A Goal-Oriented Approach to Determining and Sharing Risk Data in Distributed Interdependent Systems

Pete Burnap[1], Yulia Cherdantseva[1], Andrew Blyth[2], Peter Eden[2], Kevin Jones[3], Hugh Soulsby[3], and Kris Stoddart[4]

[1]School of Computer Science and Informatics, Cardiff University, UK
[2]Faculty of Computing, Engineering and Science, University of South Wales, UK
[3]Cyber Operations, Airbus Group Innovations, UK
[4]Department of International Politics, Aberystwyth University, UK

***Abstract*—**While the risks induced by system dependencies have been studied; little is known about modelling complex collections of supposedly independent systems at different geographical locations, which are in reality interdependent due to sharing often-unrecognized common elements. It could be argued that any risk analysis of a large infrastructure that does not take account of such interdependencies is dangerously introspective. We present a top-down, goal-to-dependencies approach to modelling and understanding such Complex Systems, which uses secure, distributed computing protocols to share risk data between the risk models of interdependent systems. We present a Bayesian-sensitivity measure of risk, which is both intuitively satisfying and accords with everyday notions of risk. The core benefit of this approach is to capture dependencies between systems and share risk data such that failure of an entity along the 'supply chain' can be rapidly propagated to those who depend on it allowing them to calculate the likely impact and respond accordingly.

Keywords-risk assessment;vulnerability analysis;senstivity analysis;failure mode analysis;interdependent systems;distributed systems

I. INTRODUCTION

In the twenty-first Century, an individual, group or enterprise depends on a range of entities, including telecommunications, supplies of energy, transportation, human resources, and governance, provided by private and public enterprises, and agencies. Other entities may in turn depend on them to form a 'supply chain'. Collectively this results in a Complex System, within which entities often operate autonomously, at different geographical locations, and there may be little coordination from an overarching authority. Such complexity often results in an entity's owner not knowing on what they depend, or the full range of entities that depend on it. In short, it cannot collate the risks associated with its 'supply chain'.

Risk assessments undertaken using enterprise risk assessment methodologies (e.g including ISO/IEC Standard 27005 [1], CRAMM [2], Octave [3]) are often carried out introspectively and define risk for individual entities of Complex Systems. A risk assessment for a hospital, for example, might highlight entities such as staff availability, medical equipment, records management systems, data communication networks, power stations, data centers, and transport infrastructures, to name a few. Some of these entities, such as staff and medical equipment, are self-provisioned. If we regard risk as the failure to achieve the provision of these entities in full working order, then the risk for self-provisioned entities can likely be defined as it is known what they depend on to be successfully deployed. However, some of the entities may be provided from outside the hospital by *external entities*. The risk of external entities not being provided is unknown because the elements they depend on are not comprehensively understood by the principal entity. However, they are probably known to the external entity, which may have performed its own risk assessment. The problem is, the two risk models are unconnected, and there is no communication between the principal and external entities pertaining to shared risk data, even though they are effectively part of the same Complex System. What makes entities interdependent is their vulnerability to the failure of elements not directly included in their individual risk models. Indeed, the failure of elements in an external entity may trigger a rapid, unanticipated and catastrophic cascade of repercussions to other, nominally separate principal entities. The role of interdependencies means that for the purpose of risk analysis, it makes sense to regard many notionally distinct entities as parts of a larger *Complex System*, a notion supported by [10].

Interdependencies mean that static one-off risk assessments on isolated entities may not suffice to manage risk within and between notionally distinct systems. Isolated models need to be linked, somehow, to build a risk model

for a Complex System. However, linking introspective risk models from distributed entities is non-trivial. Firstly, to link models requires the identification of elements common to both models. Secondly, to determine the risk for the dependency requires an understanding of the dependencies of the common element. Most existing risk modelling methods use a bottom-up approach - from failure to consequence. The principle question in this case is "What could go wrong?", which is a subjective view of risk and may lead to a failure to recognise dependencies and unpredicted circumstances. Finding elements common to multiple independent risk models using method is extremely difficult, as it is focused on independent failure modes and not common objectives. In contrast, we propose the question "What does the element depend on to be successful?". When an entity is modelled for risk using this approach, at some point the dependency on an external entity emerges, and this is when it might be more beneficial to the understanding of external dependencies if both entities are regarded as parts of a larger Complex System.

