Tools and Techniques for Analysing the Impact of Information Security

John C. Mace

School of Computing

Newcastle University

This dissertation is submitted for the degree of

Doctor of Philosophy

September 2017

To Dad

Acknowledgements

A special mention must first be made to my wife Denise for supporting, and more so, putting up with my dark moods during the writing of this thesis, especially the final arduous weeks. I would like to thank my supervisor Professor Aad van Moorsel for his wise words of advice and to Dr Simon Parkin for his guidance on the work presented in part two of this thesis. I have a huge debt of gratitude to Dr Charles Morisset who helped me enormously, and without whose help this thesis would not have been possible. I would also like to thank my examiners, Professors Paul Watson and Jason Crampton, for their time, comments and recommendations. A final note must go to the members of my 'hellish' fitness family, especially Gail and Neil Lang, Dorothy and Dennis Valums, and Mr Mark Scott. By making me run the odd marathon they ensured I did not spend the entire time in front of a computer.

Abstract

The discipline of information security is employed by organisations to protect the confidentiality, integrity and availability of information, often communicated in the form of information security policies. A policy expresses rules, constraints and procedures to guard against adversarial threats and reduce risk by instigating desired and secure behaviour of those people interacting with information legitimately. To keep aligned with a dynamic threat landscape, evolving business requirements, regulation updates, and new technologies a policy must undergo periodic review and change. Chief Information Security Officers (CISOs) are the main decision makers on information security policies within an organisation. Making informed policy modifications involves analysing and therefore predicting the impact of those changes on the success rate of business processes often expressed as workflows. Security brings an added burden to completing a workflow. Adding a new security constraint may reduce success rate or even eliminate it if a workflow is always forced to terminate early. This can increase the chances of employees bypassing or violating a security policy. Removing an existing security constraint may increase success rate but may may also increase the risk to security. A lack of suitably aimed impact analysis tools and methodologies for CISOs means impact analysis is currently a somewhat manual and ambiguous procedure. Analysis can be overwhelming, time consuming, error prone, and yield unclear results, especially when workflows are complex, have a large workforce, and diverse security requirements. This thesis considers the provision of tools and more formal techniques specific to CISOs to help them analyse the impact modifying a security policy has on the success rate of a workflow. More precisely, these tools and techniques have been designed to efficiently compare the impact between two versions of a security policy applied to the same workflow, one before, the other after a policy modification.

This work focuses on two specific types of security impact analysis. The first is quantitative in nature, providing a measure of success rate for a security constrained workflow which must be executed by employees who may be absent at runtime. This work considers quantifying workflow resiliency which indicates a workflow's expected success rate assuming the availability of employees to be probabilistic. New aspects of quantitative resiliency are

introduced in the form of workflow metrics, and risk management techniques to manage workflows that must work with a resiliency below acceptable levels. Defining these risk management techniques has led to exploring the reduction of resiliency computation time and analysing resiliency in workflows with choice. The second area of focus is more qualitative, in terms of facilitating analysis of how people are likely to behave in response to security and how that behaviour can impact the success rate of a workflow at a task level. Large amounts of information from disparate sources exists on human behavioural factors in a security setting which can be aligned with security standards and structured within a single ontology to form a knowledge base. Consultations with two CISOs have been conducted, whose responses have driven the implementation of two new tools, one graphical, the other Web-oriented allowing CISOs and human factors experts to record and incorporate their knowledge directly within an ontology. The ontology can be used by CISOs to assess the potential impact of changes made to a security policy and help devise behavioural controls to manage that impact. The two consulted CISOs have also carried out an evaluation of the Web-oriented tool.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Businesses, non-profit ventures or government agencies typically offer services or products to consumers, made available by orchestrating and executing different business processes across their organisations. Successful business process operations require a mix of accessing, collecting, processing, transforming, analysing, and storing information, often using diverse IT systems and technologies. Achieving regulation compliance, establishing trustworthiness, and gaining business advantage mean organisations are compelled to protect the confidentiality, integrity and availability of the information they govern. Chief Information Security Officers (CISOs) must manage this challenge and take responsibility as the main decision makers for setting security rules, procedures and defences commonly expressed in the form of information security policies. Aligning information security policies and business process requirements can often be a cause of tension; too much security may hinder business process productivity whilst too much freedom to process information can be a risk to its security [134]. A major challenge for a CISO can be designing and maintaining workable security policies, which may be demanding when dynamic business environments necessitate the need to regularly modify a security policy in accordance with new security requirements, or the introduction of new information processing systems. Maintaining workable policies is made harder considering human behaviour may impact and itself be impacted by information security [10, 87]. Clearly, modifying a security policy so that it stops or reduces business process productivity to an unacceptable level would render a security policy unworkable. A CISO must therefore understand the business implications of designing and modifying security policies, which leads to the aim of this thesis:

> **Aim:** *To provide new tools and more formal methodologies to help Chief Information Security Officers (CISOs) analyse the impact of information security policies on the productivity of a business process.*

1.1 Research Context

1.1.1 Business Process Workflows

Many business sectors including finance, healthcare and eScience use the concept of workflow to formally describe and efficiently orchestrate their everyday business processes using automated workflow management systems [13, 69, 79]. The exact definition of workflow can often differ across business sectors, or even organisations within the same sector [69]. The Workflow Management Coalition (WfMC) formed from global analysts, developers, researchers and consultants have been working towards the standardisation of workflow and its management since 1993. Their working definition of a workflow is:

> "An automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules." [106]

The concept of workflow is used in this thesis as it provides a suitable platform to formally express and analyse a business process. Well defined workflow descriptions are necessary for automation purposes and outline a workflow's components, their structure, and how they interrelate. Workflows may be described with differing scope (e.g. within single, or across multiple organisational departments), distributed across many different frameworks (e.g. cloud platforms [119]), and managed in many different ways (e.g. manually or automatically). By abstracting away such details this thesis does not concern itself with where a workflow is performed, nor the technologies used to manage it. Instead it focuses on the workflow itself and how its execution may be impacted by a security policy. Workflows are typically repeatable structures representing day-to-day business processes. An *instance* of a workflow is one instantiation, or execution of a workflow [106]. Productivity in terms of workflow is expressed as success rate, that is the percentage of completed instances among N attempts [69]. This thesis considers dynamic human factors, including availability, which can affect the success rate of a workflow. This thesis also introduces new techniques to generate an expected success rate under the uncertainty unavailability introduces.

1.1.2 Workflow Tasks

One common component to all workflows is a group of clearly identified atomic activities, or *tasks* (the work) that must be co-ordinated and performed in some specific and logical *ordering* (the flow), e.g. [41, 69, 174, 176]. For example, consider the abstracted purchase

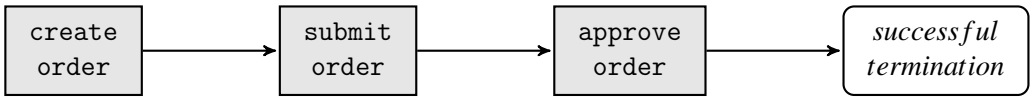


Fig. 1.1 Example of an abstracted purchase order approval workflow (POA), where the darker nodes represent tasks and the directed arrows represent the required order of task execution for the workflow to terminate successfully.

order approval workflow (POA) shown in Figure 1.1. The workflow consists of three tasks (`create order`), (`submit order`), and (`approve order`). The directed arrows indicate the order in which the tasks should be executed, for instance the execution of (`submit order`) can only follow the execution of (`create order`). The POA workflow therefore reaches *successful termination* once the last task, (`approve order`) has been performed. This simple example exhibits a linear, or sequential execution order meaning a task is enabled after the task immediately preceding it is completed. However, workflows may be more complex than this. Common scenarios include parallel splits where the execution path diverges into two or more parallel paths which execute concurrently, or conditional splits meaning the execution path diverges into two or more parallel paths of which only one will execute [176]. In the latter case, path selection is made, based on certain conditions or preferences, by some decision making agent represented in the workflow as a *choice point* [176].

1.1.3 Workflow Users

Another common aspect of workflow is a group of processing entities, or participants that perform the tasks in a workflow. These may be human, software systems, or a combination of both [69]. As previously mentioned, this thesis considers human behavioural factors in part, and therefore focus on workflows executed solely by human *users*. The work presented in this thesis, could however be adapted and applied to workflows which involve automated processing entities. User responsibilities when executing workflow tasks include interacting with computer systems in order to access and process information. A key enabler to achieving acceptable workflow success rates is the timely availability, accessibility and processibility of information to users. Information is also essential to drive workflows by informing critical decisions over which tasks need performing and when (e.g. at choice points).

1.1.4 Workflow Security Policies

Despite such a heavy demand for information, its availability and accessibility is commonly protected by implementing *security policies*. A security policy can be described as a set

of rules put in place to provide confidentiality, integrity, and availability of information by aiming to protect information and information processing systems from unauthorised access, use, disclosure, disruption, modification and destruction. Security policies may take many forms, for instance a password policy defining what form a password should take and how often it should be changed, or a network login policy which specifies how many login attempts a user can make before being locked out of the system.

A particular security concern with workflow is to ensure users with the correct clearance and capabilities are matched with appropriate tasks, and the threat of collusion and fraud is reduced [18]. For instance it would be desirable for the tasks (`submit order`) and (`approve order`) in the example `POA` workflow, shown in Figure 1.1, to be performed by different users, thus ensuring a user cannot order unauthorised goods. A workflow security policy is often expressed by sets of security constraints, many of which may have to be satisfied in conjunction when executing a workflow [6, 132, 186]. Typical workflow constraints are *authorisation constraints* that define which users can execute which tasks in any workflow instance, *separation of duty constraints* that define which tasks should not be executed by the same user in a single workflow instance, and *binding of duty constraints* that define which tasks should be executed by the same user in a single workflow instance [18, 20, 98, 188]. A security policy is therefore the third common aspect of the workflows we consider.

Having introduced the concepts of workflow tasks, workflow users, and workflow security policies, we are now in a position to give an informal definition of workflow to be:

A workflow consists of a set of ordered tasks, a set of users, and a security policy.

1.1.5 Chief Information Security Officers

Security experts are commonly employed by organisations in the form of Chief Information Security Officers (CISOs) to devise, manage and monitor information security policies. Due to the size and diverse nature of networked computerised systems and technologies used by large numbers of users within many organisations, setting a single information security policy may not be practical. According to *Information Security Policy - A Development Guide for Large and Small Companies*, published by the SANS institute [48], CISOs are recommended to take a more practical approach and express their organisation's information security policy in a number of different ways, by maintaining a collection of (sub) policies. Each of these can then express more detailed rules, constraints and procedures to guard against adversarial threats and reduce risk by instigating desired and secure behaviour of those people interacting with information legitimately during the execution of workflows.

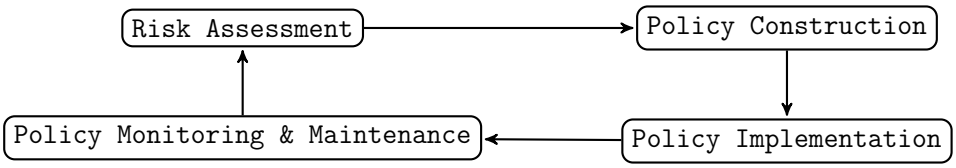


Fig. 1.2 Example of an abstracted security policy life cycle, where the policy monitoring and maintenance stage involves policy review and potential modification.

1.1.6 Security Policy Design

Complex and dynamic business environments mean periodic modifications must be made to a security policy over its operational lifetime to ensure it remains aligned with current organisational security and operational requirements. In other words a security policy must be both compliant and workable. Indeed, a security policy can be considered to be a 'living' document in the sense it is never finished, but is continuously updated as technology and employee requirements change. This progressive design process typically forms a policy life cycle consisting of a number of stages, taking a policy from its initial creation right through to its retirement. Particular examples include the *Security Policy Roadmap - Process for Creating Security Policies* (SANS institute), and *The Security Policy Life Cycle: Functions and Responsibilities* [171]. An abstracted security policy life cycle taken from [173] is shown in Figure 1.2. Of particular interest is the `Policy Monitoring & Maintenance` stage that, as the name suggests consists of: 1) monitoring user compliance with a policy and; 2) maintenance of a policy, which itself consists of two sub-phases, making policy reviews and making policy modifications. There can be many reasons, or drivers that prompt a CISO into reviewing, and possibly modifying a workflow security policy. Some possible change drivers are given in Table 1.1.

1.1.7 Security Policy Impact

Policy modifications may take many forms but intuitively consist of changing current policy content by adding new rules, removing obsolete rules, and/or updating rules or constraints, thus creating a new version of the policy. Modifications could also be applied in the context of creating a new policy required for a newly implemented business function, (i.e., changing from an empty policy to a non-empty one). Despite a CISO identifying compelling policy modifications, following a policy review, it does not mean she is necessarily free to implement them. The aforementioned policy life cycle guidelines highlight policy modifications must

Table 1.1 Example drivers for reviewing and modifying a workflow security policy

An organisation may wish to redesign and streamline a workflow by changing the process structure, the tasks involved, or the information required to complete the workflow.
Technologies and applications used by employees to access information and carry out workflow tasks may be updated or may change.
The structure of the workforce associated with performing a workflow may change in terms of employee numbers and skills.
A completely new policy may be needed following the introduction of a new business function, technology or software application used during the performance of a workflow.
An existing policy may need to be modified in order to apply it to a new business function, technology, or application.
An organisation may wish to demonstrate compliance with a particular regulation to meet legal obligations or instil consumer confidence as a trustworthy information guardian.
A new security threat may emerge when using a particular technology used to perform a workflow task, or an existing threat may no longer exist.
A policy may be found to be unworkable by slowing or stopping employees from completing workflow tasks on a regular basis.

first be recommended to, reviewed by, and agreed upon by business leaders. Intuitively, information security policies will have some level of impact on the success rate of workflows within their sphere of influence. Understanding this impact is a major concern before business leaders sign-off on any recommended modifications. A CISO must therefore present a trade-off, on the one hand justifying why a policy needs to be modified, often difficult when information security is both intangible and expensive (security metrics are being developed for this purpose, e.g. [108]). On the other hand a CISO is expected to communicate the potential impact of those modifications in a meaningful way. Clearly, any modification to a security policy that would reduce the success rate of a workflow has a negative impact for the business leader. This impact may be felt as an increase in the time it takes to complete an instance of the workflow, or it may prevent some or all workflow instances from completing. Of course, on the other hand, removing an unworkable or obsolete security constraint or procedure may have a positive impact thereby increasing the success rate of a workflow, although this may bring a potential reduction to security levels. Security policies should therefore minimise risk while not imposing undue access restrictions on those who need access to information.

Understanding the potential impact of policy modifications on workflow completion is important as modifying policies once implemented can be a long process [10]. Policies must be re-written, systems and controls reconfigured and tested, employees made aware and trained. Identifying policy modifications with an unacceptable impact before their implementation can help to avoid this. The potential impact of policy modifications is also useful to understand in scenarios where a negative impact from a modification on workflow completion is unavoidable, for instance, as the result of a regulation update. This impact may present an uncomfortable risk to the business leader, in this case, the potential for workflow failure. Two common approaches to managing risk that could be applied to workflow are, reducing the risk of failure and accepting the risk of failure [160]. One way to facilitate risk reduction is to use dynamic techniques that can adapt the way the workflow is performed according to risk indicators extracted from the environment, e.g. [148, 182]. In the case of risk acceptance, mitigation strategies can be formulated which consist of targeted actions to be taken when a risk of workflow failure materialises. For instance, a strategy may include emergency measures such as policy overrides (e.g. break glass [63]), allowing certain users to perform tasks they would not normally be authorised for. A mitigation strategy may include more long-term actions outlining staff security training or recruitment.

It is users who ultimately perform and complete tasks that form workflows (although they may not realise this), and it is users who impact and are themselves impacted by security on the ground. Information security and the human factor are therefore tightly coupled in respect to their impact on workflow success rates. From an organisational viewpoint, human behaviour, such as unavailability, meaning a user is unable to perform tasks when needed, can have a major impact on the success rate of workflows [87]. Unavailability of a user may for example, stem from being off sick or on vacation, being already occupied with performing another task, or simply their unwillingness to perform a task. Such situations make it necessary to find other, available users to perform these tasks, however the security policy may prohibit such users from doing so. Predicting and understanding the impact of this becomes a particular problem when many organisations in today's financial climate must make budget cuts and often operate workflows with minimal staffing levels. Users who are not technology minded and find information security hard to grasp can also cause delays when completing tasks, for instance, unfamiliarity with systems, misplacing data and forgetting passwords [10].

From a user's perspective, security places an added burden on their working procedures and can often become a source of tension when it causes seemingly unnecessary delays or even prevents tasks from being performed as and when required [134, 144]. For instance,

implementing a complex password policy may cause some users to forget their password which can take some time to reset. In other cases, misaligned security may prevent a workflow from completing if, for instance, users do not have the correct information access to complete certain tasks. Indeed it is widely reported that the pressures of securing information have led to many unworkable policies which, coupled with the pressures of productivity, have resulted in policy violations, user workarounds, bring your own device, and the unauthorised provisioning of cloud computing resources [119, 134, 140]. Adding more security therefore can arguably make information less secure in such cases. It is important then, for a CISO to consider the human aspect in order to provide more practical impact analysis of modifications made to an information security policy. This can be considered in terms of how human users behave and how they would likely react to policy modifications.

1.2 Security Policy Impact Analysis

As previously stated, an organisation's security policy is often expressed a a suite of sub-policies in order to target specific security concerns, many of which may be applicable to a workflow. Each policy, and any subsequent modifications made to it may, together with aspects of human behaviour, impact a workflow's successful completion in different ways. This makes it necessary for a CISO to perform many different types of impact analysis incorporating human factors. In order to do this the CISO must have appropriate workflow security impact analysis tools and techniques at her disposal.

Some work already exists in terms of analysing security policies, particularly detecting and resolving conflicts between multiple policies applied to the same business processes [112, 154, 161]. This issue often arises when policies are constructed independently by multiple authors within the same organisation [59]. Security policy compliance with workflow specifications is also considered, e.g. [97], which focuses on analysing separation of duty constraints. Of particular interest is analysing the presence and correctness of access control properties necessary for employees to access the information necessary to complete workflow tasks. Formal modelling approaches have been suggested for this purpose [9, 187, 189]. This is of particular importance in adaptive workflows whose configuration can change dynamically at runtime [175]. The inability to provide specific security properties when such workflows do change can prevent them from completing [49, 182]. Issues such as redundancy (unnecessary security constraints) and incompleteness (missing security constraints) can also be analysed using similar techniques [17]. Approaches have also been considered to analyse

whether obligations taken on to gain immediate information access can be fulfilled at a later time in accordance with a security policy [17, 111].

When it comes to predicting the impact of policy modifications on workflow completion, the process is less well articulated and less informed by any background theory, especially when considering the human element. Although some work on this area is starting to emerge (e.g. [31]), a lack of suitably aimed tools and techniques for the CISO means impact analysis is currently a somewhat manual and ambiguous procedure [134]. This can be overwhelming, time consuming, error prone, and yield unclear results, especially when workflows are complex, have a large workforce, and diverse security requirements. This thesis aims to help address this issue by providing new tools and more formal methodologies to help CISOs analyse the impact of security policies on workflow completion. More precisely, these tools and methodologies have been designed to simplify the process of impact comparison between two policies applied to the same workflow, that is one version with and the other without a potential policy modification. This work focuses on two specific types of policy impact analysis that incorporate the human factor.

1.2.1 Workflow Resiliency Metrics

The first analysis method for workflow security impact considers user availability, more precisely how the unavailability of certain users in conjunction with a security policy may impact the completion of a workflow. Availability is the *'the quality of being able to be used or obtained'*, *'the state of being otherwise unoccupied'*, or the *'freedom to do something'*[1]. In systems, availability is usually defined in terms of *up-time*. That is, states of a system are labelled either *up* or *down*, and the fraction of time in the states labelled *up* corresponds to the system's availability. In [127], Meyer and Sanders provide a general approach to the definition of metrics such as this. We consider a more specific notion of availability, namely the availability of users, which can be free or not during the execution of a workflow to execute tasks. Dynamic workplaces, complex work schedules and human behaviour mean users may not always be available during every execution of a workflow. For instance, a user may be required to perform tasks across several different workflows executing concurrently, meaning their availability for specific tasks in each workflow may be intermittent. In other cases, a user may be employed to only perform tasks in a single workflow suggesting they will always be available as required. However, sickness, vacation, tardiness and other countless reasons can still attribute to the user's unavailability. Availability and its impact on reducing

[1] Oxford English Dictionary

workflow success rate has been considered previously [141]. This approach looks at the unavailability of systems and administrators that control decisions on access requests to information assets. The impact of reducing success rates is measured as the time taken waiting for an access control decision. This is clearly a useful analysis technique for a CISO but it does not address the issue of how the security policy itself, coupled with user availability, impacts the success rate of a workflow, and therefore, is unsuitable in this regard.

