusion

A high-level service architecture has been introduced for a SERSCIS Framework to support and manage critical infrastructure ICT services. This allows risks in operating critical infrastructure to be managed by augmenting current 'slow' human-initiated management with automated and assisted management of ICT components and services. The service architecture maps to the lifecycle of an SLA: the definition of a service, the description of SLA templates, the negotiation of SLA, and the selection and invocation of services as part of a workflow (including renegotiation/revocation of SLA). To create and execute such a workflow comprising distributed services, while maintaining an overall level of dependability, requires models of information regarding performance characteristics, threats and countermeasures. The SERSCIS architecture takes a layered approach to help solve the conceptual integration, while the use of common components (a resource registry and ESB) addresses the pragmatics of integration and allows for loose coupling between the components.

Acknowledgement

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

References

[1] I. Sommerville, T. Storer, and R. Lock, "Responsibility modelling for contingency planning," in *Proceedings of the Workshop on Understanding Why Systems Fail*, June 2007.

[2] P. Verissimo, Y. Deswarte, A. Bondavalli, N. F. Neves, A. A. El Kalam, A. Daidone, and M. Correia, "The CRUTIAL architecture for critical information infrastructures," *Architecting Dependable Systems*, vol. 5135, pp. 1–27, 2008.

[3] C. Balducelli, A. Di Pietro, L. Lavalle, and G. Vicoli, "A middleware improved technology (MIT) to mitigate interdependencies between critical infrastructures," *Architecting Dependable Systems*, vol. 5135, pp. 28–51, 2008.

[4] M. Hovestadt, "Fault tolerance mechanisms for SLA-aware resource management," in *Proceedings of 11th International Conference on Parallel and Distributed Systems*, vol. 2, 2005, pp. 458–462.

[5] S. Naqvi, S. Mouton, P. Massonet, G. Silaghi, D. Battre, M. Hovestadt, and K. Djemame, *From Grids to Service and Pervasive Computing*. Springer, 2008, ch. Using SLA Based Approach to Handle Sabotage Tolerance in The Grid, pp. 153–162.

[6] L. Froihofer, K. M. Goeschka, and J. Osrael, "Middleware support for adaptive dependability," in *Proceedings of 8th International Middleware Conference*, ser. LNCS, vol. 4834. Springer, 2007, pp. 308–327.

[7] S. Benkner and G. Engelbrecht, "A generic QoS infrastructure for grid web services," in *Proceedings of International Conference on Internet and Web Applications and Services/Advanced International Conference on Telecommunications*, 2006, p. 141.

[8] M. Surridge, S. Taylor, D. De Roure, and E. Zaluska, "Experiences with GRIA — industrial applications on a web services grid," in *First International Conference on e-Science and Grid Computing*, 2005, pp. 98–105.

[9] P. Hasselmeyer, H. Mersch, B. Koller, H. N. Quyen, L. Schubert, and P. Wieder, "Implementing an SLA negotiation framework," in *Proceedings of the eChallenges Conference (e-2007)*, 2007, pp. 24–26.

[10] M. Boniface, S. Phillips, A. Sanchez-Macian, and M. Surridge, "Dynamic service provisioning using GRIA SLAs," in *Service-Oriented Computing — ICSOC 2007 International Workshops, Revised Selected Papers*, ser. LNCS, vol. 4907. Springer, 2009, pp. 56–67.

[11] M. J. Boniface, S. C. Phillips, and M. Surridge, "Grid-based business partnerships using service level agreements," in *Proceedings of Cracow Grid Workshop*, 2006.

[12] M. Hovestadt, "Service level agreement aware resource management," Ph.D. dissertation, University of Paderborn, Germany, October 2006.

[13] P. Hasselmeyer, B. Koller, L. Schubert, and P. Wieder, "Towards SLA-supported resource management," in *Proceedings of 2nd International High Performance Computing and Communications Conference*, ser. Lecture Notes in Computer Science, vol. 4208. Munich, Germany: Springer, September 2006, pp. 743–752.

[14] P. McKee, S. J. Taylor, M. Surridge, R. Lowe, and C. Ragusa, "Strategies for the service market place," in *Proceedings of GECON 2007, "Grid Economics and Business Models"*, ser. LNCS, vol. 4685, 2007, pp. 58–70.

[15] M. J. Boniface, T. A. Leonard, M. Surridge, S. J. Taylor, L. Finlay, and D. McCorry, "Accessing patient records in virtual healthcare organisations," in *Proceedings of eChallenges*, Ljubljana, Slovenia, 2005.

[16] U. Radetzki, M. Boniface, and M. Surridge, "Contextualized B2B registries," in *Proceedings of Service-Oriented Computing (ICSOC 2007)*, ser. LNCS, vol. 4749. Springer, 2007, p. 506.

[17] A. Michlmayr, F. Rosenberg, P. Leitner, and S. Dustdar, "Advanced event processing and notifications in service runtime environments," in *Proceedings of 2nd International Conference on Distributed Event-based Systems*. ACM, 2008, pp. 115–125.

[18] J. Kramer and J. Magee, "A rigorous architectural approach to adaptive software engineering," *Journal of Computer Science and Technology*, vol. 24, no. 2, pp. 183–188, 2009.

[19] *Airport CDM Guide*, EUROCONTROL, March 2009.