Working from the top down in this way (i.e. from the goals of a Complex System to its dependencies) allows failure modes to be quickly determined in the event of failure anywhere in the Complex System. This amounts to answering repeatedly the original question "What does the element depend on to be successful?" across multiple isolated models. The difference is subtle but important and is equivalent to traversing a tree recursively from the trunk (top-down), which generates all the branches, versus tracing a path from a given branch to the trunk (bottom-up), which generates only one path.

This paper presents a top-down, or goal-oriented modelling approach that enables analysts to identify which dependencies their goals are most sensitive to, thereby enabling a prioritised risk management strategy with informed investment and allocation of countermeasures; to identify the most likely sources of failure given an actual or assumed failure within the entity, enabling effective use of time in identifying the problem and returning the system to full operation; and to identify other autonomously managed entities on which it depends. These determinations can be rendered more accessible by the use of Bayesian methods and failure mode analyses.

Our risk-modelling paradigm is able to provide both a statistical assessment of various definitions of risk and also a real-time representation of the current status of all the entities in the model. The inherent scaling problem of handling potentially large Complex Systems is facilitated by a novel, distributed middleware and supporting service-oriented architecture which simplifies the construction of the original model and permits the sharing of real-time functional status data between principal and external entities, including the secure fine-grained acquisition of distributed status data, and updating of the conditional probability tables needed for meaningful statistical analysis.

II. MODELLING DEPENDENCIES WITHIN COMPLEX SYSTEMS

We present an approach to Dependency Modelling (DM) as a method of determining the risk to an enterprise through the use of a graphical model. We use *enterprise* to denote the *providing* entity under consideration or its owner, be it for instance a government, a business, a group etc.

The method for building a model is as follows. A top-down model of the enterprise is built from an overall goal to its first-level dependencies (see Figure 1), and then to next-level dependencies, and so on. Dependencies are assigned a number of possible states (failure or success being the simplest), and a conditional probability of being in those states. Repeating this for sub-processes produce a tree or graph-based model. For brevity we will not go into the technical specifics of *how*, but this model *can* be interpreted both as a Bayesian network and as a failure-mode model (see [8,9]). The probabilities are used by a Bayesian analysis engine to determine a range of sensitivities or vulnerabilities illustrating which elements in the model are most pivotal to its success, while a mode-analyzer intuitively illustrates the most likely cause of failure in a model, given that a hypothetical (for simulation purposes) or actual (for response purposes) failure has occurred. Individual models can be created for a number of *provided*

entities owned by an individual enterprise, these can be linked and scaled to model the entire enterprise, and finally, they can be linked to interdependent entities within other enterprises, to build a model of a Complex System through a scale-management method we call *Zooming*.

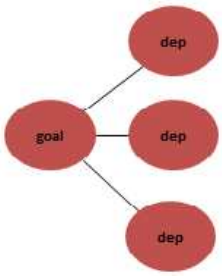

Figure 1 - Goal with dependencies

Central to the method is the role of goals. Loosely speaking risk is about failing to achieve goals. In DM terms, risk in complex systems relates to the achieving, or the failure to achieve, the provision of the fully functioning entities that goals depend upon. We refer to the successful provision as a goal. Without a goal there is no risk, and changing goals changes the threat landscape.

Every element in a Dependency Model is an abstract goal. We are not for example interested in having (say) an access control system for its own sake, but rather because we want to keep hackers out of the network. The goal would therefore be to *Keep Hackers out of the System*, not to *Have an Access Control System (ACS)*. It quickly becomes clear that all goals are abstract in this sense. Having an ACS does not necessarily achieve the goal. The successful defence of the system may be dependent on a number of factors, including technology (such as an ACS), expert knowledge (how to configure the ACS), and people (abiding by the expected security policy). To require the mere existence of the ACS leads to "box-ticking" risk management approach - this is unlikely to see the goal achieved.

Each element in the model can be regarded as a goal whose dependencies are drawn to its right, it may also be a dependency of one or more elements drawn to its left. The model can be extended in this way at will, and the process can be terminated at any stage by merely regarding some elements - for the purpose of the model - as being external entities, whose dependencies are unknown. Examples of the latter might include the reliability of the public electricity supply or a supply of clean water. All we need to know about these elements are their statistical properties to feed into the Bayesian engine when the model is complete, and we will call them *uncontrollables*. Ultimately much of the risk in Complex Systems springs from these *uncontrollables*, hence the terminology.