A suitable first step to security policy impact analysis when considering user availability is to assess whether a workflow can complete without violating a policy while assuming all users will be available during execution. This problem is known as the workflow satisfiability problem, and is a well studied problem in the literature, e.g. [41, 163, 180, 190]. Solving the workflow satisfiability involves finding a complete and valid plan which assigns the execution of tasks to users. A plan is complete if the execution of all tasks are assigned, and valid if all security constraints are satisfied. Current work tends to concern itself with finding efficient algorithms to find whether a workflow is satisfiable, or not, e.g. [35]. However, by providing a quantitative solution to the workflow satisfiability problem we can start to think in terms of impact analysis metrics which indicate whether the security policy blocks the completion of a workflow.

An extension to the workflow satisfiability problem is workflow resiliency which considers whether a workflow is resilient to a number of users becoming unavailable, in the sense that it is still satisfiable [94, 179, 180]. Resilience, or Resiliency[2], is defined as the *'capacity to recover quickly from difficulties; toughness'*, or the *'ability of a substance or object to spring back into shape; elasticity'*[3]. In general, resiliency is the ability to withstand or recover quickly from adverse conditions [1]. In terms of systems, a system is resilient if it can persistently deliver trustworthy service despite changes, such as changes to the availability of needed resources. Intuitively, as the availability level of users reduces, the resiliency of a workflow is likely to reduce. Similarly, as a security policy is tightened, for instance by adding separation of duty constraints, it is likely the resiliency of a workflow will again reduce. In both cases, finding a complete and valid plan for which all users are available to execute, is likely to become harder. On the other hand, raised user availability levels or the weakening of a security policy may raise the resiliency of a workflow. The configuration of workflow tasks defined in a task schema may also cause resiliency to change. As the design of a security policy is managed by a CISO, whilst arguably, the configuration of workflow tasks and user availability is not, we focus on the impact of security policies on

[2]We use the term resiliency inline with related literature
[3]Oxford English Dictionary

the resiliency of a workflow. In doing so, we assume task configurations and predictions of user availability to be fixed.

Current solutions to quantifying workflow resiliency define the number of k users who can be unavailable at any point during the execution of a workflow. If a workflow is k resilient it is assured that the security policy allows the remaining, available users to complete the workflow while satisfying the security policy. In terms of impact analysis, modifying a security policy may raise or lower k but does not indicate a fine grained degree of impact a modification has on the resiliency of a workflow. For instance, two different policy modifications may cause a workflow that is 2 resilient to become 0 resilient, but is the impact of both modifications actually equal? The k resiliency metric offers a binary measure in the sense that a workflow is either k resilient or not. Imagine a workflow coming with a set of users, and there are 100 possible k sized subsets of users who could become unavailable during a workflow's execution. For the workflow to be k resilient, it must be satisfiable in all 100 cases of possible user unavailability. This may not practical in many situations when analysing real world workflows, that must work, but may not always complete depending on the availability of users. However, a workflow that is found to be satisfiable in 99 possible cases of k users being unavailable is clearly better than a workflow that is satisfiable in only 1 case.

By building on existing solutions this thesis defines a new resiliency metric called *quantitative resiliency* which indicates the expected success rate of a workflow under the assumption users may become unavailable. Another new resiliency metric is *distance resiliency* which indicates the expected point a workflow will terminate. This could be useful for a CISO by indicating at what point a security policy causes a workflow to become blocked. Providing these meaningful resiliency metrics means a CISO could predict the fine grained resiliency impact of each modification to a security policy. For instance, saying one modification reduces the quantitative resiliency of a workflow from 98% to 95%, whereas a different modification reduces it from 98% to 15% enables easy resiliency comparison. This is clearly more meaningful than saying a workflow is, or is not, k resilient, as the workflow will still provide a level of service following both policy modifications, although clearly a policy with the first modification will in general provide much better service than a policy with the second.

The resiliency of a workflow could in some cases be unavoidably low due to enforced policy modifications, meaning it may be necessary to reduce the risk of workflow failure. Computing workflow resiliency at runtime could be one way to reduce the risk of a workflow failing by ensuring the most resilient assignment of task executions to users is made in accordance with current predictions of user availability. Calculating resiliency when assigning

the execution of each task could itself have an impact on the success rate of a workflow making it necessary to minimise resiliency computation time. This thesis therefore examines how resiliency computation time is affected by modifications to a security policy and explores how computation time could be reduced. In terms of risk acceptance, the resiliency of a workflow could help understand the requirements for a suitable mitigation strategy. Workflows with a single execution path will have single quantitative and distance resiliency measure. For instance a high resiliency would imply an infrequent need to perform any mitigation actions. This could favour a mitigation strategy consisting of short-term, low cost actions such as security constraint emergency overrides. Low resiliency would suggest a broader strategy including more permanent yet costly mitigation actions such as staff training and increasing user availability, by cancelling vacations or paying overtime for example. Workflows with choice points on the other-hand can have multiple execution paths, any of which may be taken when executing a workflow. Workflows of this kind may therefore have a different level of resiliency for each path, making the formation of suitable mitigation strategies much more complex, especially when a workflow contains hundreds of execution paths. Taking the resiliency average alone could be a misleading indicator of success rate, especially when a workflow contains paths of both very high and very low resiliency. Therefore this thesis provides a single, understandable measure of resiliency for a workflow with choice and suggests how it can be used to form suitable mitigation strategies.

1.2.2 Ontology Development

The second analysis method for workflow security impact focuses on security policies applied at the task level of a workflow. The analysis of workflow resiliency described in Section 1.2.1 makes the simple assumption that if a user is available and they are permitted to perform a task, the task in question will be completed successfully. As discussed in Section 1.1.7, users face many challenges in respect to information security when performing workflow tasks even when they are permitted to do so. These challenges can delay and in extreme cases prevent a user completing their assigning tasks. Security impact analysis at the task level consists of identifying how users typically behave and react in relation to security modifications, assessing how policy modifications and related employee behaviours jointly impact the success rate of a workflow, and extracting behavioural controls to manage that impact.

Modelling and formalising human behaviour in security is a topic of increasing interest and is being researched in many areas of security including insider threat prediction [28],

passwords [100] and human decision making [191]. Ideally, a CISO would like to carry out quantitative analysis of security impact at the task level (e.g. [31]) however the related research is still in its infancy, still remains complex, and remains beyond the reach of most CISOs to measure in an accurate way. Instead, a more realistic qualitative analysis approach can be considered. As a first step, the CISO must understand how users are likely to react when imposing modifications to existing security policies. To facilitate this there are potentially large amounts of diverse (and in some cases untapped) knowledge relating to information security issues and human-behavioural factors, available from a wide array of sources and represented using a variety of terms and concepts [142].

Managing and formalising knowledge of various information security concerns and exposing their interdependencies within an ontology can form an information security knowledge base, useful for security impact analysis [61, 142]. The strength of an ontology is there is no single entry point to the information within it, allowing a CISO to analyse security impact from many different perspectives. For instance a CISO may start with a potential policy modification and want to assess its typical impact on user behaviour and therefore the success rate of a workflow, or a CISO may start with a typical user behaviour negatively impacting workflow success rate and assess what behavioural controls can be put in place to manage this. Another strength of an ontology is that knowledge fragments can easily be recorded and incorporated with existing content. Most CISOs and human factors experts however do not have the expertise to create an ontology directly using existing tools as such tools are complex and aimed at ontology experts [117, 118]. Recording knowledge is potentially error-prone and time-consuming, especially if a CISO or human factors expert is relying upon an ontology expert with limited availability, and who does not fully understand the nuances of the knowledge content and its structure, as they enter it into the ontology. To address this issue, this thesis introduces two new security ontology development tools, one graphical, the other Web-oriented dedicated for CISOs and human factors experts to record and incorporate their knowledge directly and intuitively within the structure of the underlying security ontology. Both tools can automatically translate this knowledge into a machine readable ontology file for automatic analysis. The graphical tool is dedicated to ontology construction whilst the Web-oriented tool can be used to create new and edit existing ontologies including those imported from the graphical tool. Furthermore, it can be used by a CISO to access and share ontology content, and use it to qualitatively analyse the potential impact on workflow success rate at the task level when modifying security policies, and help identify behavioural controls to manage that impact.

1.3 Research Problems

The work presented in this thesis focuses on improving two analysis methods CISOs can use to predict the impact security policy modifications have on workflow completion. Both analysis methods consider human behavioural factors. The first security impact analysis method considers workflow resiliency, that is the expected success rate of a workflow when assuming users may become unavailable during execution. Current resiliency metrics offer a binary measure indicating a workflow to be resilient, or not. This may not be practical in many cases, when analysing the potential impact of modifying the security policy of a workflow that must work but may not be resilient in every instance. This shortcoming has highlighted the following research problem:

Problem 1 *CISOs need fine grained metrics to analyse the potential impact security policy modifications have on the resiliency of a workflow.*

This first problem has raised a number of research questions which in turn have driven the research presented in part 1 of this thesis. In order to analyse the expected impact of a security policy modification, it is necessary for a CISO to compare the the resiliency of a workflow with and without the policy modification. To enable practical and meaningful resiliency comparison a CISO requires quantitative resiliency measures raising the question:

Question 1.1 *How can a CISO generate a quantitative measure of resiliency for a workflow?*

Central to calculating workflow resiliency is the expected availability of users when executing a workflow. It is therefore necessary to express, or model an expectation of user availability when calculating resiliency. More precisely it is necessary to represent which users are expected to be available and when during the execution of a workflow. It may be possible, or even necessary (dependent on user availability data) to represent a level of user availability when calculating the resiliency of a workflow. We therefore ask:

Question 1.2 *How can the expected availability of users be modelled in order to calculate the quantitative resiliency for a workflow?*

It is unrealistic to expect many security constrained workflows, executed by users who may become unavailable, to be fully resilient. Security policy modifications enforced by

regulations may lower the resiliency of a workflow below an acceptable threshold of failure risk. When workflow completion is of extreme importance it is necessary to specify ways to manage this risk, for instance through risk reduction, or risk acceptance. The latter case should outline allowable mitigation actions that can be taken to avoid a workflow terminating early. If a quantitative measure of resiliency can be calculated, we ask the question:

Question 1.3 *How can a quantitative measure of resiliency help manage the risk of workflow failure?*

The first research problem and its associated questions will be tackled in Part 1 of this thesis. The second security impact analysis method considers recording and incorporating security and human factors knowledge within the structure of a security ontology. A CISO could use an ontology to analyse how human behaviour is likely to affect and be affected by a security policy, and how these behaviours may affect the completion of a workflow. Currently, ontology development is complex and not easily accessible for knowledge holders to directly record and interrelate their knowledge. This has highlighted the following research problem:

Problem 2 *CISOs and human factors experts need to record and incorporate their security and human behavioural knowledge directly within the structure of an ontology.*

This second problem has raised a number of research questions which in turn have driven the research presented in part 2 of this thesis. As with any tool development it is necessary to capture the needs, or requirements of its targeted users, which raises the question:

Question 2.1 *What are the requirements for an ontology development tool created specifically for CISOs and human factors experts?*

A large number of ontology editing tools[4], and technologies, already exist that could be utilised to create specific ontology development tools. Understanding these tools and technologies is therefore important when meeting user requirements. It is also necessary to identify suitable and intuitive ways of knowledge recording and its incorporation with existing knowledge, e.g. graphical or text based. We therefore ask:

[4] e.g. see https://www.w3.org/wiki/Ontology_editors

Question 2.2 *What methods and technologies can CISOs and human factors experts use to record and incorporate their security related knowledge within an ontology?*

Once user requirements and the means to meet those requirements are understood, a natural next step is the implementation of a specific security ontology development tool. We therefore ask the question:

Question 2.3 *How can an ontology development tool tailored specifically for CISOs and human factors experts be implemented?*

The second research problem and its associated questions will be tackled in Part 2 of this thesis.

1.4 Contributions

The motivation of this thesis is to provide new tools and techniques to help CISOs analyse the potential impact modifying a security policy has on workflow completion. Two methods of workflow security impact analysis involving human behavioural factors are considered, and have led to several contributions in terms of workflow resiliency and ontology development. More precisely, the main contributions of this research are as follows:

- **Contribution 1.** We propose a novel approach to quantifying the satisfiability and resiliency of a security constrained workflow in order to provide workflow resiliency metrics (Chapter 2). We give a definition of workflow which exposes the notion of users becoming unavailable during the execution of a workflow. We then consider the execution process of a security constrained workflow, involving the execution of tasks being assigned to users, to be a decision making process, and show how it can be expressed as a Markov decision process (Chapter 3). We show that finding such an assignment is equivalent to finding the optimal policy of a Markov decision process whose value function provides the resiliency metrics we require. We also show how changing the reward function of a Markov decision process can provide different metrics of resiliency, which we call *quantitative resiliency* and *distance resiliency*.
- **Contribution 2.** We outline a systematic approach to encode a Markov decision process, modelling the execution process of a workflow, into the probabilisitic model checker PRISM. PRISM is subsequently used to automatically generate resiliency

metrics by verifying the existence of properties in the encoded Markov decision process (Chapter 4). We illustrate with examples how this single framework can be used to generate quantitative satisfiability, quantitative resiliency, and distance resiliency metrics for a workflow. We then show how the impact of a security policy modification on the resiliency of a workflow can be predicted. We also highlight interesting analysis cases, for instance adding security constraints does not necessarily reduce resiliency, while a workflow with no quantitative resiliency may still have resiliency in terms of distance.

- **Contribution 3.** We discuss how computing the quantitative resiliency of a workflow at runtime can help reduce the risk of workflow failure by ensuring the most resilient assignment is made in accordance with the current prediction of user availability (Chapter 5). We first give an empirical assessment of policy modifications on the quantitative resiliency computation time followed by a methodology to calculate a set of artificial security constraints that can be added or removed from the security policy. The effect of these constraints is observed in some cases to reduce resiliency computation time whist maintaining the quantitative resiliency of a workflow.

- **Contribution 4.** We give a definition for a workflow with choice and show how it can be reduced to a set of workflows without choice in order to calculate the resiliency of each execution path (Chapter 5). We introduce a new metric for security impact analysis called *resiliency variance* that indicates an overall resiliency variability or volatility from the resiliency average. Several other resiliency metrics are considered for a workflow with choice; *resiliency extrema*, *resiliency distribution*, and *expected resiliency*. Discussion is also given indicating how resiliency variance could be used for predicting a suitable workflow mitigation strategy.

- **Contribution 5.** We outline requirements and implementation details of a new graphical ontology development tool (Chapter 6). The tool prototype allows CISOs and human factors experts to record and incorporate their security knowledge within the underlying ontology structure. In particular the tool enables knowledge relating to potential security constraints and controls to be related to common human behaviours, useful for security policy impact analysis. The tool, created in Visual Basic provides a template of shapes and connectors allowing ontology diagrams to be easily constructed according to the ontology structure (Chapter 7). The graphical content can be both stored and automatically translated via a Java implemented program into the machine readable ontology language OWL without need for ontology expertise.

- **Contribution 6.** We outline the requirements and implementation details of a new Web-oriented collaborative ontology development tool (Chapter 6). The tool prototype extends the Web-based, open source ontology development tool Web-Protégé, making ontology development approachable both to CISOs and human factors experts by hiding the complexities of the underlying ontology. The tool supports several collaborative features allowing knowledge capture and alignment from across a number of participating organisations (Chapter 7). Tool requirements have been extracted from semi-structured consultations with two CISOs within large organisations (Chapter 6). Evaluation of the tool has also been carried out by the two CISOs, providing useful feedback on tool design and functionality.

1.5 Publications

This work has been documented in part in the following peer reviewed publications:

1. Mace, J. C., Morisset, C., and Van Moorsel, A. P. A. (2015c). Resiliency variance in workflows with choice. In *Proceedings of the 7th International Workshop on Software Engineering for Resilient Systems*, SERENE'15, pages 128–143

2. Mace, J. C., Morisset, C., and Van Moorsel, A. P. A. (2015b). Modelling user availability in workflow resiliency analysis. In *Proceedings of the 3rd Symposium and Bootcamp on the Science of Security*, HotSoS'15. Article 7

3. Mace, J. C., Morisset, C., and Van Moorsel, A. P. A. (2015a). Impact of policy design on workflow resiliency computation time. In *Proceedings of the 12th International Conference on the Quantitative Evaluation of Systems*, QEST'15, pages 244–259

4. Mace, J. C., Morisset, C., and Van Moorsel, A. P. A. (2014). Quantitative workflow resiliency. In *Proceedings of the 19th European Symposium on Research in Computer Security*, ESORICS'14, pages 344–361

5. Mace, J. C., Van Moorsel, A. P. A., and Watson, P. (2011). The case for dynamic security solutions in public cloud workflow deployments. In *Proceedings of the 41st IEEE/IFIP International Conference on Dependable Systems and Networks Workshops*, DSN-W'11, pages 111–116

6. Mace, J. C., Parkin, S. E., and Van Moorsel, A. P. A. (2010a). A collaborative ontology development tool for information security managers. In *Proceedings of the*

4th ACM Symposium on Computer Human Interaction for Management of Information Technology, CHIMIT'10. Article 5

7. Mace, J. C., Parkin, S. E., and Van Moorsel, A. P. A. (2010b). Ontology editing tool for information security and human factors experts. In *Proceedings of the 2nd International Conference on Knowledge Management and Information Sharing*, KMIS'10, pages 207–212

1.6 Thesis Structure

This chapter has motivated the research presented in this thesis and two research problems it looks to address in regards to analysing the impact security policies have on workflow completion. The research contributions and peer reviewed publications supporting the validity of the research have also been presented. The remaining chapters of this thesis are as follows:

- **Chapter 2.** A formal definition of security constrained workflow is given and shown to be satisfiable if a complete and valid plan can be found, that is a plan assigning the execution of tasks to users which satisfies all security constraints. The workflow definition is extended in a way that exposes the notion of users becoming unavailable at runtime. A workflow is shown to have a level of resiliency if a feasible complete and valid plan can be found. A plan is feasible in the sense that users are available to execute it. Quantitative measures are then defined in the form of workflow metrics. Related work on the workflow satisfiability problem and workflow resiliency is also presented.

- **Chapter 3.** An abstracted workflow execution process, involving the execution of tasks being assigned to users, is presented as a decision making process. It is shown how a workflow execution process is partly under the control of a process agent choosing tasks and user assignments, and partly random brought about by the uncertainty introduced by probabilistic user availability. The workflow execution process is modelled as a Markov decision process whose optimal value function provides the required workflow metrics. This single framework is shown to provide several resiliency metrics by changing the reward function which associates rewards to process transitions.

- **Chapter 4.** A systematic approach is given to encode a Markov decision process, modelling the execution process of a workflow, into the probabilisitic model checker

PRISM. The PRISM tool is introduced, including an overview of its high level modelling language, and how properties of probabilistic models encoded in PRISM are verified. PRISM is subsequently used to automatically generate resiliency metrics by verifying the existence of properties in the encoded Markov decision process. By example, it is shown how generating the resiliency of a workflow can be used to analyse the expected impact of a policy modification.

- **Chapter 5.** Two techniques are presented showing how calculating quantitative resiliency can help manage the risk of workflow failure. The first technique, aligned with reducing the risk of workflow failure, considers the computation of resiliency at runtime and ensuring it is maximised when taking the current prediction of user availability into account. The second technique, aligned with accepting the risk of workflow failure, considers the formation of mitigation strategies. In particular it is shown how calculating resiliency metrics can help form mitigation strategies for workflows containing choice.

- **Chapter 6.** The management of organisational knowledge is discussed highlighting how it can be recorded and incorporated into an ontology to form a security knowledge base. Information security knowledge holders are introduced, whose knowledge of information security and human-behavioural factors requires incorporating into an ontology, enabling qualitative impact analysis of security policy modifications on workflow completion. An overview of current ontology technologies and tools is provided before reviewing a number of ontological approaches to information security. The result of consultations carried out with two CISOs are also presented regarding security policy review and modification management.

- **Chapter 7.** The requirements and high-level implementation details are provided for two prototype security ontology development tools that enable CISOs and human factors researchers to collaborate. The first tool is graphical in nature, the second is Web-oriented. Both tools incorporate a user's entered content within the structure of an underlying information security and human factors ontology, and automatically translate it to machine readable ontology files written in the Web Ontology Language (OWL). Evaluation of the Web-oriented tool is provided by the same two CISOs consulted previously.

- **Chapter 8.** The thesis concludes with a summary of its contributions, reflections on the research problems posed in Section 1.3, and a number of interesting future directions the research could follow.

Part I

Metrics

Chapter 2

Workflow

When designing a security policy, a CISO may need to analyse how a policy design is likely to impact the completion of a workflow under different scenarios of user availability. A security policy has an impact if a workflow cannot be completed in a valid way, that is, without a policy violation or by necessitating some authorised remediation action, such as a policy override. This chapter provides the foundations for security policy impact analysis, and begins by formally defining a workflow before introducing the problem of whether a security policy has an impact on workflow completion whilst assuming all users are available to execute all tasks, a problem known as the workflow satisfiability problem. The workflow definition is then extended to expose the notion of probabilistic user availability during a workflow's execution, before describing the problem of finding whether a workflow can still be completed in a valid way under the condition that users may be unavailable, a concept known as workflow resiliency. New metrics for workflow resiliency are then introduced called *quantitative resiliency* and *distance resiliency*. In Chapter 3 we use our definition of workflow and encode the decision process of assigning the execution of tasks to users under the uncertainty of user availability as a Markov decision process. Using this single framework we show how our metrics can be generated to indicate the satisfiability and resiliency of a workflow, which are the concepts introduced in this chapter. In Chapter 4 we show how these metrics can be computer generated by encoding the Markov decision process in the probabilistic model checker PRISM, and how the impact of a security policy on the completion of a workflow can be analysed.

2.1 Workflow Specification

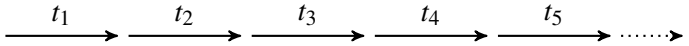
In general, a workflow is the automation of a business process. Typically, a business process is formally represented as a *workflow specification* which is input to an automated workflow management system for execution [69]. A workflow specification is often composed from the different components introduced in this section; namely a task schema, workflow users, and security policy. Note these components may be constructed by independent 'design' actors, or by the same actor adopting different 'design' roles.

2.1.1 Task Schema

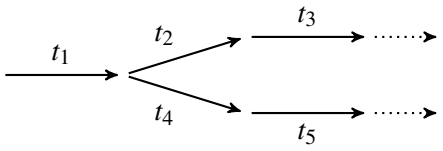
One common component of all workflow specifications is a workflow *task schema* which represents a business process in terms of what work needs to be done and in what order. More precisely, a task schema expresses a group of clearly identified atomic activities, or *tasks* (the work), that must be co-ordinated and executed in some specific and logical *ordering* (the flow) to accomplish the goal of the business process [41, 69, 174, 176]. We assume a task schema is constructed by a *schema designer* who defines both workflow tasks and any constraints on the order in which those tasks should be executed.

Ordering Constraints

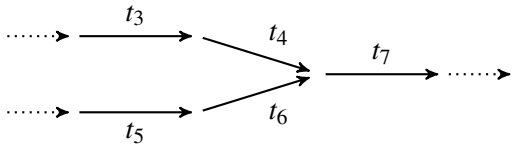
A task schema only defines tasks and the order in which they should be executed, and does not express how tasks are executed. We therefore consider a finite set of abstract workflow tasks as a set T. We make the assumption that all tasks are executed sequentially and the order in which tasks should be executed is defined by *ordering constraints*. A common ordering constraint indicates two tasks should be executed sequentially, that is the execution of one task should only follow the execution of the other [176].



In this example, all tasks should be executed sequentially, that is the execution of task t_2 is permitted after the execution of t_1, the execution of task t_3 is permitted after the execution of t_2, and so on. It can be imagined the output from task t_i constitutes the input of the next task in the sequence, t_{i+1}. The tasks are constrained in such a way that only one execution order is permitted, however task schemas may be more complex than this. For instance, ordering constraints may also permit parallel splits where two or more sequences of tasks are executable at the same time.



In this example, the execution of both tasks t_2 and t_4 is permitted only after the execution of t_1. The output from t_1 constitutes the input of t_2 and t_4 in this case. The tasks in the same branch (e.g. $t_2, t_3, \ldots, t_m$) should be executed sequentially, however their execution order in relation to tasks in another, parallel, branch (e.g. $t_4, t_5, \ldots, t_n$) does not matter. In other words, the execution order of tasks from different parallel branches can be interleaved. For instance, it is permitted for the execution of t_4 to take place before t_2, between t_2 and t_3, or after t_3, and so on. Similarly, ordering constraints may also specify parallel joins where two or more parallel sequences of tasks join into a single sequence.



In this example, the execution of t_5 is permitted only after the execution of both t_2 and t_4 meaning both parallel branches must be executed before t_5. The combined output from both t_2 and t_4 constitutes the input of t_5 in this case. Ordering constraints may also permit exclusive choice where only one sequence of tasks needs to be executed from a set of task, and we discuss this more in Chapter 5. Although not considered here, the repetition of single tasks (loops) or groups of tasks (sub-workflows) may also be permitted in practice [176]. Including such ordering constraints would add an extra layer of complexity which is unnecessary in the understanding of our approach. We therefore leave task schemas containing loops and sub-workflows for future work.

Ordered Sets

Before we give a formal definition of a task schema it is useful to first consider the concept of ordered sets. A strict partially ordered set (or poset) $(Q, <)$ is a pair consisting of a distinct

set of elements Q and a binary relation[1] $<$ on Q satisfying the following three properties:

$$\forall q \in Q, q \not< q \tag{2.1}$$

$$\forall q, q' \in Q, q < q' \Rightarrow q' \not< q \tag{2.2}$$

$$\forall q, q', q'' \in Q, q < q' \wedge q' < q'' \Rightarrow q < q'' \tag{2.3}$$

That is, $<$ satisfies the properties of irreflexivity, antisymmetry, and transitivity. The set Q is referred to as the *ground set* and $<$ as a *strict partial order* on Q. For any pair $q, q' \in Q$ the notation $q < q'$, $q' > q$, and $(q, q') \in <$ is used interchangeably and indicates q and q' are comparable. The order placed on Q by $<$ is *strict* in the sense $q < q'$ means q is strictly before q'. Furthermore, the order is *partial* as not all pairs of elements in Q may be comparable; there may exist $q, q' \in Q$ such that $q \not< q'$ and $q' \not< q$. The elements q and q' are said to be incomparable in this case and denoted by $q||q'$ as neither one precedes the other. The order-extension principle states that every partial order can be extended to a total order [124]. A total order is placed on Q when all pairs $q, q' \in Q$ are comparable, that is when $<$ satisfies the property of totality:

$$\forall q, q' \in Q, q < q' \vee q' < q \tag{2.4}$$

Given a poset $(Q, <)$, a linear extension of $<$ is a total order $\prec$ on the ground set Q that is consistent with the original partial order $<$, meaning $q \prec q'$ whenever $q < q'$. Algorithms for generating a poset's set of linear extensions has been well studied, e.g. [146, 150]. As a final note, the unique ordering of elements for any totally ordered set $(Q, \prec)$, where $q_1 \prec q_2 \prec \cdots \prec q_{|Q|}$, can be written as a sequence $(q_1, q_2, \ldots, q_{|Q|})$.

Task Schema Definition

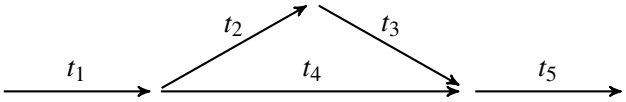
Given a set of abstract tasks T, ordering constraints, and the concept of ordered sets we define a task schema to be:

Definition 1 *A* task schema *is a strict partially ordered set $(T, <)$, where T is a finite set of distinct tasks and $<$ is a strict partial order on T.*

Each execution of a workflow is called a *workflow instance*. Given a task schema $(T, <)$, each *valid* workflow instance involves all tasks in T being executed in a sequential order that

[1] A binary relation on a set A is a collection of ordered pairs of elements in A. In other words, $<$ is a subset of $A \times A$.

respects the partial order $<$. In other words, the task execution order of a valid workflow instance is a linear extension of $<$. In [41], Crampton and Gutin introduced the notion of an *execution schedule* for a task schema $(T,<)$; namely, a linear extension of the set of workflow tasks. We write σ to denote an execution schedule for $(T,<)$, and Σ for the set of all execution schedules for $(T,<)$. We now introduce an example task schema $(T_1,<_1)$ where $T_1 = \{t_1,t_2,t_3,t_4,t_5\}$ and $<_1 = \{(t_1,t_2),(t_1,t_4),(t_2,t_3),(t_3,t_5),(t_4,t_5)\}$. The strict partial order $<_1$ can be illustrated as follows:



Note, examples of sequential ordering, parallel splits and parallel joins are evident. The existence of incomparable task pairs $(t_2,t_4) \notin <_1$ and $(t_3,t_4) \notin <_1$ mean three execution schedules exist, such that the set of all execution schedules for $(T_1,<_1)$ is $\Sigma_1 = \{\sigma_{11},\sigma_{12},\sigma_{13}\}$, and:

$$\sigma_{11} = (t_1,t_2,t_3,t_4,t_5)$$
$$\sigma_{12} = (t_1,t_2,t_4,t_3,t_5)$$
$$\sigma_{13} = (t_1,t_4,t_2,t_3,t_5)$$

Therefore, any workflow instance satisfies $(T_1,<_1)$ if its task execution order equals one of the three execution schedules in Σ_1.

2.1.2 Workflow Users

Another common component of all workflow specifications is a group of task processing entities in the form of human users. Users are important as they are often needed to take responsibility for executing, or for initiating the execution of workflow tasks. Users detail such as name, skills or role, are not necessary in the understanding of our approach, therefore we define a finite group of abstract workflow users to be:

Definition 2 *A finite group of* workflow users *is a set U.*

We will only consider workflows where all tasks are executed in some form by users and make the assumption that only one user is necessary to execute a single task. Furthermore, we assume a workforce, or group of users is designated by a *workforce designer* for the specific purpose of executing all instances of a workflow. One can imagine the workforce designer

analyses what users are needed to execute a workflow in terms of size, type, experience, knowledge, skills and quality. Having defined the task schema $(T_1, <_1)$ in Section 2.1.1, we now consider the the set of workflow users $U_1 = \{u_1, u_2, u_3, u_4\}$.

2.1.3 Security Policy

The third common component of all workflow specifications is a workflow *security policy* which expresses a collection of security constraints defining which groups of tasks each user is permitted to execute in any instance of a workflow. A user is given permission to execute a task through the act of assignment, effectively giving the user all necessary system permissions and data access to complete the task in question. Often data access should only be given to users with the correct clearances and skills, meaning the execution of certain tasks by users may need to be constrained. We assume a security policy is constructed by a *security designer*, considered in this case, to be equivalent to a Chief Information Security Officer (CISO).

Security Constraints

A security policy may be a conjunction of many different kinds of security constraints [18, 20, 98, 188]. Common security constraints found in workflow include:

- **Authorisation constraints** defining which users are permitted to execute which tasks.
- **Separation of duty constraints** defining pairs of tasks that should be executed by two different users in the same instance of a workflow.
- **Binding of duty constraints** defining pairs of tasks that should be executed by the same user in the same instance of a workflow.
- **Counting constraints** defining the maximum number of tasks a user should execute in the same instance of a workflow.
- **Seniority constraints** defining pairs of tasks that should be executed by two different users, with one user being senior to the other, in the same instance of a workflow.

We consider a security policy consisting of the three most common constraints appearing in the literature, that is authorisation, separation of duty and binding of duty constraints. Further constraint types would add an extra layer of complexity which is unnecessary in the understanding of our approach.

Authorisation Constraints

In practice, users in large organisations are often grouped according to their skills into roles which are subsequently bound to workflow tasks [29]. Roles have a many to many relationship meaning a user can have multiple roles and a role can have multiple users. Binding a user to a role grants them permissions to access systems and information in ways that are necessary to execute those tasks to which their role is bound. This access control mechanism defined around roles and permissions is commonly known as role based access control (RBAC) [152].

To simplify understanding we do not explicitly define an RBAC style policy binding users to roles and roles to tasks, but instead consider its derivation as a relation $A \subseteq T \times U$ defining a set of *authorisation constraints*, stating which users are authorised to execute which tasks in any instance of a workflow. We assume A is inclusive such that $(t_i, u_j) \in A$ indicates user u_j is authorised to execute task t_i, and the set of all users authorised to execute t_i is the set $V_i = \{u_j \in U \mid (t_i, u_j) \in A\}$. It is straightforward to derive A from an RBAC policy, for example imagine two roles r_1 and r_2 which are bound to tasks t_1 and t_2 respectively. If a user u_1 is bound to both r_1 and r_2 then $(t_1, u_1) \in A$ and $(t_2, u_1) \in A$. If a second user u_2 is bound to r_1 but not r_2 then $(t_1, u_2) \in A$ and $(t_2, u_2) \notin A$. We assume for an authorisation relation A to be usable it must contain at least one user authorised to execute each task. In other words, given a set of tasks T, set of users U, and an authorisation relation A, $\forall t \in T, \exists u \in U \Rightarrow (t, u) \in A$.

Separation and Binding Constraints

Next we consider two other constraints common to workflows. *Separation of duty* constraints are used to prevent errors and fraud by restricting which tasks or groups of tasks a user can execute in the same instance of a workflow. A common example is preventing a user whose role allows them to submit and authorise expenses claims from authorising their own expenses claim. The relation $S \subseteq T \times T$ denotes a set of separation of duty constraints where for any $(t, t') \in T$, the same user must not execute both t and t'. *Binding of duty* constraints are used for continuity and restricting information flow by binding the execution of groups of tasks to individual users. For example, from a security viewpoint it may be a requirement to limit the dissemination of information accessed when executing several tasks to the same user. The relation $B \subseteq T \times T$ therefore denotes a set of binding of duty constraints where for any $(t, t') \in B$, the same user must execute both t and t'.

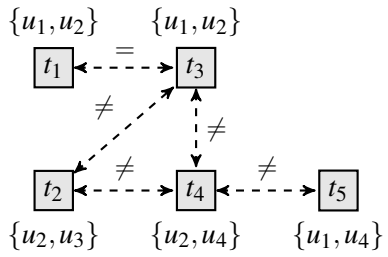


Fig. 2.1 Illustration of workflow security policy (A_1, S_1, B_1), where $\neq$ indicates a separation of duty between tasks, $=$ indicates a binding of duty between tasks, and $\{u_i, \ldots, u_n\}$ indicates the users authorised to execute a task.

Security Policy Definition

Having introduced authorisation constraints, separation of duty constraints, and binding of duty constraints we now give a definition of a security policy.

Definition 3 *A* security policy *is a tuple* (A, S, B)*, where* A *is a set of authorisation constraints,* S *is a set of separation of duty constraints, and* B *is a set of binding of duty constraints.*

In addition to the task schema $(T_1, <_1)$ and set of users U_1, defined in Sections 2.1.1 and 2.1.2 respectively, we now consider a workflow security policy (A_1, S_1, B_1), where:

$$A_1 = \{(t_1,u_1),(t_1,u_2),(t_2,u_2),(t_2,u_3),(t_3,u_1),(t_3,u_2),(t_4,u_2),(t_4,u_4),(t_5,u_1),(t_5,u_4)\}$$
$$S_1 = \{(t_2,t_4),(t_2,t_3),(t_3,t_4),(t_4,t_5)\}$$
$$B_1 = \{(t_1,t_3)\}$$

For clarity, an illustration of the security policy (A_1, S_1, B_1) is given in Figure 2.1. Each node represents a task $t_i \in T_1$ and labelled with the set V_i of all users authorised to execute t_i. Each edge between tasks t_i and t_j indicates a separation of duty constraint if labelled $\neq$ (i.e., $(t_i, t_j) \in S$), or a binding of duty constraint if labelled $=$ (i.e., $(t_i, t_j) \in B$). Note, the policy indicates user u_2 is authorised to execute tasks t_2 and t_4, but because $(t_2, t_4) \in S$, u_2 should not execute both tasks in the same workflow instance. Furthermore, u_2 is authorised to execute tasks t_1 and t_3, and because $(t_1, t_3) \in B$, if u_2 executes t_1 then u_2 should also execute t_3 in the same workflow instance, and vice versa.

2.1.4 Workflow Specification Definition

Having defined a task schema in Section 2.1.1, a set of users in Section 2.1.2, and a security policy in Section 2.1.3, we now give a definition for a workflow specification in a similar fashion to Wang and Li [180], and Crampton, Gutin and Yeo, [41]:

Definition 4 *A* workflow specification *is a tuple* $WS = ((T,<),U,(A,S,B))$*, where* $(T,<)$ *is a task schema,* U *is a set of workflow users, and* (A,S,B) *is a security policy.*

We are now in a position to consider $WS_1 = ((T_1,<_1),U_1,(A_1,S_1,B_1))$ to be a workflow specification.

2.2 Workflow Satisfiability

When designing a security policy, it is useful for the CISO, to establish whether a workflow specification defined in Section 2.1 is satisfiable, that is, whether the workflow specification can be implemented and executed without having to violate the specification's security policy.

2.2.1 Workflow Plans

Given a workflow specification $WS = ((T,<),U,(A,S,B))$, a workflow *plan* specifies an assignment of users in U to execute workflow tasks in T. A plan is represented as a function $\pi : R \to U$, where $R \subseteq T$, and given a task $t \in R$, $\pi(t) = u$ indicates the execution of t is assigned to user u. A plan π is said to be *complete* if the following property holds:

$$R = T \tag{2.5}$$

That is, π assigns assigns the execution of every task $t \in T$ to a user. A plan π which is not complete is deemed *partial*, that is when $R \subset T$. Furthermore, a plan π is *valid* if it satisfies the security policy (A,S,B); more precisely, if π satisfies the following three properties:

$$(t,\pi(t)) \in A \tag{2.6}$$

$$\forall (t,t') \in S,\ \pi(t) \neq \pi(t') \tag{2.7}$$

$$\forall (t,t') \in B,\ \pi(t) = \pi(t') \tag{2.8}$$

Table 2.1 All complete and valid plans for workflow specification WS_1, where a table entry $\pi_i \times t_i$ is the user u_i assigned the execution of task t_i by plan π_i.

	t_1	t_2	t_3	t_4	t_5
π_{11}	u_1	u_2	u_1	u_4	u_1
π_{12}	u_1	u_3	u_1	u_2	u_1
π_{13}	u_1	u_3	u_1	u_2	u_4
π_{14}	u_1	u_2	u_1	u_4	u_1
π_{15}	u_2	u_3	u_2	u_4	u_1

That is, all assignments of users to execute tasks in π are authorised by A, and all separation and binding constraints defined in S and B are respected. We write Π for the set of all possible plans for a workflow, and $\Pi_V \subseteq \Pi$ for the set of all complete and valid plans. In the case of $WS_1 = ((T_1, <_1), U_1, (A_1, S_1, B_1))$, the workflow specification defined in Section 2.1, five complete and valid plans exist such that $\Pi_{V1} = \{\pi_{11}, \ldots, \pi_{15}\}$. Each $\pi \in \Pi_{V1}$ is shown in Table 2.1, where $\pi_i \times t_j$ is the user returned by $\pi_i(t_j)$.

Note how more than one complete and valid plan may exist for a given workflow specification, whilst for others no complete and valid plan may exist. For example, imagine a policy with conflicting separation and binding of duty constraints $(t_1, t_2) \in S$ and $(t_1, t_2) \in B$. Note also that many other complete plans may exist which are not valid, given by $\Pi \setminus \Pi_V$. Furthermore, with the security constraints we consider, a plan π can be complete and valid independently of any task execution order. Task ordering is important however when considering the unavailability of users when executing a workflow, a concept which is introduced in Section 2.3.

2.2.2 Workflow Satisfiability Problem

Having defined a workflow specification $WS = ((T, <), U, (A, S, B))$ in Section 2.1.4 and workflow plan π in Section 2.2.1, we give a definition of workflow satisfiability to be:

Definition 5 *A workflow specification $WS = ((T, <), U, (A, S, B))$, is* satisfiable *if there exists a plan π that is complete and valid.*

The problem of finding whether a workflow specification is satisfiable is commonly known as the *workflow satisfiability problem*, defined to be:

Definition 6 *Given a workflow specification $WS = ((T, <), U, (A, S, B))$, the* workflow satisfiability problem *consists of finding a complete and valid plan π, or an answer that no such plan exists.*

In Section 2.2.1, we showed by example that multiple complete and valid plans may exist for any workflow specification WS, however, the workflow satisfiability problem requires only one complete and valid plan to be found for WS to be satisfiable. If WS is found to be satisfiable then every instance could be executed without violating any security constraints. Finding a complete and valid plan π may be relatively simple, for instance consider a security policy where $S = B = \emptyset$, where there are no separation or binding of duty constraints. In this case, it is enough for a plan π to assign each user u to execute a task t whilst ensuring $(t,u) \in A$. If there is no such user, WS is unsatisfiable. If $S \neq \emptyset$ and/or $B \neq \emptyset$ it may be necessary in the worst case to try all combinations of users to tasks in order to determine the existence of a plan π that is both complete and valid. Indeed, the workflow satisfiability problem has been shown to be NP-hard for any workflow specification containing even simple constraints such as separation and binding of duty constraints [41, 179]. If no complete and valid plan is found then every workflow instance would have to be terminated early to preserve security, force the security policy to be violated, or some authorised mitigation action taken, in order to complete execution [151]. Clearly, this situation can be avoided by considering the workflow satisfiability problem at design time and therefore make changes to the workflow specification as necessary, in particular changes to the security policy which is the focus of this thesis.