III. ANALYZING THE MODEL

Bayesian network literature [8,9] often describes relationships through a dynastic metaphor. Our *dependencies* would be called the *parents* of goals, and our goals the *children* of dependencies. The model can be viewed in a number of different ways. It can be used to determine failures modes, and if we include conditional probability tables (CPTs) it can be treated as a Bayesian network, whereupon a Bayesian analyser can determine the probability that each element will be in each possible state. To make Bayesian inferences we need to know the statistical relationships between parent and child. The degree to which a goal is achieved is expressed as the state of the goal element. Each element may have any number of possible states greater than one. A common number is two,

typically associated with failure and success, though it is sometimes convenient to have more. A power station for example, could be able to provide varying levels of supply. In accordance to the concept of the success model, states are labelled according to some value judgement. Typical names for pairs of states would then be bad-good, failure-success, red-green, no-yes or 0-1.

If we incorporate real-time updating (via sensor data collected via an Internet of Things model, for example), DM can provide a "living" risk model where the impact of changes in state can be immediately seen in the rest of the model. For example the reduction in risk due to the introduction of a countermeasure can be used to decide the cost-effectiveness of the countermeasure, or by experimenting with various economies it is possible to find one that entails least extra risk.

The relationship between a goal and its parents is expressed by a conditional probability table (CPT), which lists the probability of the goal being in each state according to the states of its parents. If the nature of the relationship boils down to a logical AND or OR function then the CPT has a simple structure consisting merely of 0s and 1s, but more subtle relationships can be accommodated. If the goal is an uncontrollable then this table, in an isolated model, is just a list of the probabilities it will be in each possible state. The power of DM in Complex Systems becomes more evident when the states of uncontrollables are dynamically pulled into the model from other distributed models that contain the actual states and dependencies of these uncontrollables, as described in Section V.

IV. COMPARISON TO OTHER MODELLING APPROACHES

A brief introduction to traditional risk modelling techniques is available in [11], and several reviews of risk assessment in critical complex system exist [13-16]. In summary the most popular approaches, include Fault Tree Analysis [4], Attack Trees [6], Cause-consequence/effect diagrams [7], Bayesian networks [8,9] and CORAS diagrams [12]. We add to this, enterprise risk assessment methods such as ISO/IEC Standard 27005 [1], CRAMM [2], and Octave [3].

The enterprise risk assessment methods work by defining a list of assets, determining their vulnerabilities, assigning threats to those vulnerabilities, and calculating the impact of a threat exploiting a vulnerability. Little attention has been given to dependencies between assets, or cause and impact throughout the rest of the enterprise should vulnerability attack occur, though this has begun to some extent with ISO/IEC 31000:2009 [5], which begins to frame risks in a goal-oriented manner but without full attention to dependencies. These considerations are addressed in this paper.

Fault Trees, Event Trees and Attack trees can be used to focus on identifying potential faults, events or attacks that could cause a Complex System to fail. Cause-effect diagrams and CORAS threat models also work from failure to potential cause and subsequent effect. These are all, in principle, graph-theoretic modeling approaches, with nodes and edges representing how elements depend on one and other. Our approach is broadly related to the same type of graph. If probability data is added to the graph a Bayesian analyzer can interpret these approaches as well as the method proposed here. The key difference between our modeling approach and the others is our focus on the success of achieving goals (*top-down*) and not likely causes of failure (*bottom-up*), which we now elaborate.

Let us first say what we mean by *bottom-up* and *top-down* approaches to defining risk, since these terms are used to mean different things by different authors. The bottom-up, fault-oriented approach of working from *cause to impact* introduces the following difficulty. The failure of some simple element (such as a bolt working loose) could ultimately lead, via a set of interdependencies, to the failure of (say) a major power station. Methods such as Fault Trees and Event Trees could be used to model this risk, and they would track the failure of successive subsystems, working from the loose bolt upwards towards the failure of the power station. This is easy to understand and logical.

However, the loose bolt is just one of literally thousands of events that could cause the power station to fail. It could for instance fail due to industrial action in its fuel supply chain, or through an EU directive restricting use of fossil fuel, or because an aircraft hit a distribution network. The question then arises, *why would we focus on the bolt working loose?*. In other words this approach is restricted to investigating only those avenues that occur to us when we carry out the analysis, and may fail to include the factors, which actually turn out to be critical, and thereby fail to span the relevant domain of causality.