2.3 Workflow Resiliency

Although a workflow specification may be satisfiable, it still needs to be executed in a stepwise manner to respect task ordering in an environment that may be uncertain. We introduce the notion of uncertainty by considering the availability of users who may be available or not during the execution of a workflow. A workflow specification $WS = ((T,<),U,(A,S,B))$ will *deadlock* if no user $u \in U$ can be assigned the execution of a task $t \in T$ without violating the security policy. It may be that every user is prevented the assignment due to the workflow's security policy (A,S,B), or that every every user not prevented the assignment by the security policy (A,S,B) is unavailable. In the former case the workflow is not satisfiable, in the latter, the workflow is not resilient. Having defined whether a workflow is satisfiable in Section 2.2, we now consider whether a workflow is resilient. A workflow is resilient if there exists a complete and valid plan which can be implemented at runtime even when users are unavailable. The level of resilience for workflows is therefore called workflow resiliency.

2.3.1 Execution Specification

We assume users are benevolent and their availability is dependant on points in time relative to the progress of a workflow, and not the tasks themselves. We therefore consider the availability of users as a temporal notion in the sense that any changes to it will occur as the execution of a workflow progresses. To expose the notion of user availability we introduce an *execution specification* composed from the different components introduced in this section; namely an execution schema and availability forecast.

Execution Schema

The future availability of each user $u \in U$ can be expressed as a sequence of probabilistic availability predictions made for successive equally spaced steps of a workflow instance. These steps represents specific known events that will occur during every instance of a workflow and, depending on the required granularity for predicting user availability, may be defined in many ways. It is intuitive however to only consider those events for which knowing the possible availability of users is relevant. A step, called an *execution step*, therefore represents the event of a task being assigned to, and being executed by, a user. A user availability prediction is relevant to this event as it indicates which users if any, will likely be available to execute the selected task. To model the fact that a workflow can finish we consider a *termination step* which is the last possible step of a workflow. We write X for the set of execution steps, $x_\perp$ for the termination step, and define an execution schema to be:

Definition 7 *An* execution schema *is a totally ordered set* $(Z, \prec)$*, where* $\prec$ *is a total order on* $Z = X \cup \{x_\perp\}$ *which is a set of distinct workflow steps such that* X *is a set of execution steps and* $x_\perp$ *is the termination step.*

Workflow steps are expressed as a total order due to the assumption in Section 2.1.1 that tasks are executed sequentially.

Availability Forecast

Knowing the forthcoming availability of users is important for a workforce designer in order to provide assurance that all tasks will be executed. That is, what availability a user is likely to have and when they are likely to have it. In many cases it may be possible to forecast a user's availability for future instances of a workflow, and we call such a forecast an *availability forecast*. One data source for an availability forecast is the past availability of a user in the workplace, especially during the time frame of previous instances of the workflow

in question. Predictions may be made based on the user's average availability over that time period and used as their availability forecast for all future instances of the workflow.

Something more fine grained may be required, for instance providing an availability forecast for each forthcoming workflow instance. This may require analysing both previous availability history and future work schedules over the time period of the workflow instance (assuming this is known). Future work schedules can be likened to a workplace calendar where users outline their future availability on a day-by-day basis in the form of appointments. For example, a user may be 'busy' or 'out of office', they may be 'tentative', or they may 'free'. From a qualitative perspective, the knowledge and opinions of a user's manager and human resources personnel may also be taken into account. Furthermore, availability forecasts may be recalculated over time as more and more instances of a workflow take place, and future work schedules are updated.

Definition 8 *Given an execution schema* $(Z, \prec)$ *and set of users U, an* availability forecast *is a function* $\theta : (Z \setminus \{x_\perp\}) \times U \rightarrow [0,1]$*, where* $Z \setminus \{x_\perp\}$ *is the set of all execution steps in Z.*

Given an execution step $x \in Z \setminus \{x_\perp\}$ and a user $u \in U$, an availability forecast θ returns the predicted availability of u at execution step x as a probability. If $\theta(x,u) = 1$ then u is certain to be available at step x, whilst if $\theta(x,u) = 0$ then u is certain to be unavailable at step x. Expressing the availability of a user at an execution step as a probability allows an availability forecast to be classified according to different user availability levels. Given an execution schema $(Z, \prec)$, where $X = Z \setminus \{x_\perp\}$ is the set of execution steps in Z, an availability forecast θ is classified as *full* iff:

$$\forall u \in U, \forall x \in X, \theta(x,u) = 1 \tag{2.9}$$

That is all users are certain to be available for every execution step, and therefore available to be assigned the task selected for execution at each step x. An availability forecast θ is classified as *binary* iff:

$$\forall u \in U, \forall x \in X, \theta(x,u) = 1 \vee \theta(x,u) = 0 \tag{2.10}$$

That is all users are certain to be either available or unavailable for every execution step, and therefore either available or unavailable to be assigned the task selected for executionat each step x. It follows that θ is classified as *probabilistic* in any other case.

Table 2.2 User availability forecast θ_1, where a table entry $x_i \times u_i$ is the probability of user u_i being available at execution step x_i.

	u_1	u_2	u_3	u_4
x_1	0.8	0.7	0.1	0.1
x_2	0.8	0.8	0.9	0.1
x_3	0.8	0.3	0.7	0.1
x_4	0.6	0.9	0.7	0.4
x_5	0.8	0.9	0.7	0.0

Execution Specification Definition

Having defined an execution schema $(Z, \prec)$, and availability forecast θ we give a definition for an execution specification to be:

Definition 9 *An* execution specification *is a pair* $ES = ((Z, \prec), \theta)$ *where* $(Z, \prec)$ *is an execution schema and* θ *is an availability forecast.*

We now consider $ES_1 = ((Z_1, \prec_1), \theta_1)$ to be an execution specification, where $Z_1 = \{x_1, x_2, x_3, x_4, x_5, x_\perp\}$, and $\prec_1$ defines the total order of steps $(x_1, x_2, x_3, x_4, x_5, x_\perp)$. For the availability forecast $\theta_1 : (Z_1 \setminus \{x_\perp\}) \times U' \to [0,1]$, the set of users $U' = U_1$, defined in Section 2.1.2. Table 2.2 shows the availability forecast θ_1, where given $X_1 = Z_1 \setminus \{x_\perp\}$, a table entry $x_i \times u_j$ is the probability returned by $\theta_1(x_i, u_j)$.

2.3.2 Workflow Execution Specification

We introduce the concept of user availability to the execution of a workflow by considering a *workflow execution specification*, defined to be:

Definition 10 *A* workflow execution specification *is a pair* $WES = (WS, ES)$ *where* $WS = ((T, <), U, (A, S, B))$ *is a workflow specification and* $ES = ((Z, \prec), \theta)$ *is an execution specification.*

An workflow execution specification $WES = (WS, ES)$ is said to be valid if WS and ES are compatible. More precisely, given that $WS = ((T, <), U, (A, S, B))$, $ES = ((Z, \prec), \theta)$, $\theta : (Z \setminus \{x_\perp\}) \times U' \to [0,1]$, and $\mathbb{ES}$ is the set of all execution specifications, a workflow execution specification $WES = (WS, ES)$ is *valid*, for any $ES \in \mathbb{ES}$, if the following property holds:

$$|T| = |Z \setminus \{x_\perp\}| \wedge U = U' \tag{2.11}$$

Table 2.3 All complete and valid step-task mappings for workflow specification WS_1, where a table entry $\mu_i \times x_i$ is the task t_i mapped to the execution step x_i by μ_i.

	x_1	x_2	x_3	x_4	x_5
μ_{11}	t_1	t_2	t_3	t_4	t_5
μ_{12}	t_1	t_2	t_4	t_3	t_5
μ_{13}	t_1	t_4	t_2	t_3	t_5

That is the number of tasks must equal the number of execution steps as one task is executed per step, and θ must be an availability function for the users defined in the workflow specification WS. We define the satisfiability of a workflow execution specification to be:

Definition 11 *A workflow execution specification $WES = (WS, ES)$, is* satisfiable *if the workflow specification WS is satisfiable.*

Therefore, if a complete and valid plan π exists which satisfies a workflow specification WS then a workflow execution specification $WES = (WS, ES)$ is also satisfiable. Next we consider a *step-task mapping*, which maps execution steps in $X = Z \setminus \{x_\perp\}$ to tasks in T, to be a function $\mu : X \to T$. A step-task mapping μ is said to be *complete* if the following property holds:

$$\forall t, t' \in T, t \neq t', \exists x, x' \in X, x \neq x', \mu(x) = t \wedge \mu(x') = t' \tag{2.12}$$

That is, μ maps every execution step to a different task. Furthermore, a step-task mapping is valid if it respects the strict partial ordering over tasks defined by $<$. More precisely, a step-task mapping is *valid* if the following property holds:

$$\forall t, t' \in T, \forall x, x' \in X, t \leq t' \wedge \mu(x) = t \wedge \mu(x') = t' \Rightarrow x \leq x' \tag{2.13}$$

That is, if steps x and x' map to tasks t and t' respectively, and $t < t'$ then x should be ordered before x'. It follows that given a valid workflow execution specification $WES = (WS, ES)$, where $WS = ((T, <), U, (A, S, B))$ and $ES = ((Z, \prec), \theta)$, all complete and valid step-task mappings can be derived for WES directly from $(Z, \prec)$ and $(T, <)$. We denote the set of all complete and valid step-task mappings as M, where $|M| = |\Sigma|$, where Σ is the set of execution schedules for $(T, <)$. In Section 2.1.1, three execution schedules were shown to exist for the example task schema $(T_1, <_1)$, meaning $|M_1| = 3$. The three task-step mappings $\mu_{11}, \mu_{12}, \mu_{13} \in M_1$ are shown in Table 2.3, where a table entry $\mu_i \times x_j$ is the task returned by $\mu_i(x_j)$.

2.3.3 Feasible Plans

At runtime, a satisfiable workflow may not always be able to complete without violating the security policy. If users are unavailable during a workflow instance it may be necessary to continue the instance by assigning the execution of a task to a user who is available but not authorised to execute the task. We therefore consider whether a plan π is feasible. Informally, a plan π is feasible if each user u, assigned the execution of a task t by π, is not unavailable at the execution step task t is selected for execution. Plan feasibility can be represented as a function $\rho : \Pi \to [0,1]$, which given a plan π returns the maximum probability of π being implemented at runtime. Given $WES = (WS, ((Z, \prec), \theta))$ is a valid workflow execution specification, where $X = Z \setminus \{x_\perp\}$ is the set of all execution steps, and M is the set of all complete and valid step-task mappings for WES:

$$\rho(\pi) = arg\max_{\mu \in M} \left[\prod_{x \in X} \theta(x, \pi(\mu(x))) \right] \tag{2.14}$$

A plan π is therefore *feasible* iff:

$$0 < \rho(\pi) \leq 1 \tag{2.15}$$

If $\rho(\pi) = 0$ then π has no probability of being implemented at runtime, and is not feasible. Clearly, for plans π and π', if $\rho(\pi)$ is close to 1 and $\rho(\pi')$ is close to 0, then π has a much higher chance of being implemented at runtime than π'. Every plan π which exists for a workflow execution specification $WES = (WS, ES)$ will have a feasibility of $\rho(\pi) = 1$, iff θ is a full availability forecast (Section 2.3.1, Property 2.9). When considering the workflow satisfiability problem, if a complete and valid plan π can be found, such that WES is satisfiable, then π will also be feasible.

Given the workflow specification $WS_1 = ((T_1, <_1), U_1, (A_1, S_1, B_1))$ and execution specification $ES_1 = ((Z_1, \prec_1), \theta_1)$, we now consider the workflow execution specification $WES_1 = (WS_1, ES_1)$. Taking the complete and valid plan π_{15} (Section 2.2.1), the set of all valid step-task mapping $M_1 = \{\mu_{11}, \mu_{12}, \mu_{13}\}$ (Section 2.3.2), and the availability forecast θ_1 shown in Table 2.2, the probability of π_{15} being implemented under each $\mu_{1j} \in M_1$ is shown in Table 2.4. The feasibility of π_{15} is therefore given to be $\rho(\pi_{15}) = 0.060$, correct to 3 decimal places.

Table 2.4 Probabilities of plan π_{15} being implemented under each step-task mapping $\mu_i \in M_1$, and where the feasibility of π_{15} is given to be $\rho(\pi_{15}) = 0.060$, correct to 3 decimal places.

	μ_{11}	μ_{12}	μ_{13}
$\theta_1(x_1, \pi_{15}(\mu_i(x_1)))$	0.700	0.700	0.700
$\theta_1(x_2, \pi_{15}(\mu_i(x_2)))$	0.900	0.900	0.100
$\theta_1(x_3, \pi_{15}(\mu_i(x_3)))$	0.300	0.100	0.700
$\theta_1(x_4, \pi_{15}(\mu_i(x_4)))$	0.400	0.900	0.900
$\theta_1(x_5, \pi_{15}(\mu_i(x_5)))$	0.800	0.800	0.800
$\prod_{x \in X_1} \theta(x, \pi(\mu_i(x)))$	0.060	0.045	0.035

2.3.4 Quantitative Resiliency

Having defined a complete and valid plan π in Section 2.2.1, and a feasible plan π in Section 2.3.3, a workflow execution specification $WES = (WS, ES)$ has a level of resiliency if there exists a feasible complete and valid plan π. We represent the resiliency of a workflow as a function $\Gamma_Q : \mathbb{WES} \to [0,1]$, which given a workflow execution specification $WES = (WS, ES)$, returns a probability indicating the resiliency of WES. Note, we write $\mathbb{WES}$ to denote the set of all workflow execution specifications. Any complete and valid plan π which satisfies WES has a level of feasibility $0 \leq \rho(\pi) \leq 1$, where $\rho(\pi) = 0$ indicates π is not feasible. Although more than one complete and valid plan for WES may exist, denoted by the set Π_V, the feasibility of each $\pi \in \Pi_V$ may be different. Given WES, the goal of Γ_Q is to find a complete and valid plan $\pi \in \Pi_V$ whose feasibility maximises the function ρ. We now define *quantitative resiliency* of a workflow to be:

Definition 12 *The* quantitative resiliency *of a workflow execution specification* $WES = (WS, ES)$ *is:*

$$\Gamma_Q(WES) = arg \max_{\pi \in \Pi_V} [\rho(\pi)] \tag{2.16}$$

If $|\Pi_V| = 0$, then $\Gamma_Q(WES) = 0$, meaning WES can only have a quantitative resiliency greater than 0 if it is also satisfiable. Trivially, if WES is not satisfiable then the set of all complete and valid plans $\Pi_V = \emptyset$. If however, $\Gamma_Q(WES) = 0$ and $|\Pi_V| > 0$, then WES is satisfiable but not resilient which means no complete and valid plan exists which is also feasible. Therefore a workflow execution specification WES can be satisfiable and not resilient, but it cannot be unsatisfiable and resilient. That is WES must be satisfiable to be resilient.

Expressing quantitative resiliency as a probability means $\Gamma_Q(WES)$ indicates the expected success rate of WES in N instances, assuming forecasted user availability remains unchanged. That is, the probable percentage of instances that can be completed without having to violate the security policy. It is important to note that at runtime, a workflow may achieve a higher or lower success rate than indicated by Γ_Q. This is due to the fact that at runtime, users will either be available or not. Quantitative resiliency therefore indicates the maximum expected success rate which we suggest would be useful to a CISO when modifying a security policy before its implementation. The actual quantitative resiliency value may not be important, what is important is that it enables a workflow execution specification WES with different versions of a security policy to be ranked in order of expected success rate. Clearly, a security policy (A,S,B) has more impact on the success rate of WES than a policy (A',S',B'), if the quantitative resiliency of WES coming with (A,S,B) is less than the quantitative resiliency of WES coming with (A',S',B').

Quantitative Satisfiability

Having defined Γ_Q, we now give a quantitative answer to the workflow satisfiability problem defined in Section 2.2.2. As workflow satisfiability is independent of user availability, the satisfiability of a workflow execution specification $WES = (WS,((Z,\prec),\theta))$ can be established if θ is a full availability forecast (Section 2.3.1, Property 2.9). It follows that the feasibility of all plans $\pi \in \Pi_V$, where Π_V is the set of all complete and valid plans for WES, is $\rho(\pi) = 1$, therefore the quantitative resiliency of WES is $\Gamma_Q(WES) = 1$. We represent the satisfiability of a workflow as a function $\Gamma_S : \mathbb{WES} \to \{0,1\}$, which given a workflow execution specification returns 1 if WES is satisfiable, or 0 otherwise. We therefore define *quantitative satisfiability* to be:

Definition 13 *Given a workflow execution specification $WES = (WS,((Z,\prec),\theta))$ such that θ is full, the* quantitative satisfiability *of WES is:*

$$\Gamma_S(WES) = \begin{cases} 1 & \text{if } \Pi_V \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

2.3.5 Distance Resiliency

A workflow becomes *deadlocked* at an execution step x_i if no users are available at x_i, or no user can be assigned the execution of a task at x_i without violating the security policy. From a security perspective, it may be the case that a CISO wants to establish whether a security

policy causes a workflow to deadlock, and at which execution step x_i the deadlock occurs. To help a CISO analyse this problem we consider the notion of *distance resiliency*. Finding the distance resiliency of a workflow execution specification $WES = (WS, ((Z, \prec), \theta))$ consists of finding the expected maximum number of execution steps $x \in X$, where $X = Z \setminus \{x_\perp\}$, to be completed before WES becomes deadlocked.

To compute distance resiliency we turn to probability theory which states the expected value of a discrete random variable is the probability-weighted average of all possible values [74]. In other words, each possible value the random variable can assume is multiplied by its probability of occurring, and the resulting products are summed to produce the expected value[2]. More formally, let Y be a discrete random variable with a finite number of n outcomes, $y_1, y_2, \ldots..., y_n$, which occur with probabilities, $p_1, p_2, \ldots, p_n$, respectively. The expected value of Y, denoted as $E[Y]$, is therefore defined to be:

$$E[Y] = y_1 p_1 + y_2 p_2 + \cdots + y_n p_n = \sum_{i=1}^{n} y_i p_i \tag{2.18}$$

By convention, $y_i p_i = 0$ whenever $y_i = 0$ or $p_i = 0$, even when $p_i \neq 0$ or $y_i \neq 0$ respectively. Since all probabilities p_i add up to 1, that is $p_1 + p_2 + \ldots + p_n = 1$, the expected value $E[Y]$ is the weighted average, with p_i's being the weights. Using the concept of expected value to compute the distance resiliency of a workflow, let K be a random variable denoting the number of completed execution steps, given a plan π, a step-task mapping μ, a set of execution steps X, and an availability forecast θ. Consider the following table, which tabulates the number of completed execution steps k and the probability $p(K = k)$ that exactly that number of execution steps will be completed. For compactness, here p_i denotes the probability the user assigned by π is available to execute the task mapped to execution step x_i, that is $p_i = \theta(x_i, \pi(\mu(x_i)))$.

k	$p(K = k)$
0	$1 - p_1$
1	$p_1(1 - p_2)$
2	$p_1 p_2(1 - p_3)$
	$\ldots$

For example, the probability that no execution steps will be completed (i.e. $k = 0$), is the probability of the first execution step x_1 not being completed (i.e. $1 - p_1$). The probability that only the first execution step x_1 will be completed (i.e. $k = 1$), is the probability of the first

[2]I would like to thank Professor Jason Crampton for his guidance on using expected value.

Table 2.5 The expected number of completed execution steps for workflow execution specification $WES_1 = (WS_1, ((Z_1, \prec_1), \theta_1))$, using step-task mapping μ_{11} and plan π_{15}, is computed to be 1.65508, where k is the expected number of completed execution steps and $p(K = k)$ is the probability a discrete random variable $K = k$.

k	$p(K = k)$	$k * p(K = k)$
0	1.00000(1 - 0.7) = 0.30000	0.00000
1	0.70000(1 - 0.9) = 0.07000	0.07000
2	0.63000(1 - 0.3) = 0.44100	0.88200
3	0.18900(1 - 0.4) = 0.11340	0.34020
4	0.07560(1 - 0.8) = 0.01512	0.06048
5	0.06048(1 - 0.0) = 0.06048	0.30240
	$\sum_{k=0}^{5} = 1.00000$	$\sum_{k=0}^{5} = 1.65508$

execution step x_1 being completed (i.e. p_1), multiplied by the probability of the immediate succeeding execution step x_2 not being completed (i.e. $1 - p_2$). The expected number of completed execution steps is then defined by the following formula:

$$\sum_{i=1}^{|X|} i(1-p_{i+1}) \prod_{j=1}^{i} p_j = 1(1-p_2)p_1 + 2(1-p_3)p_1p_2 + 3(1-p_4)p_1p_2p_3 + \dots \tag{2.19}$$

$$= p_1 - p_1p_2 + 2p_1p_2 - 2p_1p_2p_3 + 3p_1p_2p_3 - \dots \tag{2.20}$$

$$= p_1 + p_1p_2 + p_1p_2p_3 + \dots \tag{2.21}$$

$$= \sum_{i=1}^{|X|} \prod_{j=1}^{i} p_j \tag{2.22}$$

We now consider a function $\lambda : M \times \Pi \times \mathbb{X} \times \Theta \to [0, |\mathbb{X}|]$, which given a step-task mapping μ, a plan π, a set of execution steps X, and an availability forecast θ, returns the maximum expected number of execution steps to be completed under plan π before a workflow execution specification $WES = (WS, ((Z, \prec), \theta))$ becomes deadlocked:

$$\lambda(\mu, \pi, (Z \setminus \{\perp_x\}), \theta) = \sum_{i=1}^{|X|} \prod_{j=1}^{i} \theta(x_j, \pi(\mu(x_j))) \tag{2.23}$$

For the workflow execution specification $WES_1 = (WS_1, ((Z_1, \prec_1), \theta_1))$, and using step-task mapping μ_{11} (Section 2.3.3) and plan π_{15} (Section 2.2.1), output of the function λ is shown in Table 2.5. The expected number of completed execution steps $\lambda(\mu_{11}, \pi_{15}, X_1, \theta_1) =$ 1.65508. We represent the distance resiliency of a workflow as a function $\Gamma_D : \mathbb{WES} \to \mathbb{R}$,

which given a workflow execution specification WES, returns the expected number of completed execution steps before WES becomes deadlocked. Given the set of all step-task mappings M and the set of all complete and valid plans Π_V, we define *distance resiliency* to be:

Definition 14 *The* distance resiliency *of a workflow execution specification* $WES = (WS, ES)$*, where* $ES = ((Z, \prec), \theta))$*, is:*

$$\Gamma_D(WES) = arg \max_{\substack{\mu \in M \\ \pi \in \Pi_V}} [\lambda(\mu, \pi, (Z \setminus \{\perp_x\}), \theta)] \tag{2.24}$$

Given WES, the goal of Γ_D is to find a complete and valid plan $\pi \in \Pi_V$ and step-task mapping $\mu \in M$ that maximise the function λ. If $\Gamma_D(WES) = |X|$ then a valid plan $\pi \in \Pi_V$ exists that is also complete and feasible, due to the fact that all execution steps are expected to be completed. This means Γ_D provides an alternative metric for analysing the workflow satisfiability problem introduced in Section 2.2. That is, if $\Gamma_D(WES) = |X|$ then $\Gamma_S(WES) = 1$. Using distance resiliency, a CISO could look to modify security constraints associated with tasks executable at the deadlocking step x_i. Any plans that can be found for all preceding steps can be considered executable valid plans.

2.4 Related Work

In this section we introduce current work related to the workflow satisfiability problem and workflow resiliency.

2.4.1 Workflow Satisfiability Problem

In some cases a pen and paper approach may be appropriate to solve the workflow satisfiability problem, especially when the number of workflow tasks, users, and security constraints is small. In many cases workflows are large, meaning a pen and paper approach soon becomes intractable. In light of this, the workflow satisfiability problem has become a well studied problem in the literature, with an aim to providing more efficient solutions. In [39], Crampton defines a model for workflow which incorporates separations and bindings of duty, and cardinality constraints which state certain tasks must be performed a certain number of times. An algorithm is presented which determines whether a valid user-task assignment exists for a workflow. Essentially, the algorithm tries to generate a pair of valid users which can

be assigned to the two tasks either separated, or bound by a constraint. If a valid pair can be found for every constraint the algorithm returns *true* indicating a valid solution to the workflow satisfiability problem exists. Details are also provided showing how the algorithm can be incorporated into a workflow reference monitor that ensures user-task assignments made at runtime are granted only if a solution to the workflow satisfiability problem exists for the remainder of the workflow. Computational complexity for the algorithm and analysis of its performance is also presented. Yang et al., in [190] consider the workflow satisfiability problem and its computational complexity along two directions. The first considers solving the workflow satisfiability problem for task constrained workflows with linear and parallel control patterns (task ordering), and more complex workflows incorporating choice points; the second considers role constraints imposed by a role based access control (RBAC) policy and computing a minimum set of roles that can complete a given workflow. Their results indicate the workflow satisfiability problem is in general intractable which motivates placing restrictions on control patterns in a workflow and access control policies in order to limit workflow satisfiability problem to tractable cases of practical interest.

Sun et al., in [163] also consider the workflow satisfiability problem in the context of RBAC where users are assigned to roles, and roles are assigned to tasks. Such a system removes the administration cost of assigning individual users directly to tasks [152]. Role-based constraints are considered, for instance a role r may have to be assigned to at least k users. RBAC constraints are also consider similar to separations and bindings of duty. For instance a user may not be assigned one role if already assigned to another (mutual exclusion), or they can only be assigned a role if assigned to another (prerequisite). An approach to analyse constraint consistency is provided as well as a method to find whether a solution to the workflow satisfiability problem exists, in other words can an assignment of users be found for a workflow whilst satisfying all RBAC constraints. The problem is formulated as the Boolean satisfiability problem (SAT), the problem of determining if there exists an interpretation that satisfies a given Boolean formula [37]. This approach enables the use of existing well known SAT solvers, e.g. SAT4J [107].

RBAC in relation to the workflow satisfiability problem has also be considered by Wang and Li in [179, 180]. They propose a natural extension to RBAC called role-and-relation-based access control, or R^2BAC which is used to determine a user's relationship with other users in addition to the RBAC constraints mentioned in [163]. These user relationships are considered before allowing a user to be assigned to a certain workflow task. For, instance, users assigned to separate roles in a traditional RBAC model may be related, e.g. parent - child. Solving the workflow satisfiability problem under this model is shown in general

to be NP-hard when considering separations and bindings of duty. The authors make the assumption that in practice, the number of tasks in a workflow will be small and apply tools from parameterised complexity to better understand the complexities of finding a solution to the workflow satisfiability problem [52]. Their results indicate that algorithms can be developed to efficiently solve the workflow satisfiability problem in these practical cases. The complexity bounds of Wang and Li's approach are improved by the algorithms developed by Crampton et. al., in [41]. The type of constraints under consideration is also extended to include counting constraints, a generalisation of cardinality constraints discussed in [39], and constraints based on hierarchal organisational structures.

Cohen et al., in [35] use the SAT solver SAT4J mentioned previously to encode the workflow satisfiability problem defined by Wang and Li in [179, 180] in order to compare its performance to an implementation of the generic algorithm presented by the authors in [36]. This algorithm was demonstrated to be fixed-parameter tractable for user-independent constraints such as separations and bindings of duty. This means the workflow satisfiability problem can be solved efficiently in cases where the values for the fixed parameter are small. Rather than generating all plans, equivalence classes are used to reduce the amount of information the algorithm must keep at each step. From the author's experiments they observe the algorithm implemented in C++ solves all generated workflow satisfiability problem instances with a *yes/no* decision whereas SAT4J does not. In many cases the algorithm was faster than SAT4J apart from those instances with few security constraints.

The algorithm proposed to solve the workflow satisfiability problem defined by Bertino et al., in [18] involves generating an actual valid user-task assignment for a workflow rather than deciding whether a workflow is satisfiable or not. This is described as planning and consists of two phases; role planning which tries to assign roles to workflow tasks whilst meeting constraints similar to [163]; and user planning which tries to assign individual users within those roles (we take the latter approach of assigning individual users in this work and do not consider roles). The algorithm proposed is recursive in nature, effectively building an assignment graph of all valid assignments for the workflow. A graph colouring technique for security constrained workflows to find a user-task assignment is presented by Kohler and Schaad in [99]. Their approach is to find an assignment that satisfies the workflow satisfiability problem with the minimal number of users motivated by a requirement to reduce cost. They define the notion of policy-based deadlocks where the execution of a workflow becomes blocked and cannot reach completion without administrative intervention. Two types of policy-based deadlock are considered; the first comes from conflicting constraints that prevent further execution of a workflow which can be handled by reference monitors as

suggested by Crampton in [39]; the second stems from tasks assigned to human users where a combination of constraints and user unavailability lead to situations where the workflow cannot complete (this is the same problem addressed by the work presented in Part 1 of this thesis). In terms of the second type of policy-based deadlock the authors illustrate when these situations can occur but do not consider how the likelihood of such situations can be quantified.

Most existing approaches address the workflow satisfiability problem from a computational point-of-view, by finding the most efficient algorithm to compute a plan, either returning a complete and valid plan if one exists, or nothing. Furthermore, existing approaches to the workflow satisfiability problem assume users to be always available, which in practice, may not always be case.

2.4.2 Workflow Resiliency

Workflow resiliency has been considered in different ways by a number of works appearing in the literature. Wang and Li took a first step towards quantifying the resiliency of a workflow by addressing the problem of how many users can become unavailable before a workflow becomes unsatisfiable. They also showed the problem of finding the resiliency of a workflow to be NP-hard, even for a workflow with simple security constraints [179, 180]. Wang and Li, first consider three common scenarios allowing them to classify how the availability of users may change during a workflow instance:

- *Level 1 (Static)* - a number of users are unavailable before the execution of a workflow instance while available users will not be unavailable during the execution. This scenario assumes execution will be carried out in a relatively short period of time, say fifteen minutes. It is assumed available users are unlikely to become unavailable during that execution and the set of available users is stable.

- *Level 2 (Decremental)* - users may be unavailable before or during the execution of a workflow instance, and unavailable users will not become available again. This scenario assumes execution will be carried out in a relatively long period of time, say within one day. It is assumed some users may not be available that day, or some available users may have to become unavailable before the workflow instance is complete and not be available again until the next day. The set of available users therefore becomes smaller and smaller during the day.

- *Level 3 (Dynamic)* - users may be unavailable before and during the execution of a workflow instance and unavailable users may become available again. This scenario assumes execution will be carried out over a long period of time, for example only a single task is performed each day. Since the set of users may differ from day to day, the set of available users may differ from execution step to execution step.

Our concept of an availability forecast θ, defined in Section 2.3.1 can be classified according to these three resiliency levels: Level 1 (static), the availability of a user is the same for all execution steps; Level 2 (decremental) the availability of a user for an execution step is equal or less than for the immediate preceding execution step; Level 3 (dynamic) the availability of a user for an execution step is equal, less or more than for the immediate preceding execution step. More formally, given a workflow execution specification $WES = (WS, ((Z, \prec), \theta))$, with $X = Z \setminus \{x_\perp\}$ being the set of execution steps, the availability forecast θ can be classified as static iff:

$$\forall u \in U, \forall x_i, x_j \in X, \theta(x_i, u) = \theta(x_j, u) \tag{2.25}$$

In the static case the availability of a user for any execution step x_i is the same as any other execution step x_j. Note, a full availability forecast θ, defined in Section 2.3.1 (Property 2.9), can be classified as static. An availability forecast θ can be classified as decremental iff:

$$\forall u \in U, \forall x_i, x_j \in X, x_i < x_j, \theta(x_j, u) \leq \theta(x_i, u) \tag{2.26}$$

The availability of a user for any execution step x_i in this case is either equal or less than their availability for execution step x_{i-1}. It follows that θ can be classified as dynamic in any other case. Wang and Li then go on to define a workflow to be k-resilient if an complete and valid plan can be found even when a maximum of any k users become unavailable during the workflow's execution. Furthermore, a workflow is defined to be k-resilient under the three resiliency levels. A workflow in this case is equivalent to our definition of a workflow specification $WS = ((T, <), U, (A, S, B))$ (Section 2.1.4).

Definition 15 *Given an integer $k \geq 0$, a workflow specification $WS = ((T, <), U, (A, S, B))$ is* static $k-$resilient *if for all $U' \subseteq U$, such that $|U'| = k$, $WS = ((T, <), (U \setminus U'), (A, S, B))$ is satisfiable.*

In [180], the establishment of decremental k-resiliency, and dynamic k-resiliency below, is described as a two-player game. Note, we write $\leftarrow$ to denote an assignment operation.

Definition 16 *Given an integer $k \geq 0$, a workflow specification $WS = ((T,<),U,(A,S,B))$ is* decrementally $k-$resilient *iff Player 2 can always win the following two-player game when playing optimally:*
Initialisation: $U_0 \leftarrow U$, $T_0 \leftarrow T$, $k_0 \leftarrow k$ and $i \leftarrow 1$
Round i of the game:

1. *Player 1 selects a set U'_{i-1} such that $\left|U'_{i-1}\right| \leq k_{i-1}$, $U_i \leftarrow (U_{i-1} \setminus U'_{i-1})$ and $k_i \leftarrow (k_{i-1} - \left|U'_{i-1}\right|)$.*
2. *Player 2 selects a task $t_i \in T_{i-1}$ such that $\forall t_j \in T, t_j < t_i \Rightarrow t_j \notin T_{i-1}$.*
   Player 2 selects a user $u \in U_i$.
   $\pi(t_i) \leftarrow u$ and $T_i \leftarrow T_{i-1} \setminus \{t_i\}$.
   If π is not a valid partial plan with respect to the sequence $t_1,\ldots,t_i$ then Player 1 wins.
3. *If $T_i = \emptyset$ then Player 2 wins, otherwise, let $i \leftarrow i+1$ and the game goes on to the next round.*

Definition 17 *Given an integer $k \geq 0$, a workflow specification $WS = ((T,<),U,(A,S,B))$ is* dynamically $k-$resilient *iff Player 2 can always win the following two-player game when playing optimally:*
Initialisation: $U_0 \leftarrow U$, $T_0 \leftarrow T$ and $i \leftarrow 1$
Round i of the game:

1. *Player 1 selects a set U'_{i-1} such that $\left|U'_{i-1}\right| \leq k_{i-1}$ and $U_i \leftarrow (U_{i-1} \setminus U'_{i-1})$.*
2. *Player 2 selects a task $t_i \in T_{i-1}$ such that $\forall t_j \in T, t_j < t_i \Rightarrow t_j \notin T_{i-1}$.*
   Player 2 selects a user $u \in U_i$.
   $\pi(t_i) \leftarrow u$ and $T_i \leftarrow T_{i-1} \setminus \{t_i\}$.
   If π is not a valid partial plan with respect to the sequence $t_1,\ldots,t_i$ then Player 1 wins.
3. *If $T_i = \emptyset$ then Player 2 wins, otherwise, let $i \leftarrow i+1$ and the game goes on to the next round.*

The idea of rounds, used in Definitions 16 and 17, is equivalent to our notion of execution steps, defined by the set $X = Z \setminus \{x_\perp\}$, such that each round/step correspond to assigning the execution of one task to one user, and the execution of all tasks is assigned sequentially. It follows, the sequence of all possible rounds $i,\ldots,n$ is equivalent to an execution schema $(Z,\prec)$ defined in Section 2.3.1, where $i,\ldots,n \equiv x_i,\ldots,x_n$.

In practice, finding an executable complete and valid plan for all possible k sized subsets of unavailable users may be demanding and often unmanageable. Furthermore, in definitions of k-resiliency, each k sized subset is assumed equally likely which may not be realistic in many real-life cases. Also, k-resiliency offers a somewhat binary result by indicating a

workflow to be k resilient or not. What may not be clear to a CISO is how much security policy modifications actually impact the resiliency of a workflow. For instance, a CISO could add 2 separation of duty constraints to a security policy, both of which have an impact by causing a workflow execution specification WES to go from being 1 resilient to 0 resilient. But is the impact of such constraints equal? Can certain constraints have more of an impact than others on the resiliency of a workflow? To help a CISO answer these questions we take a probabilistic approach to workflow resiliency. Rather than analysing how many k users can become unavailable, quantitative resiliency considers the probability of a workflow completing without having to violate the security policy, and does so under the assumption that user availability is probabilistic. We believe quantitative resiliency would allow a CISO to understand the impact for particular security modifications in a fine grained manner by finding a workflow's quantitative resiliency before and after the modification, and analysing the degree of change.

In another response to the resiliency problem, Wang et al., introduce the notion of resiliency policies in the context of access control systems, with an aim to overcome the problem of user unavailability, by enabling access rather than restricting it [109]. Resiliency policies state a minimum number of users that must hold particular permissions thereby introducing an acceptable level of redundancy into the system. The resiliency checking problem is introduced and an algorithm defined to determine whether a given access control state satisfies a resiliency policy, a problem shown to be NP-hard in the general case. A methodology is also provided to check the consistency of resiliency and separation of duty policies to ensure a user is not given too many permissions by the resiliency policy therefore enabling them to violate the separation of duty policy.

A similar approach is presented by Paci et al., in [139] who consider introducing resiliency constraints into processes containing web services and user performed tasks, expressed in the Web Services Business Process Execution Language (WS-BPEL) [92]. Resiliency constraints state the minimum number of users that must be authorised to execute a task and therefore provide some assurance that the workflow will still complete if some users become unavailable. A process is said to be user failure resilient if a user-task assignment can be found that meets both resiliency and security constraints. The set of all possible valid assignments is generated and stored to decided which user should be assigned to a task if the pre-assigned user becomes unavailable at runtime. As previously mentioned Wang and Li stated a workflow as k-resilient if it could still complete when any k users became unavailable at any point in the workflow. This work is similar in that the constraints ensure an assignment can still be found to complete the workflow even when k users become

unavailable. Lowalekar et al., in [110] show multiple assignments may exist with the same level of k-resiliency and that it may be necessary to choose the most favourable one. Particular workflow security attributes are assessed and an assignment chosen that minimises the diffusion of business knowledge across all users. Each role assigned a task is itself assigned an ordered set of potential role delegates, of which one becomes active when the parent role becomes unavailable. Criticality values are used to decide whether the delegation may proceed or the task be suspended. By considering resiliency in terms of the number of users who can be come unavailable, these approaches bring a similar limitation to Wang and Li's work as they do not quantify the likelihood of a workflow completing under an assumption of user unavailability which is necessary in order to analyse the potential impact of security policy changes.

Cloud computing with its seemingly unlimited supply of on demand computing resources offers an attractive platform to deploy and efficiently execute workflows, especially those processing large data sets used in domains such as eScience [79]. One major barrier to cloud computing adoption are a number of security issues including the sharing of processing infrastructures often in distant and sometimes unknown locations, and administrated by third parties [119]. In [181], Watson considers the partitioning of workflows across federated clouds whilst meeting security constraints, meaning sections of a workflow considered security critical can be kept in-house whilst other sections can de outsourced to public clouds. A methodology is presented which applies the Bell-LaPadula multi-level security model [15] to constrain how workflows can be partitioned across a set of cloud platforms, and how all deployments can be found that satisfy the security model. This problem is similar to solving the WSP where a valid cloud deployment is essentially an assignment of tasks to clouds (akin to users). Each task is deployed based on where the previous task is deployed meaning complex security constraints such as separations and bindings of duty are not considered. A cost model is applied to choose the optimal deployment with the lowest cost but quantitative measures indicating success rate are also not considered. This work is extended by Zhenyu in [183] by assuming cloud platforms may fail during workflow execution and use a pre-generated cost weighted graph of all partitioning options in order to find the cheapest alternative valid deployment to complete the workflow, if one exists. This work does not however consider the future 'availability' of cloud platforms when choosing an alternative deployment which could reduce the likelihood of having to redeploy parts of a workflow. Graphs are also used in [123] by Massacci et al., used to assess whether a process is resilient to both users becoming unavailable and changes to user privileges at runtime. Acyclic hypergraphs are used to model a process where each node represents

either an organisational goal, tasks, roles and users. The graph is traversed to see if a path exists linking goals, actions, roles and users but provides no quantitative measures regarding predicted success rate.

A cost-based approach to increase the resiliency of a workflow has been suggested by Basin et al., in [11] and [12]. The authors analyse the trade-off between security and business objectives, and the avoidance of deadlock in workflows caused by a combination of a security policy and the unavailability of certain users. They define the existence of an obstruction free enforcement mechanism as a decision problem that overcomes scenarios where no valid user-task assignment exists by reallocating roles to users at runtime to satisfy security constraints. A new assignment of users to roles is calculated with the minimum cost to risk (of a user taking on a role), administration and maintenance. This is feasible in certain business domains but may have limited application in workflows where roles are more specialised; although the risk of adopting a role is considered, is adding an untrained user to the role doctor for example to satisfy a security policy better than overriding the policy and enabling a constrained but qualified doctor? In [40], Crampton et al., define an approach to find the 'best' or 'least bad' user-task assignment for a workflow that is unsatisfiable. The approach, called Valued WSP assumes security constraints and user-task permissions can be violated, or overridden in order to complete a workflow. Violation risk is expressed as a cost and an algorithm is defined to find a user-task assignment that completes the workflow with the minimum cost, shown to be fixed-parameter tractable with user-independent security constraints. Although this approach is applied to the WSP, it could be adapted to find the least bad user-task assignment that completes a workflow in those instances where user unavailability would otherwise force early termination.