By contrast, top-down goal-oriented approach corresponds to recursively expanding branches from trunk outwards, which *generates an entire tree*. As a loose analogy, bottom-up is like checking a bicycle inner-tube for a slow leak by examining some suspect parts with a magnifier, while top-down is plunging it into water and checking for bubbles.

V. LINKING ISOLATED RISK MODELS AND SHARING RISK DATA

Dependency models, as explained in Section II, support the identification of external entities on which an enterprise depends. They also support the establishment of *critical dependencies* – the entities the enterprise is most sensitive to, and are most pivotal to their success. When these critical dependencies have been identified they can in turn be investigated by examining their own dependencies through the process we called *Zooming*. This could be a continual cycle where multiple *zooms* create a model of interdependent systems across an infrastructure. When considering complex 'supply chains', this is a very powerful tool. In reality however, an enterprise owner is unlikely to be willing to share all of its dependency data with other parties – not to mention the security risk of centralizing such information. Thus, the current scope of the system focuses on supporting *zooming* by sharing a limited view of an enterprise's dependency data, determined by the requirement to access the data. If an entity has a dependency on an external entity, its owner can periodically request access to state data from the model of that external entity, obtain the current state of the goal on which they depend, and instantly incorporate this understanding in their own model.

The data embodying models in our proposed system is stored, managed and distributed in a way that both gives some protection against attack by hostile parties attempting to learn of system vulnerabilities from the model, and also manages the scaling problem of handling the data for large models whose complexity grows roughly exponentially with depth. The idea is that each entity's data is stored and managed by only the enterprise owning the entity, and access to it is through a network connection, the URL for which is securely managed and distributed based on the access restrictions as defined by the enterprise.

We recommend that the URL point to a disaster-resilient endpoint - for example a mirrored site or secure datacenter - away from the geographical location of the entity. The reason for this is because, should the entity succumb to a major incident, the data will not be able to be retrieved if it is destroyed in the same incident.

When an enterprise creates a model, it stores it in a *secured Internet-accessible location* with and associated Uniform Resource Identifier (URI), and allows limited access to other enterprises, to view parts of the model. A dependency model consists of a number of goals, each with a Globally Unique IDentifier (GUID) that is automatically created for every new goal. The state and probability data for each goal is thus individually discoverable. If an analysis requires data regarding an external entity, the analyst can contact the owner of that entity through an agreed protocol, and request access to data for the goals on which they depend. For example, if a hospital depends on an electronic patient admission service that is remotely hosted, it can contact the hosting enterprise and request access to the data of the goal that represents the correct functioning of a patient admissions service. If agreed, the external entity will grant access to that specific goal but not the rest of the model. This is enterprise-

based, fine-grained access control. Of course, the negotiation could include access to the dependencies of the goals, on which they depend, iteratively to whatever depth is both desired and permitted. The negotiation of access to goal states and the competition and political ramifications that are associated with it are outside the scope in this paper. At present we consider only the technical aspects, enabling the secure sharing of state data from dependency models.

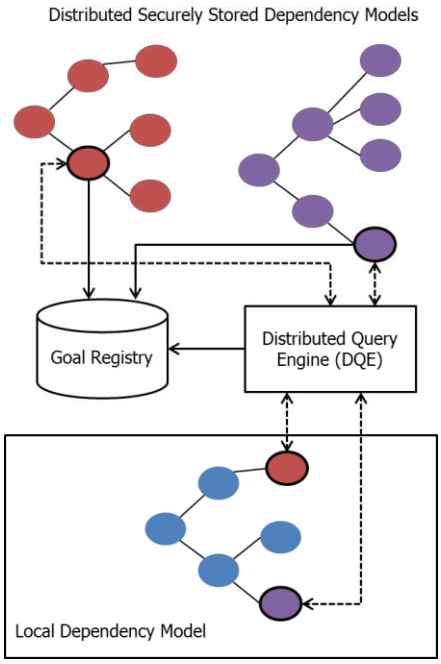

Figure 2 - Distributed Model "Zooming"

The URIs (including GUID) of goals that are the subject of sharing agreements are automatically pushed to a global *Goal Registry* (see Fig. 2), which essentially contains a list of unique goal identifiers and URIs pointing to the network addressable endpoint to which queries for the state of that goal can be sent. Once the agreement has been made to share model data, the *dependent entity* creates a goal in their local dependency model and attaches its state information to the URI of the remote dependency. To populate the state information, the local model sends a request for the status of that goal to a *Distributed Query Engine (DQE)* (see Fig. 2). A DQE exists within each enterprise that agrees to share their model data. Acting in client mode, it sends requests for goal status to a DQE acting in server mode at remote sites. Thus, data requesters also act as data providers – a relationship that is true of service providers in Complex Systems. The state and probability data is returned to the requester (via a secure cryptographic protocol) and added to the goal in the local model. If the state is different to the one already in the model, the model is updated to reflect the change in state. This gives an instant view of how failure within a remote system affects a local system, and enables interdependencies within a complex system to be continually updated. The Goal Registry can be pinged periodically to request the location of state and probability data for external goals. In an emergency situation this could be every few seconds.

VI. Case Study

In this section we present an example of two isolated dependency models: a hospital and a data center. The latter hosts and provides the electronic admissions service for the former. We demonstrate how external dependencies are discovered within an isolated model and then show how the two models are merged.

Figure 3 (left) shows a snippet of the dependency model for a data center. It focuses on "Systems OK" - the correct functioning of the IT systems hosted by the data centre. This data center provides the patient admissions service to a hospital at a different geographical location. The hospital has performed its own risk assessment and produced a dependency model which is partially shown in Figure 3 (right), focused on "IT Systems OK", which includes a dependency "Admission System OK" – the availability and correct functioning of the electronic patient admission system. This is an external dependency, provided by the data center. Therefore, "Systems OK" and "Admission System OK" are one and the same, i.e. a common element in both models. They just happen to have been given different names by their modelers, but discussion between the two enterprises will establish that this goal is the common element in both models. By using the distributed data sharing method described in Section V, we can add a reference to the actual goal state data, stored and managed in the model of the external dependency. Using our Distributed Query Engine (DQE), we can mine the current status data for "Systems OK" from the remotely stored model, update the hospital model, and identify any changes.

Using a real-world example to describe how unexpected events can impact on Complex Systems, we consider the Buncefield incident[1] where a series of explosions at a major oil storage facility caused a nearby business park to burn down. As it happens a data center on that site provided the electronic patient admissions system for a hospital 50 miles away. At the time, there was no way to gauge the impact this would have on the hospital's admission system, nor the dependencies of that system.

With our approach, an analysis of the data center could have quickly judged which of the goals within the model had failed. Cause of failure is not important at this point. However, by informing the dependencies of the failure, the risk model can be updated and appropriate action can be quickly taken. One of the failed goals would be "Systems OK". The hospital would periodically be using the DQE to query the state of its dependencies, which would obtain a response from the data center model that "Systems OK" had a success probability of 0. The DM software would immediately change the state of the goal representing this in the hospital's model and the impact would be discovered instantly.

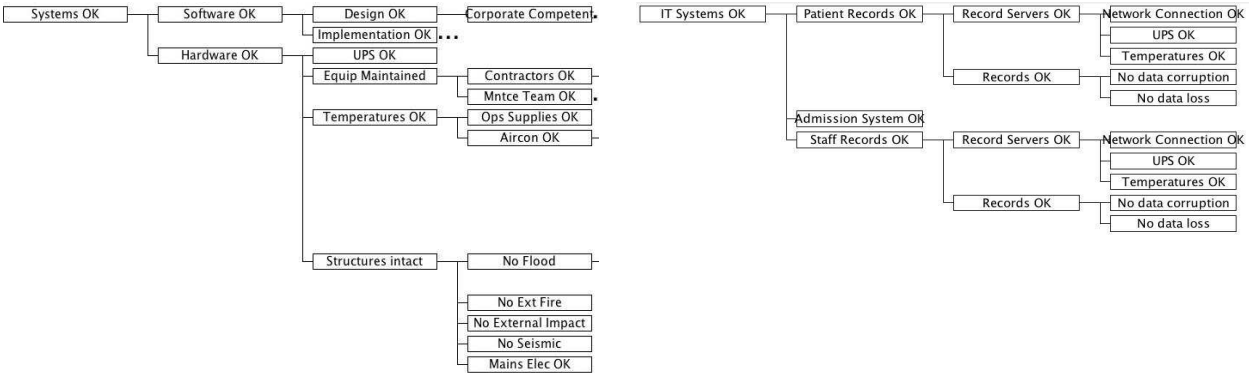

Figure 3 – Partial Data center Dependency Model (left), and Partial Hospital Dependency Model (right)

[1] The Buncefield Incident – Final Report. 2008. Available at http://www.buncefieldinvestigation.gov.uk/reports/

Figure 4 (top) shows the hospital model again, this time with success probability scale displayed. “IT Systems OK” has a success probability of 0.45 while the other goals are in their current state.