Overriding security constraints to enable a workflow to complete has been considered in other works appearing in the literature. Notably, Wainer et al., in [178] consider the explicit overriding of security constraints by defining a notion of privilege by assigning levels of priority to each constraint and maximum override levels to user roles. In the presence of exceptions such as user unavailability, users can potentially perform tasks they would not normally be permitted to do by overriding constraints whose priority level is equal to or below their given override level. Brunel et al., in [23] suggest a a security policy may still be satisfied even when some security constraints may be violated, for instance to ensure a workflow completes when certain users become unavailable at runtime. The authors define an approach of violation management that allows a security constraint to be overridden but stipulates conditions or obligations that must be fulfilled at a later time that once completed ensure compliance with the security policy is till met. El Bakkali

suggests enhancing resiliency through the delegation of tasks and the placement of criticality values over workflows [55, 56]. User delegates are chosen on their suitability but may lack competence; this is considered the 'price to pay' for resiliency in a similar way to the risk of a user adopting a role in the work by Basin et al., [11, 12]. As delegation takes place at a task level it is not currently clear whether a workflow could still complete while meeting security constraints. Delegation is also consider to overcome situations when users become unavailable by Crampton and Morisset in [43]. The authors suggest a mechanism that can automatically respond to the unavailability of users by delegating a task to the most qualified user available, however as this work is placed in the context of access control it does not consider the workflow satisfiability or resiliency problems.

Similarly to the workflow satisfiability problem most current approaches address workflow resiliency from a computational point-of-view. The current literature on workflow resiliency also tends towards user availability being a binary notion in the sense that users are either available or not, and does not consider availability to be probabilistic as we do in our approach. In Chapter 3 we introduce a decision making approach for analysing the satisfiability and resiliency of a workflow specification. We use this single framework to provide metrics for workflow satisfiability and resiliency, and aid CISOs in designing workable security policies. This is something the current literature on the workflow satisfiability problem and workflow resiliency does not consider to the best of our knowledge.

2.5 Summary

When designing a security policy, a CISO may need to analyse how a policy design is likely to impact the completion of a workflow under different scenarios of user availability. Such analysis at design time, rather than finding out the impact at runtime, could avoid unworkable policies being put into practice. A security policy can have an impact if it prevents any available users from being assigned the execution of a task, thus causing a workflow to become deadlocked. Such a scenario may occur when all authorised users are unavailable. A security policy therefore has an impact if a workflow cannot be completed in a valid way, that is, without a policy violation or by necessitating some authorised remediation action, such as a policy override. Intuitively, as the availability of users decreases, a policy is likely to have more impact as it becomes harder to ensure valid workflow completion.

In this chapter we considered finding the impact of a workflow security policy under different levels of user availability. First, we assumed a CISO may need to know whether a workflow is satisfiable, in other words, whether a security policy has an impact on workflow

completion under the condition that all users are available. If a workflow is found to be unsatisfiable, it can never be completed without violating the security policy, regardless of user availability. Second, a CISO may also need to know whether a workflow is resilient, that is, whether a workflow can still be completed in a valid way under the condition that users may be unavailable. We defined two new metrics for workflow resiliency called *quantitative* and *distance resiliency*. Quantitative resiliency indicates the maximum probability of valid workflow completion whilst assuming user availability to be probabilistic. Distance resiliency indicates the maximum expected number of steps completed before a workflow becomes deadlocked. A step in this case is the point in a workflow where the execution of a task is assigned to a user.

Furthermore, we have given a formal definition of an execution workflow specification which exposes the notion of probabilistic user availability during the execution of a workflow. In Chapter 3 we use our definition of workflow and encode the stepwise process of assigning the execution of tasks to users as a Markov decision process. Using this single framework we show how the satisfiability, quantitative resiliency, and distance resiliency of a workflow can be generated. In Chapter 4 we show how these metrics can be computer generated by encoding the Markov decision process in the probabilistic model checker PRISM, and how the impact of a security policy on the completion of a workflow can be analysed. We believe the metrics can be used by a CISO for fine grained analysis of how a policy design is likely to impact the satisfiability and resiliency of a workflow, the concepts introduced in this chapter. We suggest a CISO can analyse the likely impact of a policy modification, such as adding a separation of duty constraint, by generating a metric with and without the modification and observing the degree of change.

Chapter 3

Generating Workflow Metrics

In general, metrics are standards of measurement, commonly used by organisations, by which efficiency, performance, progress, or quality of a plan, process, or product can be assessed. In Chapter 2 we defined a workflow execution specification and introduced workflow satisfiability and workflow resiliency. We then proposed quantitative measures for these concepts in the form of *workflow metrics*. We suggest a CISO can use these metrics to assess the satisfiability and resiliency of a workflow, and undertake fine grained impact analysis of modifications to a workflow's security policy by generating a metric with and without the modification, and observing the degree of change.

In this chapter we first consider the execution of a workflow to be a stepwise decision making problem consisting of discrete steps where an assignment decision and event is actioned. Assignment decisions represent the selection of a task and a user, whilst assignment events represent the assignment of a task's execution to a user. Event success is probabilistic as the availability of users is considered to be probabilistic. We describe the concept of decision making processes before introducing a Markov decision process (MDP) which is an analytical tool for modelling and studying stepwise decision problems. We then show how a workflow execution process can be modelled as an MDP before defining two reward functions, one for generating quantitative satisfiability and resiliency, the other for generating distance resiliency. Reward functions place motivational rewards on specific process transitions, and solving the MDP consists of finding the expected maximum reward in the initial state of the process for a given reward function. Using this single MDP framework we show how the satisfiability and resiliency metrics can be generated by changing the reward function. In Chapter 4 we show how these metrics can be computer generated by encoding the Markov decision process in the probabilistic model checker PRISM, and how the impact of a security policy on the completion of a workflow can be analysed.

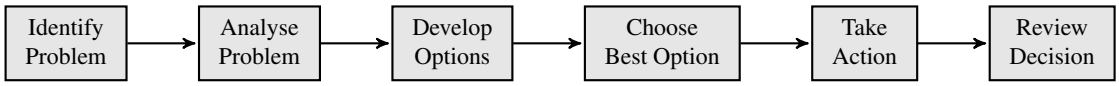


Fig. 3.1 A typical normative process whose outcome is a sequence of decisions made rationally after careful reasoning of the expected reward.

3.1 Decision Making Processes

In practice, workflows are often complex structures whose execution is carried out in a stepwise manner. At each step, an *assignment decision* must be executed, selecting the next task t and a user u to execute t, followed by the execution of an *assignment event* which assigns the execution of t by u. We assume if u is available then t will be executed, if u is unavailable then t will not be executed. User availability means assignment decisions must often be resolved under uncertainty, in the sense of not knowing with certainty which users will be available at future steps. Assignment decisions can therefore force the process to a step where no user authorised to execute a task is available, and therefore deadlock [11].

To expose this notion, we consider a workflow execution process to be a sequential decision making process partly under the control of a *process agent* executing the entire process, and partly random. That is, a process agent controls assignment decisions regarding task and user selection, but is not fully in control of whether a selected user will be available to execute a selected task. Executed assignment events may or may not be successful in this case. We also consider a workflow execution process to be an optimisation problem in the sense of resolving assignment decisions in a way that maximises the possibility of finding an optimal plan π, that is a plan that is complete and valid, and whose feasibility $\rho(\pi)$ is higher than any other complete and valid plan. In Section 3.2 we show how the workflow satisfiability and resiliency metrics introduced in Chapter 2 can be generated with this decision making approach.

3.1.1 Decision Making

In general, decision making is the process of selecting between two or more courses of action in order to solve a problem. Correct decisions provide longterm opportunities for accomplishment, growth, and success, while wrong decisions can lead to loss, instability, and failure. Indeed, without decisions, activities would not be possible, resources would not be put to use, and problems would not be solved. Two central concepts exist in decision theory, these are *preferences* and *options*. Roughly speaking, a process agent 'prefers' the 'option' of action A over action B when A is more desirable than B for the process agent in

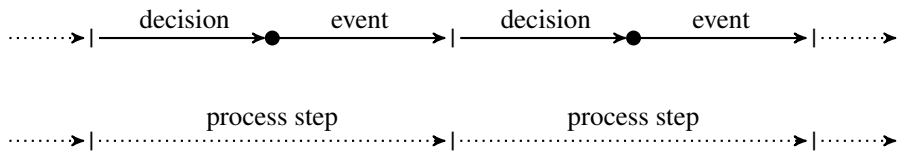


Fig. 3.2 Partial example of a decision process showing pairs of decision and event actions that together form a sequence of decision process steps, which form the decision process itself.

question. A process agent's preference between actions may be influenced by many factors, for instance biases and expected payoff or reward. Decision making made rationally after careful reasoning of the expected reward is described as *normative*, and is often a step by step process, identifying a problem, gathering information, and assessing action options as shown in Figure 3.1. Taking a stepwise approach in this way can help increase the chances of choosing the best actions in the long-term. Alternatively, bad choices can often be made, for example, due to insufficient information or unawareness of the consequences.

3.1.2 Decision Process

In this section we introduce general terms and concepts for a decision process.

Process States

A *decision process* can be in one particular configuration at any point, called a *state*. We write $\mathbb{S}$ for the set of all states describing all possible configurations of a process, where $s_0 \in \mathbb{S}$ is the initial process state. A decision process can only be in one state at any point in time, called the *current state*. As a decision process progresses, it moves through a series of *states*, each one being unique in the sense they contain unique contextual information about the process. We assume a state to at least include a *process history* of all actions executed in all previously visited states. We therefore write $s.\mathtt{h}$ to denote the process history in state s.

Process Actions

A decision process, is a discrete sequence of *actions* where a decision is made, an event occurs, another decision is made, another event occurs, and so on. Roughly speaking, a *decision* $\mathtt{d}$ is an action representing the choice of what to do next in a process, chosen from a finite set of alternative actions, whilst an *event* $\mathtt{e}$ is an action representing some outcome of a decision. It follows that each decision triggers some corresponding event, and we reduce

each corresponding decision/event pair to a discrete *process step* as illustrated in Figure 3.2. In order for a process to terminate we also consider a process termination action `t`. More formally we write $\mathbb{D}$ for the set of decision actions, $\mathbb{E}$ for the set of event actions, $\mathbb{T}$ for the set of termination actions, and $\mathbb{A} = \mathbb{D} \cup \mathbb{E} \cup \mathbb{T}$ for the set of all process actions.

Process Transitions

The execution of decision, event, and termination actions causes the process to transition from one unique state to the next. All possible process transitions are represented by a *transition function* $\mathbf{p} : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to [0,1]$, where $\mathbf{p}(s,a,s')$ returns the probability of reaching state s' when the action a is executed in state s. A transition function $\mathbf{p}$ therefore defines which states a process can transition to. Executing an action a in state s, such that $\mathbf{p}(s,a,s') = 0$, for any state s', means the execution of a will not cause the process to transition to another state. If $\mathbf{p}(s,a,s') = 0$ for any action a and any state $s' \neq s$, then the process cannot transition to any other state from s. In this case s is a *termination state* which is an absorbing state, that is a state that, once entered, cannot be left.

If in a state s there exists two (or more) actions $a, a' \in \mathbb{A}$, such $\mathbf{p}(s,a,s') = 1$ and $\mathbf{p}(s,a',s'') = 1$, then a non-deterministic choice exists between executing a or a'. Note, executing a in s can move the process to only one state s' with a probability of 1. We apply this notion to decisions and say that a choice between executing decision actions in a state s is non-deterministic and must be resolved by the *process agent* executing the entire process. If in a state s there exists an action a, such that $\mathbf{p}(s,a,s') = p$ and $\mathbf{p}(s,a,s'') = 1-p$ then by executing a the process can move to one of two states, that is, to s' with a probability of p and s'' with a probability of $1-p$. We apply this notion to events and say the success of an event action is under the influence of external factors coming from the surrounding environment, and is therefore probabilistic. This is in the sense that if an event action is successful in state s, the process moves to s' with probability p, or if the event action fails, the process moves to s'' with probability $1-p$.

The choice of decisions which can be executed in a state s are partly under the control of the process agent, and partly random due to the influence of external factors. The process agent has part control in the sense that they 'control' what decision is executed in s, and thereby move the process to a new state s'. External factors have part influence as they partly 'influence' previous transitions which caused the process to be in state s where the process agent must resolve the non-deterministic choice. It follows that a process agent must resolve non-deterministic choices under the uncertainty introduced by external factors, in the sense that the agent is uncertain which future states will be reached, and therefore what

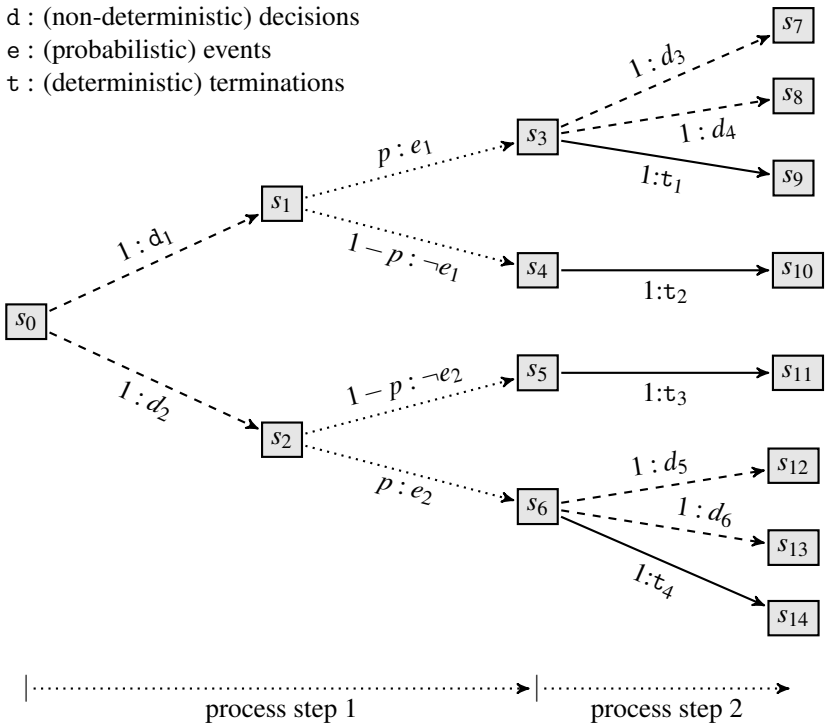


Fig. 3.3 Partial example of a decision making process consisting of non-deterministic decisions, probabilistic events, and deterministic termination actions, and where s_0 is the initial process state.

future decisions can be executed. In a similar way to decisions, an event executed in a state s is partly under the control of the process agent, and partly under the influence of external factors. The agent has part control in the sense that they partly 'control' previous transitions which caused the process to be in state s where the event is executed. External factors have part influence as they 'influence' whether the event executed in s succeeds or fails.

Figure 3.3 shows a partial example of a decision process where s_0 is the initial state. In s_0, a non-deterministic choice exists between executing actions $\mathtt{d}_1$ and $\mathtt{d}_2$ which must be resolved. The process agent can control whether the process moves to either state s_1 or s_2 by choosing to execute either $\mathtt{d}_1$ or $\mathtt{d}_2$ respectively. Let us assume the process agent chooses to execute $\mathtt{d}_1$ and the process moves to s_1 with a probability of 1. An event $\mathtt{e}_1$ may be executed at s_1 which moves the process to state s_3 with the probability of success p, or moves the process to state s_3 with the probability of failure $1 - p$. We label the transition for event a_1 failing as $\neg e_1$. These two probabilistic transitions illustrate a point in the process where the process agent does not control which state the process moves to.

If the process moves to s_3 and $s_3.\mathtt{h}$ is valid according to some criteria, a non-deterministic choice exists between executing $\mathtt{d}_3$ and $\mathtt{d}_4$. If $s_3.\mathtt{h}$ is invalid, a termination action $\mathtt{t}_1$ is

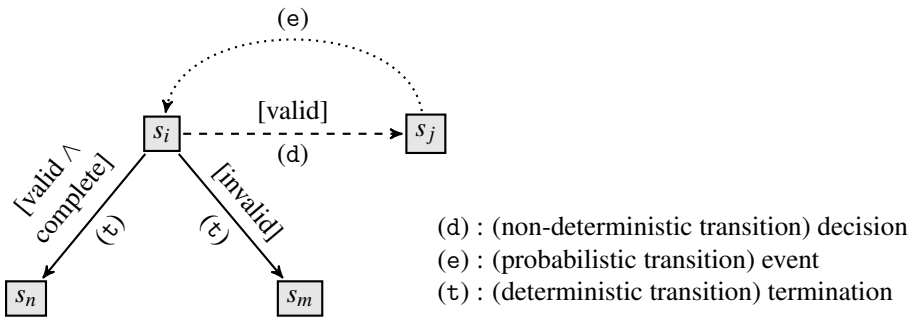


Fig. 3.4 Process step of a decision process consisting of non-deterministic decisions, probabilistic events, and deterministic termination actions, where s_n is a successful termination state and s_m is an unsuccessful termination state.

executed and the process ends unsuccessfully by moving to the termination state s_9 with a probability of 1. Note that, although event $\mathtt{e}_1$ may be successful, it may be incorrect according to some criteria, making $s_3.\mathtt{h}$ invalid. One could say in this case the choice to execute decision $\mathtt{d}_1$ was a bad one. Similarly, $s_4.\mathtt{h}$ is invalid as event $\mathtt{e}_1$ has failed, meaning the process ends unsuccessfully by moving to the termination state s_{10}. This example illustrates how the choice to execute an action in an early state can restrict the choices of decision actions in later states. For instance at s_0, the process agent has a chance of choosing any of the decsion actions $\mathtt{d}_1, \ldots, \mathtt{d}_6$ later in the process, depending on the choice between $\mathtt{d}_1$ and $\mathtt{d}_2$ in state s_0. If $\mathtt{d}_2$ is chosen the choice to execute either $\mathtt{d}_3$ or $\mathtt{d}_4$ will never be available as state s_3 will not be reached. Furthermore, if the goal of the agent is to get to the end of the process, and the probability of $\mathtt{e}_1$ succeeding is higher than the probability of $\mathtt{e}_2$ succeeding, choosing $\mathtt{d}_1$ over $\mathtt{d}_2$ would seem the preferred choice.

Process Steps

The notion of process steps allows a decision process to be presented as a state transition diagram shown in Figure 3.4, which represents a process step i. Each node represents a state the process can move to in step i. We consider the presence of termination actions (t) to cause deterministic transitions from state s_i to either s_n or s_m, shown as solid arcs; decision actions (d) to cause non-deterministic transitions from state s_i to a state s_j, shown as a dashed arc; and event actions (e) to cause probabilisitic transitions from state s_j to a state s_i, shown as a dotted transition. A process step i starting in state s_i proceeds as follows:

1. (a) If the process history $s_i.\mathtt{h}$ is invalid, based on some criteria, a termination action t is executed moving the process from s_i to a *termination state* s_m. The process

ends unsuccessfully in state s_m.

(b) Else, if the process history s_i.h is valid based on some criteria and complete, that is all steps have been executed, a termination action t is executed moving the process from s_i to a termination state s_n. The process ends successfully in state s_n.

(c) Else, the process history s_i.h is valid, based on some criteria, and a chosen decision action d is executed moving the process from s_i to s_j. In state s_j, s_j.h = s_i.h $\leftarrow$ d, meaning the process history in s_j is the process history of the previous state s_i plus the decision d executed in s_i.

2. In state s_j an event action e is executed moving the process probabilistically from s_j to a state s_i. Depending on the probabilistic success of the executed event, the process history s_i.h may or may not be valid based on some criteria. The process then moves to the next process step $i+1$.

The problem the process agent faces is to resolve each non-deterministic choice in state s_i, for each progressive process step i, in such a way that maximises the chance of reaching the termination state s_n, where the process ends successfully. To do so requires the decision action choices of the process agent to be optimal choices.