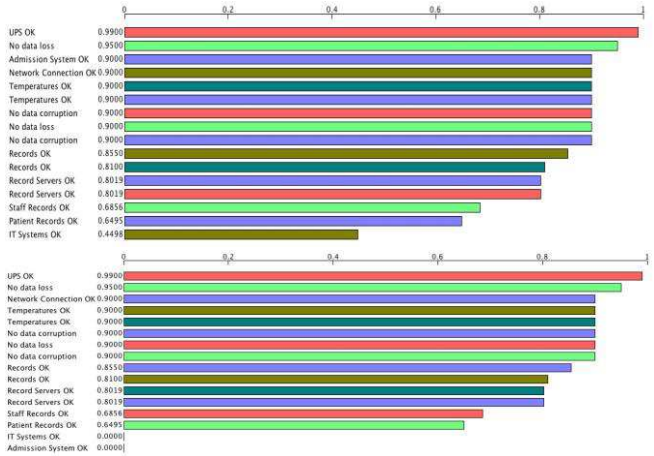

Figure 4 – Probability Table for the Hospital Model before (top) and after (bottom) ‘Admissions System Failure’

If we change the success probability of “Admission System OK” to 0, that is, we assume the DQE has updated the model following the incident, we can see from Figure 4 (bottom) that the “IT Systems OK” goal now has a success probability of 0, as the failure of a critical dependency has had an impact on its success. This is an instant update following the failure of an external entity.

VII. CONCLUSION

Prior to this work it was not usual to join risk models and share functional status data such that a change in the status of an external entity could be immediately incorporated into the models of its dependencies. Existing work into vulnerability/sensitivity analysis is capable of producing probability statistics and likely causes of failure within Complex Systems, but existing risk analyses are geared to handle only preconceived failures. Potential causes of failure change all the time and as cyber-terrorism becomes more of a threat, it is valuable to focus on prerequisites for success rather than what might fail, since this can generate the entire dependency tree rather than isolated routes from failure to consequence. Taking a goal-oriented view of risk enables such coverage, whereas a failure-oriented view can lead to overlooked threats, static outdated risk models and a myopic picture of vulnerabilities. The ability to evaluate the likelihood of a foreseen event fails to address the threat from unforeseen ones.

A top-down approach builds models that spans the entire causal domain and hence finds and evaluates all threats, not just the ones the model-writer thought of at the time of analysis. A more detailed technical description of the method can be found in The Open Group’s Dependency Modeling (O-DM) standard which was co-authored by the authors of this article[2].

[2] https://www2.opengroup.org/ogsys/catalog/C133

VIII. ACKNOWLEDGEMENTS

This work is funded by the Airbus Group Endeavr Wales (AG-EW-14:700033) scheme under the SCADA Cyber Security Lifecycle (SCADA-CSL) programme. We are grateful to Prof. David Slater and Prof. John Gordon for their input into to the paper and development of the case study concepts.

Dr Pete Burnap is an associate professor/senior lecturer in the School of Computer Science & Informatics at Cardiff University, UK. He holds a PhD degree in Computer Science from Cardiff University, UK. His research focus is cyber conflict, crime and security more specifically, the analysis and understanding of online human and software behaviour, with a particular interest in emerging and future risks posed to civil society, business (economies) and governments, using computational methods such as machine learning and statistical data modelling.

Dr Yulia Cherdantseva received her PhD degree from Cardiff University, UK in 2015. Dr Cherdantseva is currently a Lecturer at the School of Computer Science & Informatics at Cardiff University, UK. Her research has been concerned with the integration of security into business process models, security knowledge representation and risk assessment in SCADA systems.

Prof Andrew Blyth received his PhD in computer Science in 1995 from Newcastle University, UK. He is currently the Director of the Information Security Research Group (ISRG) at the University of SouthWales, UK. Over the past 20 years he has published many papers on the subject of Cyber Security and Computer Forensics. Professor Blyth also has function as an expert witness for various law enforcement agencies.

Peter Eden received his B.Sc.(Hons) degree in computer forensics from the University of South Wales (USW), Pontypridd, Wales in 2014. In 2016, he joined the Information Security Research Group, University of South Wales School of Computing Engineering and Science, as a Lecturer. His current research interests include embedded device forensics, SCADA forensics and incident response.

Dr Kevin Jones holds a BSc in Computer Science and MSc in Distributed Systems Integration from De Montfort University, Leicester where he also obtained his PhD in 2010. He is the Head of Airbus Group Innovations Cyber Operations team and is responsible for research and state of the art cyber security solutions in support of the Airbus Group (Airbus, Airbus Helicopters, Airbus Defence & Space, and Airbus HQ).

Hugh Soulsby studied Electrical and Electronic Engineering in The Polytechnic of Wales. From 2003 to 2014 he worked for Airbus Defence and Space (through many name changes) as a Validation and Verification Engineer on cryptography. He currently works as a Cyber Security Research Engineer for Airbus Group in Newport, UK.

Dr Kristan Stoddart gained his PhD from Swansea University, UK in 2006. Currently, he is a Reader at the Department of International Politics at Aberystwyth University, UK. He is also Deputy Director of the Centre for Intelligence and International Security Studies (CIISS).

REFERENCES

[1] ISO, 2009. Risk Management – Vocabulary, iSO Guide 73:2009.

[2] SANS Institute Whitepaper. A Qualitative Risk Analysis and Management Tool – CRAMM. Published 2002 . Available at http://www.sans.org/reading_room/whitepapers/auditing/qualitative-risk-analysis-management-tool-cramm_83

[3] Sans Institute Whitepaper. Using Vulnerability Assessment Tools To Develop an OCTAVE Risk Profile . 2004. Available at http://www.sans.org/reading_room/whitepapers/auditing/vulnerability-assessment-tools-develop-octave-risk-profile_1353

[4] IEC, 1990. Fault Tree Analysis (FTA), IEC 61025.

[5] ISO/IEC 31010:2009, *Risk management – Risk assessment techniques* . 2009. Available at http://www.iso.org/iso/home/standards/iso31000.htm

[6] Schneier, B., 1999. Attack trees: modeling security threats. Dr. Dobb's Journal of Software Tools 24 (12), 21–29.

[7] Mannan, S., Lees, F.P., 2005. Lees' Loss Prevention in the Process Industries, vol. 1, 3rd ed. Butterworth-Heinemann.

[8] Rausand, M., Høyland, A., 2004. System Reliability Theory, Models, Statistical Meth- ods and Applications, 2nd ed. Wiley.

[9] Charniak, E., 1991. Bayesian networks without tears: making Bayesian networks more accessible to the probabilistically unsophisticated. AI Magazine 12 (4), 50–63.

[10] Rinaldi, S.M., Peerenboom, J.P., Kelly, T.K., 2001. Identifying, understanding and analyzing critical infrastructure dependencies. IEEE Control Systems Magazine, 11–25.

[11] Braendeland G, Refsdal A, Stolen K. Modular analysis and modelling of risk scenarios with dependencies. Journal of Systems and Software, Volume 83, Issue 10, October 2010, Pages 1995-2013, ISSN 0164-1212, DOI: 10.1016/j.jss.2010.05.069.

[12] Brændeland G, Dahl H, Engan I and Stølen K. Using Dependent CORAS Diagrams to Analyse Mutual Dependency. Lecture Notes in Computer Science, 2008, Volume 5141/2008, 135-148, DOI: 10.1007/978-3-540-89173-4_12

[13] Agrawal, V. A Comparative Study on Information Security Risk Analysis Methods, Journal of Computers, 2017, Volume 12/1, 57-67, DOI: 10.17706/jcp.12.1.57-67

[14] Zio, E. Challenges in the vulnerability and risk analysis of critical infrastructures, Reliability Engineering & System Safety, 2016 Volume 152, 137–150.

[15] Knowles, W., Prince, D., Hutchison, D., Pagna Disso, J.F., Jones, K. A survey of cyber security management in industrial control systems, International Journal of Critical Infrastructure Protection, 2015 Volume 9, 52–80.

[16] Cherdantseva, Y., Burnap, P., Blyth, A., Eden, P., Jones, K., Soulsby, H., Stoddart, K. A review of cyber security risk assessment methods for SCADA systems, Computers & Security, 2016 Volume 56, 1-27