Decision Policy

In general, a 'policy', not to be confused with a security policy, is a detailed undertaking that attempts to provide an answer for all possible contingencies. A policy is particularly important when a process agent is automated, such as a workflow management system, and operating in environments where information on which decisions must be based is dynamic. Hence, in the case of a decision process we consider a *decision policy* to be a function $\delta : \mathbb{S} \rightarrow \mathbb{A}$, which given a state s returns the action to execute in s. Note, the action returned by a decision policy $\delta(s)$ is only dependant on the state s, and not on any other states. Each decision policy δ therefore resolves all non-deterministic choice by defining a single course of sequential actions among several possible courses of sequential actions. In respect to executing event actions in a state s, a policy δ returns the event to execute, however as previously stated, the process moves to a new state s' with a probability p. By implementing a decision policy, a process agent can partly control the process towards a longterm outcome, which may be the result of executing thousands of actions, or it may be the result of executing just a few actions. In terms of the execution process we assume the preferred outcome of the

process agent is for the process to end in a successful termination state, shown as state s_n in Figure 3.4. Because the outcome can be influenced by external factors, the problem is to find the optimal decision policy from among all possible decision policies. An optimal decision policy in this case, is the decision policy which, if implemented, maximises the chance of the process ending in a successful termination state.

Figure 3.5 shows an example decision process for which there exists three possible decision policies δ_1, δ_2, and δ_3. Implementing δ_1 means $\delta_1(s_0) = \mathtt{d}_1$ and the process ending in either termination state s_{10} or s_{11}. Indeed, implementing any one of these three policies means the process can end in one of two termination states. Decision policy δ_2 gives the process the highest chance of reaching a termination state, in this case s_{12}. This is because the probability of event $\mathtt{e}_2$ being successful, that is $\mathbf{p}(s_2, \mathtt{e}_2, s_6) = 0.8$, is higher than $\mathtt{e}_1$ executed under δ_1, and $\mathtt{e}_3$ executed under δ_3. In fact the maximum probability or reaching any termination state is 0.8 under decision policy δ_2. However, if the decision process ends in s_{12} it has been executed unsuccessfully due to the process history $[\mathtt{d}_2]$ being invalid according to some criteria. The process can end successfully in only two termination states s_{10} under δ_1, and s_{14} under δ_3. The maximum probability of the process ending in a successful termination state is 0.7 under δ_3, making δ_3 the optimal decision policy in this case. In Section 3.1.3 we introduce an analytical tool suitable for finding an optimal decision policy for the decision processes we consider.

3.1.3 Markov Decision Process

A *Markov decision process* (MDP) is an analytical tool which provides a framework for modelling and studying a wide range of multi-step decision, or optimisation problems where the results are somewhat random and partially under the control of a decision maker [16, 83]. An MDP consists of a set of states and will be in one of these states at each discrete time step. The transition from one state in the MDP to another is governed both probabilistically and by the decision maker's choice to execute an action from those available in the current state. Therefore the next state depends on the current state and the executed action, but is independent of all previous states and actions. This independence satisfies the *Markov property* [121].

An MDP extends the decision processes introduced in Section 3.1.2 by associating each transition between states with a particular motivational reward, which is returned to the decision maker as each transition occurs. Solving an MDP consists of defining a policy that specifies the action the decision maker will execute when in a particular state of the process.

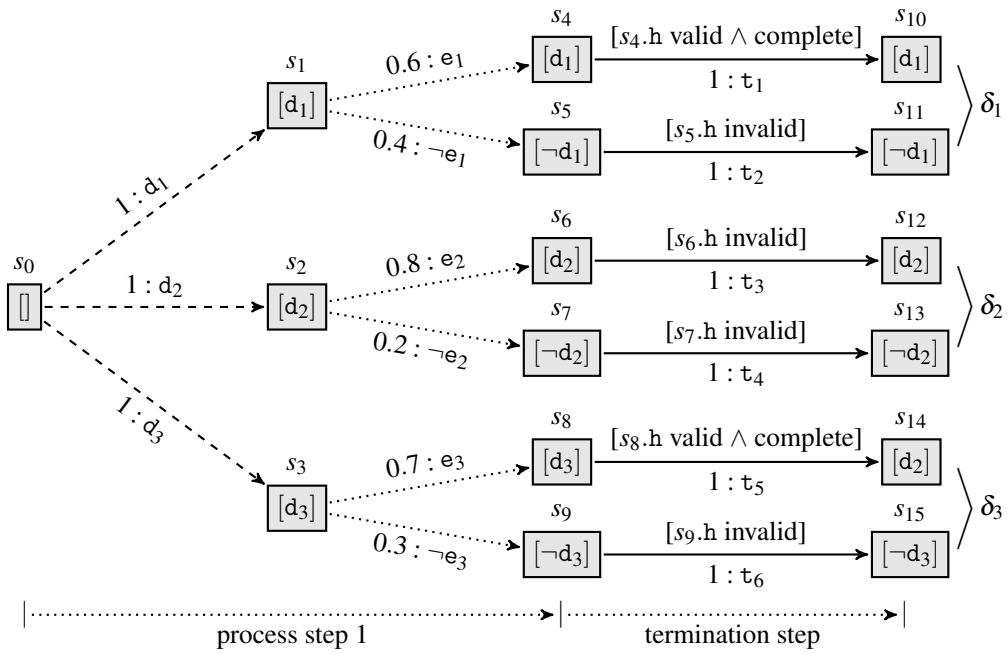


Fig. 3.5 Example decision making process with three possible decision policies δ_1, δ_2, δ_3, where s_{10} and s_{14} are successful termination states, and δ_3 is the optimal decision policy, maximising the probability of the decision process reaching a successful termination state, which is s_{14} in this case.

The objective is to define an optimal policy, that is a policy that maximises the expected reward collected by the MDP, such that the action to be executed in a particular state is the one that returns the highest expected longterm reward. Collecting the rewards is a somewhat inductive process which involves finding all reachable future states from the current state and collecting the rewards associated with the transitions taken to reach those states. Many approaches have been devised to solve an MDP in order to find the optimal policy, most notably policy iteration [16, 83], value iteration [16] and, when probabilities and rewards are unknown, reinforcement learning [165].

When applied to the decision processes introduced in Section 3.1.2, an MDP is formally represented as tuple $(\mathbb{S}, \mathbb{A}, \mathbf{p}, \mathbf{r})$ where $\mathbb{S}$, $\mathbb{A}$, and $\mathbf{p}$ are a set of states, set of actions and transition function also introduced in Section 3.1.2. The association of rewards to transitions is defined by a *reward function* $\mathbf{r} : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to \mathbb{R}$, such that $\mathbf{r}(s, a, s')$ describes the reward associated with executing action a in state s causing the process to transition to state s'. A *policy* for an MDP is equivalent to a decision policy introduced in Section 3.1.2, that is a function δ associating each state with an action. The *value* of a policy δ for an MDP in state

s is given to be:

$$V^{\delta}(s) = \sum_{s' \in \mathbb{S}} \mathbf{p}(s, \delta(s), s') \mathbf{r}(s, \delta(s), s') + \beta \sum_{s' \in \mathbb{S}} \mathbf{p}(s, \delta(s), s') V^{\delta}(s')$$

We call $V^{\delta} : \mathbb{S} \to \mathbb{R}$ the *value function* of δ. More or less weight to 'future' values is given by the discount factor β, where $0 \leq \beta < 1$. The *optimal* policy $\delta^* : \mathbb{S} \to \mathbb{A}$ is then defined to be:

$$\delta^*(s) = arg\max_{a \in \mathbb{A}} \left[\sum_{s' \in \mathbb{S}} \mathbf{p}(s, a, s') \, \mathbf{r}(s, a, s') + \beta \sum_{s' \in \mathbb{S}} \mathbf{p}(s, a, s') V^*(s') \right]$$

Here $V^* : \mathbb{S} \to \mathbb{R}$ is the value function of δ^*. Note that since $\beta < 1$, the optimal policy is always defined, even when $s = s'$. It is possible to show that $V^*(s) \geq V^{\delta'}(s)$, for any other policy δ' and any state s, and we refer to [16] for further details about the proof of this property and further details on the notion of MDP. For clarity the parts of the optimal policy function δ^* are as follows:

optimal action to execute in s — $\delta^*(s)$

analyse executing all actions in s — $arg\max_{a \in \mathbb{A}}$

expected immediate reward for executing a in s — $\sum_{s' \in \mathbb{S}} \mathbf{p}(s, a, s') \, \mathbf{r}(s, a, s')$

expected longterm reward for executing a in s — $\beta \sum_{s' \in \mathbb{S}} \mathbf{p}(s, a, s') V^*(s')$

3.2 Computing Workflow Metrics

In this section we introduce an approach to finding the satisfiability and resiliency of a workflow, by modelling a workflow decision process as a Markov decision process (MDP). We then define reward functions for each of the satisfiability and resiliency problems we consider. Generating the workflow metrics described in Chapter 2 therefore consists of solving the MDP with the appropriate reward function.

3.2.1 Workflow Markov Decision Process

We now use the concepts introduced in Section 3.1 to show how a workflow execution process can be modelled as a Markov decision process (MDP). Henceforth we will refer to such an MDP as a *workflow Markov decision process* (MDP_W). A process agent in the case of

workflow is an automated workflow management system which controls decisions regarding task and user selection according to a decision policy δ. The optimal value function V^* of the MDP_W, returns the workflow metrics we require when using suitable reward functions. Therefore it is necessary to solve the MDP_W to find the optimal policy δ^* that maximises V^*.

Workflow Process States

In Section 3.1.2, the set of process states is denoted as $\mathbb{S}$. Given a workflow execution specification $WES = (((T,<),U,(A,S,B)),((Z,\prec),\theta))$, we say any state $s \in \mathbb{S}$ in an MDP_W is of the form $(x,(t,u),\pi)$ where $x \in Z$ is a workflow step, (t,u) is an assignment decision $\mathtt{d} \in \mathbb{D}$ representing the action of choosing a user $u \in U$ to be assigned the execution of a task $t \in T$, and π is a plan. Therefore, given any state $s = (x,(t,u),\pi)$, its process history $s.\mathtt{h} \equiv \pi$. The initial state of an MDP_W is $s_0 = (x_1, null, \pi)$, where x_1 is the first execution step, null is an assignment decision (t,u) associated a zero value, and $\pi : R \to U$, where $R \subset T$, is a valid partial plan that does not assign the execution of any task to any user. In other words, $R = \emptyset$. A termination state is of the form $s = (x_i, null, \pi)$, where $x_i \neq x_1$ is any first workflow step except the first, *null* is a null assignment decision, and π is a plan. A termination state of the form $s = (x_\perp, null, \pi)$ is successful iff the plan π is complete and valid. As all workflow steps must complete for π to be complete it follows a state $s = (x_i, null, \pi)$ can be a successful termination state only iff $x_i = x_\perp$.

Workflow Process Actions

In Section 3.1.2, we assumed a decision process to be a sequence of discrete process steps where either a decision action $\mathtt{d}$ or event action $\mathtt{e}$ is executed, or a termination action $\mathtt{t}$ is executed. Process steps in a decision process are represented in an MDP_W as the set of workflow steps Z for a workflow execution specification $WES = (((T,<),U,(A,S,B)),((Z,\prec),\theta))$ where $x_\perp \in Z$ is the termination step, and $X = Z \setminus \{x_\perp\}$ is the set of execution steps. Also in a decision process, the set $\mathbb{A} = \mathbb{D} \cup \mathbb{E} \cup \mathbb{T}$ was given to be the set of all process actions. In the case of an MDP_W we consider the set $\mathbb{D}$ to be the set of all assignment decision actions (t,u). Each assignment event $\mathtt{e} \in \mathbb{E}$ is denoted as a pair (x,u) representing the action of assigning the execution of a task to a user u at assignment step x. Assignment event (x,u) has the probability of success $\theta(x,u) = p$, meaning a task will be executed at step x with probability p. If $\theta(x,u) = 0$ then a task will not be executed at step x, and the assignment event (x,u) is said to have failed. Each termination action $\mathtt{t} \in \mathbb{T}$, represents the action of terminating a

workflow if the current step is the termination step $x_\perp$, or if the process moves to an invalid state s which we discuss in the next section.

Workflow Process Transitions

Given a workflow execution specification $WES = (((T,<),U,(A,S,B)),((Z,\prec),\theta))$, we now discuss how an MDP_W moves from state to state by way of transitions, and show how the transition function $\mathbf{p}$ defined in Section 3.1.2 prevents redundant transitions and states in an MDP_W. First, we show how the finite set of assignment decisions which can be executed in a state s is defined by $\mathbf{p}$. An assignment decision (t',u') is executed in any state $s = (x_i,(t,u),\pi)$, such that $\pi : R \to U$ is a plan where $R \subseteq T$, if the following two properties hold:

$$x_i \neq x_\perp \wedge \pi \text{ is valid} \wedge |R| = i-1 \wedge (t \in R \vee R = \emptyset) \tag{3.1}$$

$$\forall t'' \in T, t'' < t', \exists u \in U \Rightarrow \pi(t'') = u \tag{3.2}$$

State $s = (x_i,(t,u),\pi)$ satisfies Property 3.1 if the termination step $x_\perp$ has not been reached, the plan π is valid, the number of tasks whose executions have been assigned by π equals the number of completed steps, and either π assigns the execution of task t to a user, or π does not assign the execution of any tasks. State $s = (x_i,(t,u),\pi)$ satisfies Property 3.2 if the execution of task t at step x_i respects the ordering defined by the task schema $(T,<)$ when considering all tasks selected at previous workflow steps. If Properties 3.1 and 3.2 both hold, then the MDP_W will move to a new state $s' = (x_i,(t',u'),\pi)$ with the probability $\mathbf{p}(s,(t',u'),s') = 1$. If both properties do not hold for executing assignment decision (t',u') in state $s = (x_i,(t,u),\pi)$ then $\mathbf{p}(s,(t',u'),s') = 0$.

Next, we show how the finite set of assignment events which can be executed in a state s is defined by $\mathbf{p}$ to be a singleton set. An assignment event (x_j,u_q) is executed in any state $s = (x_i,(t,u_p),\pi)$, such that $\pi : R \to U$ is a plan where $R \subseteq T$, if the following two properties hold:

$$x_i \neq x_\perp \wedge |R| = i-1 \wedge t \notin R \tag{3.3}$$

$$x_j = x_i \wedge u_q = u_p \tag{3.4}$$

State $s = (x_i,(t,u),\pi)$ satisfies Property 3.3 if the termination step $x_\perp$ has not been reached, the number of tasks whose executions have been assigned by π equals the number of completed steps, and π does not assign the execution of task t to a user. State $s =$

$(x_i,(t,u),\pi)$ satisfies Property 3.4 if x_j equals the current execution step x_i and u_q is the user u_p selected at step x_i to be assigned the execution of task t. If Properties 3.3 and 3.4 both hold then the MDP$_W$ will move to either a new state $s' = (x_{i+1},(t,u),\pi(t) \leftarrow u)$ with the probability $\mathbf{p}(s,(x_i,u),s') = \theta(x_i,u)$, or a new state $s'' = (x_{i+1},(t,u),\pi)$ with the probability $\mathbf{p}(s,(x_i,u),s') = 1-\theta(x_i,u)$. In the former case, state s' indicates the completion of step x_i and creates a new plan by assigning the execution of t to u, denoted as $\pi(t) \leftarrow u$. Note, given the set of all execution steps $X = Z \setminus \{x_\perp\}$, if $i = |X|$ then $x_{i+1} = x_\perp$. In the latter case, state s'' also indicates the completion of step x_i but does not create a new plan as u was not available to be assigned the execution of t. If Properties 3.3 and 3.4 both do not hold in $s = (x_i,(t,u),\pi)$ then $\mathbf{p}(s,(x_i,u),s') = 0$.

A termination action $\mathtt{t}$ is executed in state $s = (x_i,(t,u),\pi)$, such that $\pi : R \rightarrow U$ is a plan where $R \subseteq T$, if one of the following two properties hold:

$$\pi \text{ is valid} \wedge R = T \tag{3.5}$$

$$(x_i \neq x_\perp \wedge \pi \text{ is valid} \wedge |R| < i-1) \vee (x_i = x_\perp \wedge R \neq T) \vee (x_i \neq x_1 \wedge \pi \text{ is invalid}) \tag{3.6}$$

State $s = (x_i,(t,u),\pi)$ satisfies Property 3.5 if the plan π is valid, and the execution of all tasks have been assigned to a user by π, that is π is complete. The MDP$_W$ is said to end successfully in this case. State $s = (x_i,(t,u),\pi)$ satisfies Property 3.6 if either the termination step $x_\perp$ has not been reached and the number of tasks whose executions have been assigned by π is less than the number of completed steps, or the termination step $x_\perp$ has been reached and the execution of all tasks have not been assigned to a user by π, or step x_i is not the first workflow step and π is invalid. The MDP$_W$ is said to end unsuccessfully in this case. If Property 3.5, or Property 3.6 hold in $s = (x_i,(t,u),\pi)$ then the MDP$_W$ will move to a termination state $s' = (x_i, null, \pi(t))$ with the probability $\mathbf{p}(s,\mathtt{t},s') = 1$. If neither property holds in state s then $\mathbf{p}(s,\mathtt{t},s') = 0$ meaning the MDP$_W$ will not terminate in s.

Workflow Decision Policy

In Section 3.1.2 a decision policy was defined as a function $\delta : \mathbb{S} \rightarrow \mathbb{A}$ which given a process state, returned the action to execute in that state. In an MDP$_W$ a *workflow decision policy* δ, given a state $s = (x_i,(t,u),\pi)$, returns a single assignment decision (t,u), assignment event (x,u), or termination action $\mathtt{t}$ depending on the properties of s. Given a workflow execution specification $WES = (WS,ES)$, many workflow decision policies δ may exist, each one resolving all non-deterministic choice to define a single course of sequential actions among several possible courses of sequential actions. In Section 2.2.1 we defined Π to be the set of

Table 3.1 Assignment decisions returned at each execution step x_i by all workflow decision policies δ that enable the workflow specification WS_1 to be satisfied.

Plan	Execution Schedule	Decision Policy	Execution Steps x_1	x_2	x_3	x_4	x_5
π_{11}	σ_{11}	δ_1	(t_1,u_1)	(t_2,u_2)	(t_3,u_1)	(t_4,u_4)	(t_5,u_1)
	σ_{12}	δ_2	(t_1,u_1)	(t_2,u_2)	(t_4,u_4)	(t_3,u_1)	(t_5,u_1)
	σ_{13}	δ_3	(t_1,u_1)	(t_4,u_4)	(t_2,u_2)	(t_3,u_1)	(t_5,u_1)
π_{12}	σ_{11}	δ_4	(t_1,u_1)	(t_2,u_3)	(t_3,u_1)	(t_4,u_2)	(t_5,u_1)
	σ_{12}	δ_5	(t_1,u_1)	(t_2,u_3)	(t_4,u_2)	(t_3,u_1)	(t_5,u_1)
	σ_{13}	δ_6	(t_1,u_1)	(t_4,u_2)	(t_2,u_3)	(t_3,u_1)	(t_5,u_1)
π_{13}	σ_{11}	δ_7	(t_1,u_1)	(t_2,u_3)	(t_3,u_1)	(t_4,u_2)	(t_5,u_4)
	σ_{12}	δ_8	(t_1,u_1)	(t_2,u_3)	(t_4,u_2)	(t_3,u_1)	(t_5,u_4)
	σ_{13}	δ_9	(t_1,u_1)	(t_4,u_2)	(t_2,u_3)	(t_3,u_1)	(t_5,u_4)
π_{14}	σ_{11}	δ_{10}	(t_1,u_1)	(t_2,u_3)	(t_3,u_1)	(t_4,u_4)	(t_5,u_1)
	σ_{12}	δ_{11}	(t_1,u_1)	(t_2,u_3)	(t_4,u_4)	(t_3,u_1)	(t_5,u_1)
	σ_{13}	δ_{12}	(t_1,u_1)	(t_4,u_4)	(t_2,u_3)	(t_3,u_1)	(t_5,u_1)
π_{15}	σ_{11}	δ_{13}	(t_1,u_2)	(t_2,u_3)	(t_3,u_2)	(t_4,u_4)	(t_5,u_1)
	σ_{12}	δ_{14}	(t_1,u_2)	(t_2,u_3)	(t_4,u_4)	(t_3,u_2)	(t_5,u_1)
	σ_{13}	δ_{15}	(t_1,u_2)	(t_4,u_4)	(t_2,u_3)	(t_3,u_2)	(t_5,u_1)

all possible plans for WES and $\Pi_V \subseteq \Pi$ for the set of all complete and valid plans for WES. In Section 2.3.2 we defined the set of all complete and valid step-task mappings to be M, where $|M|$ equals the number of execution schedules σ. The number of existing decision policies which enable the satisfiability of WS, is $|\Pi_V| \times |M|$.

Under a full availability forecast θ (Section 2.3.1, Property 2.9) any one of these decision policies could be implemented for an automated workflow management system to enable satisfiability when executing WES. In cases where users may not always be available, the optimal strategy will be one of these workflow decision policies. Solving an MDP_W allows us to find the optimal strategy and generate the workflow metrics we require when using appropriate reward functions which we discuss in the next section. As a final note, three execution schedules were shown to exist in Section 2.1.1 for workflow specification WS_1, such that $|M_1| = 3$. In Section 2.2.1 five complete and valid plans were shown to exist for WS_1, such that $|\Pi_{V1}| = 5$. It follows that 3 * 5 = 15 workflow decision policies exist which enable the satisfiability of WS_1. Table 3.1 shows the assignment decisions returned at each execution step x_i by all 15 of these workflow decision policies δ.

3.2.2 Workflow Metric Reward Functions

In this section we define two reward functions for an MDP$_W$. The first, $\mathbf{r}_Q$, is used for generating either the quantitative satisfiability or quantitative resiliency of a workflow when solving an MDP$_W$. The second, $\mathbf{r}_D$, is used for generating the distance resiliency of a workflow when solving an MDP$_W$.

Quantitative Resiliency and Satisfiability

We now define an MDP$_W$ reward function $\mathbf{r}_Q(s,a,s')$ for both quantitative resiliency and satisfiability metrics. Both cases are concerned with the process reaching a successful termination state $s' = (x_\perp, null, \pi)$ where the plan π is complete and valid, therefore a single reward function is sufficient. As the maximum quantitative satisfiability or resiliency for any workflow execution specification WES is 1, we associate a reward of 1 to any transition that moves the process from a non-termination state $s = (x_i, a, \pi)$ to a successful termination state s' iff π is a complete and valid plan, following the execution of a termination action $\mathtt{t}$ in s. Given $\pi : R \to U$ is a plan, such that $R \subseteq T$, we give the reward function $\mathbf{r}_Q$ to be:

$$\mathbf{r}_Q((x_i,a,\pi),a',(x_i,a'',\pi)) = \begin{cases} 1 & \text{if } \pi \text{ is valid} \wedge R = T \wedge a' \in \mathbb{T} \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

Given a valid execution workflow schema $WES = (WS, ((Z, \prec), \theta))$, an MDP$_W$ for computing the quantitative resiliency and satisfiability of WES is written as $(\mathbb{S}, \mathbb{A}, \mathbf{p}, \mathbf{r}_Q)$. In the case of computing the quantitative resiliency of WES, the workflow metric function $\Gamma_Q = V^*(s_0)$, where $s_0 \in \mathbb{S}$ is the initial state and V^* is the optimal value function of an MDP$_W$ $(\mathbb{S}, \mathbb{A}, \mathbf{p}, \mathbf{r}_Q)$. For computing the quantitative satisfiability of WES, the availability forecast θ must be full (Section 2.3.1, Property 2.9). In this case the workflow metric function $\Gamma_S = V^*(s_0)$.

Distance Resiliency

We now define an MDP$_W$ reward function $\mathbf{r}_D(s,a,s')$ for a distance resiliency metric. Distance resiliency is concerned with the process completing execution steps. The execution of an assignment event moves the process to a state s and completes an execution step x. As the maximum probability for any assignment event being successful is 1 we associate a reward of 1 to any transition that moves the process from state $s = (x_i, a, \pi)$ to a new state s', following the execution of an assignment decision $\mathtt{d}$ in s, or iff π is a complete and valid

Fig. 3.6 Example workflow task schema $(T_2, <_2)$ and security policy (A_2, S_2, B_2) used to illustrate solving a workflow Markov decision process (MDP_W).

plan, a termination action `t` in state s. Note, because each reward of 1 is received after the completion of an execution step, no reward is received if x_i is the first step x_1. Given that $\pi : R \to U$ is a plan, such that $R \subseteq T$, we give the reward function $\mathbf{r}_D$ to be:

$$\mathbf{r}_D((x_i, a, \pi), a', (x_i, a'', \pi)) = \begin{cases} 1 & \text{if } \pi \text{ is valid} \wedge R = T \wedge a' \in \mathbb{T} \\ 1 & \text{if } i \neq 1 \wedge a' \in \mathbb{D} \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

Given a valid execution workflow schema $WES = (WS, ((Z, \prec), \theta))$, an MDP_W for computing the distance resiliency of WES is written as $(\mathbb{S}, \mathbb{A}, \mathbf{p}, \mathbf{r}_D)$. For computing the distance resiliency of WES, the workflow metric function $\Gamma_D = V^*(s_0)$, where V^* is the optimal value function of an MDP_W $(\mathbb{S}, \mathbb{A}, \mathbf{p}, \mathbf{r}_D)$.

3.2.3 Solving Workflow Markov Decision Process

We now illustrate by example how quantitative satisfiability, quantitative resiliency and distance resiliency is generated by solving an MDP_W.

Example Execution Workflow Specification

First, we consider a small valid workflow execution specification $WES_2 = (WS_2, ES_2)$, such that the workflow specification $WS_2 = ((T_2, <_2), U_2, (A_2, S_2, B_2))$ consists of the following:

- A task schema $(T_2, <_2)$ where $T_2 = \{t_1, t_2\}$ and $t_1 < t_2$
- A set of users $U_2 = \{u_1, u_2\}$
- A security policy (A_2, S_2, B_2) where $A_2 = \{(t_1, u_2), (t_2, u_1), (t_2, u_2)\}$, $S_2 = \{(t_1, t_2)\}$, and $B_2 = \emptyset$

Task schema $(T_2, <_2)$ and security policy (A_2, S_2, B_2) are illustrated in Figure 3.6. The execution specification $ES_2 = ((Z_2, \prec_2), \theta_2)$, which is compatible with WS_2, consists of the following components:

- An execution schema $(Z_2, \prec_2)$ where $Z_2 = \{x_1, x_2, x_\perp\}$ and $x_1 < x_2 < x_\perp$

- An availability forecast $\theta_2 : (Z_2 \setminus x_\perp) \times U_2 \to \mathbb{R}$

With this small example it is quite straightforward to calculate the set Π_2 of all plans for WES_2 to be:

	t_1	t_2
π_{21}	u_1	u_1
π_{22}	u_1	u_2

	t_1	t_2
π_{23}	u_2	u_1
π_{24}	u_2	u_2

It is also straightforward to see the set of complete and valid plans $\Pi_{V2} \subseteq \Pi_2$ to be $\Pi_{V2} = \{\pi_{23}\}$. A single step-task mapping μ_2 exists where $\mu_2(x_1) = t_1$ and $\mu_2(x_2) = t_2$.

Quantitative Satisfiability

In order to generate quantitative satisfiability we must consider the availability forecast θ_2 for WES_2 to be full, and we write $WES_{21} = (WS_2, ((Z_2, \prec_2), \theta_{21}))$ for this. Therefore for all execution steps $x \in Z_2 \setminus \{x_\perp\}$ and every user $u \in U_2$, the probability of u being available at x is $\theta_{21}(x,u) = 1$. Definition 13 states that if the set of all complete and valid plans $\Pi_V \neq 0$ for a workflow execution specification WES, then the quantitative satisfiability $\Gamma_S(WES) = 1$. It follows that because $\Pi_{V2} \neq \emptyset$ the quantitative satisfiability of WES_{21} is $\Gamma_S(WES_{21}) = 1$.

Figure 3.7 shows how the quantitative satisfiability is generated for the workflow execution specification $WES_{21} = (WS_2, ((Z_2, \prec_2), \theta_{21}))$ using an MDP$_W$ $(\mathbb{S}, \mathbb{A}, \mathbf{p}, \mathbf{r}_Q)$. State s_0 is the initial process state. States s_0 and s_4 are states where a non-deterministic choice exists which must be resolved between executing assignment decisions. States s_1, s_2, s_6 and s_7 are states where probabilistic assignment events are executed, however because θ_{21} is full, the probability of each assignment event being successful is 1. States s_5, s_{10}, and s_{11} are termination states, with s_{10} being the only successful termination state. The transition function $\mathbf{r}_Q$ associates a reward of 1 to the transition from state s_8 to state s_{10}, such that $\mathbf{r}_Q(s_8, \mathtt{t}_2, s_{10}) = 1$. The successful termination property (Section 3.2.1, Property 3.5) is satisfied in this case by state s_8. All transitions to any other termination state are associated with a reward of 0 due to the unsuccessful termination property (Section 3.2.1, Property 3.6).

Figure 3.7 shows the maximum expected rewards at states s_0 and s_4 where a non-deterministic choice exists. Starting at the termination states s_{10} and s_{11}, all transition rewards are collected up in an inductive manner to give the maximum expected reward at state s_4 to be $V^*(s_4) = 1$, where V^* is the optimal value function of the MDP$_W$. Similarly, collecting all transition rewards starting from the termination state s_5 and those already collected at s_4, gives the maximum expected reward at the initial state s_0 to be $V^*(s_0) = 1$. As the quantitative satisfiability metric $\Gamma_S = V^*(s_0)$, it follows the quantitative satisfiability of WES_{21} is $\Gamma_S(WES_{21}) = 1$.

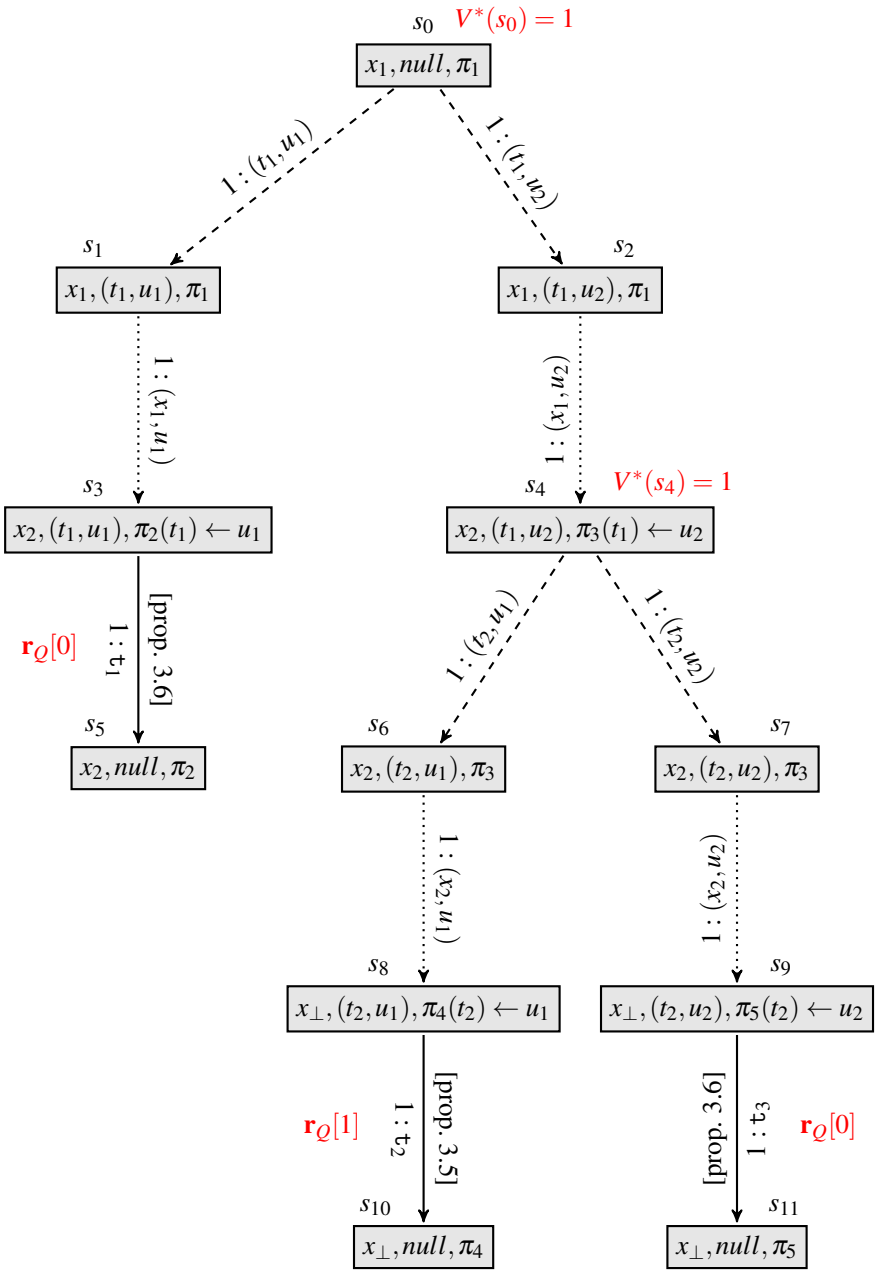


Fig. 3.7 Quantitative satisfiability calculation for workflow execution specification WES_{21} using an MDP_W, where $\mathbf{r}_Q$ is the reward received in a termination state s and $V^*(s)$ is the maximum expected reward in state s where non-deterministic choice exists. State s_{10} is the only successful termination state.

Quantitative Resiliency

In order to generate quantitative resiliency for WES_2, we consider the availability forecast θ_{22} to be:

	u_1	u_2
x_1	0.8	0.6
x_2	0.8	0.7

We then write $WES_{22} = (WS_2, ((Z_2, \prec_2), \theta_{22}))$ to denote this workflow execution specification. Definition 12 states quantitative resiliency is the maximum feasibility found for all plans $\pi \in \Pi_V$ where Π_V is the set of all complete and valid plans for a workflow execution specification. For example WES_{22}, the set of all complete and valid plans is a singleton set $\Pi_{V2} = \{\pi_{23}\}$. The quantitative resiliency of WES_{22} is therefore the feasibility of π_{23} given by the function ρ. As only a single step-task mapping μ_2 exists, the feasibility of π_{23} is given to be:

$$\begin{aligned}
\rho(\pi_{23}) &= \theta(x_1, \pi_{23}(\mu_2(x_1)))\theta(x_2, \pi_{23}(\mu_2(x_2))) \\
\rho(\pi_{23}) &= \theta(x_1, \pi_{23}(t_1))\theta(x_2, \pi_{23}(t_2)) \\
\rho(\pi_{23}) &= \theta(x_1, u_2)\theta(x_2, u_1) \\
\rho(\pi_{23}) &= 0.6 * 0.8 \\
\rho(\pi_{23}) &= 0.48
\end{aligned}$$

Therefore the quantitative resiliency of WES_{22} is $\Gamma_Q(WES_{22}) = 0.48$. Figure 3.8 shows how the quantitative resiliency is generated for the workflow execution specification $WES_{22} = (WS_2, ((Z_2, \prec_2), \theta_{22}))$ using an MDP_W $(\mathbb{S}, \mathbb{A}, \mathbf{p}, \mathbf{r}_Q)$. State s_0 is the initial process state. States s_0 and s_6 are states where a non-deterministic choice exists which must be resolved between executing assignment decisions. States s_1, s_2, s_9 and s_{10} are states where probabilistic assignment events are executed. Because θ_{21} is probabilistic, the probability of each assignment event being successful is dependant on the availability of users defined by θ_{21}. For instance, the process transitions from state s_1 to state s_3 with a probability of 0.8 as the probabilistic availability of user u_1 at execution step x_1 is 0.8. Otherwise the process transitions from state s_1 to state s_4 with a probability of 0.2. States s_7, s_8, s_{15}, s_{16}, and s_{17} are termination states, with s_{15} being the only successful termination state. The transition function $\mathbf{r}_Q$ associates a reward of 1 to the transition from state s_{11} to state s_{15}, such that

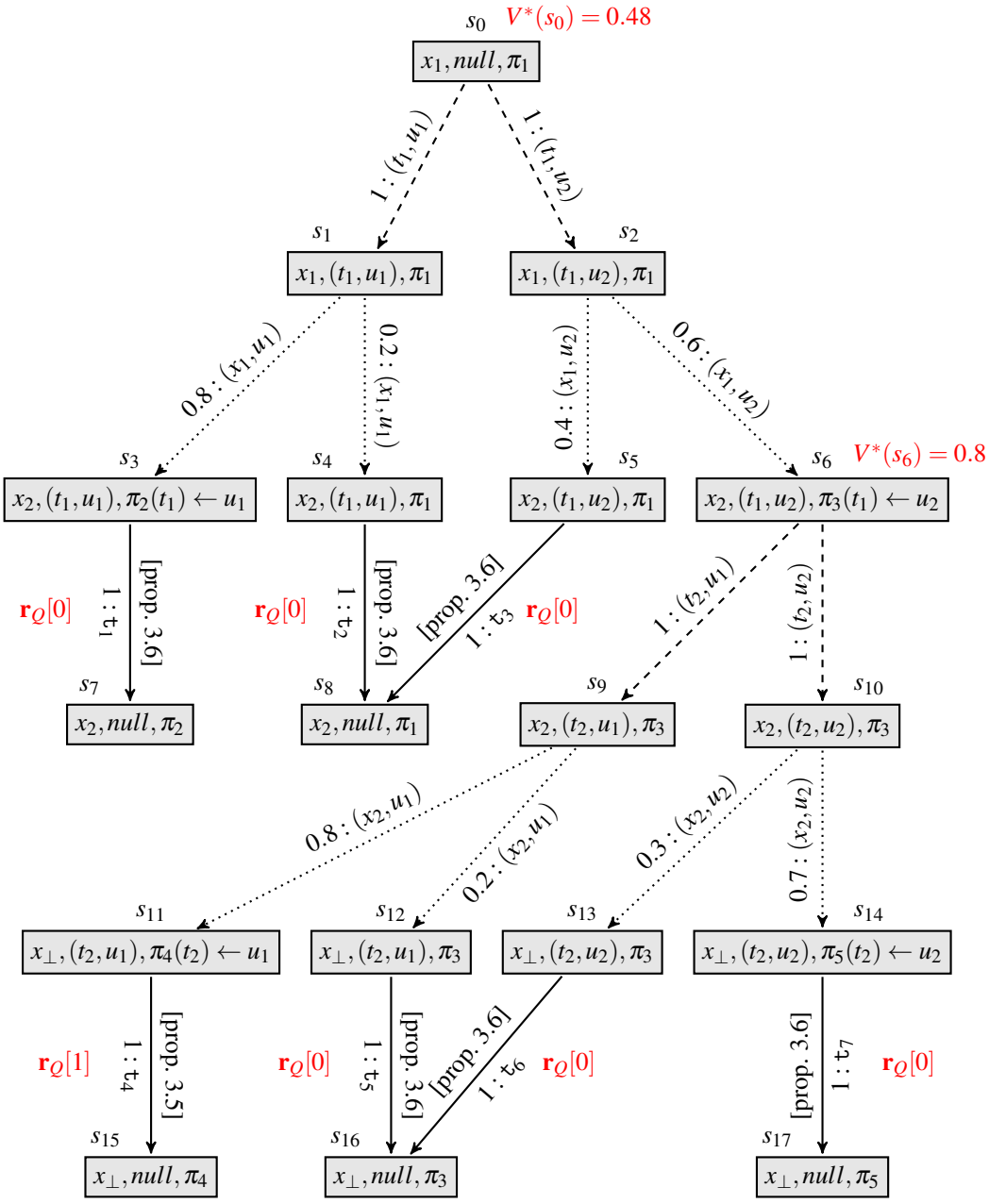


Fig. 3.8 Quantitative resiliency calculation for workflow execution specification WES_{22} using an MDP_W, where $\mathbf{r}_Q$ is the reward received in a termination state s and $V^*(s)$ is the maximum expected reward in state s where non-deterministic choice exists. State s_{15} is the only successful termination state.

$\mathbf{r}_Q(s_{11}, \mathtt{t}_4, s_{15}) = 1$. The successful termination property (Section 3.2.1, Property 3.5) is satisfied in this case by state s_{11}. All transitions to any other termination state are associated with a reward of 0 due to the unsuccessful termination property (Section 3.2.1, Property 3.6).

Figure 3.9 shows the maximum expected rewards at states s_0 and s_6 where a non-deterministic choice exists. Starting at the termination states s_{15}, s_{16}, and s_{17}, all transition rewards are collected up in an inductive manner to give the maximum expected reward at state s_6 to be $V^*(s_6) = 0.8$, where V^* is the optimal value function of the MDP_W. Similarly, collecting all transition rewards starting from the termination states s_7 and s_8, and those already collected at s_6, gives the maximum expected reward at the initial state s_0 to be $V^*(s_0) = 0.48$. As the quantitative resiliency metric $\Gamma_Q = V^*(s_0)$, it follows the quantitative resiliency of WES_{22} is $\Gamma_Q(WES_{22}) = 0.48$.

Distance Resiliency

We now consider generating distance resiliency for the workflow execution specification $WES_{22} = (WS_2, ((Z_2, \prec_2), \theta_{22}))$ defined in the previous section. Definition 14 states distance resiliency is the maximum expected number of steps completed found for all plans $\pi \in \Pi_V$ where Π_V is the set of all complete and valid plans for a workflow execution specification. For the example WES_{22}, the set of all complete and valid plans is a singleton set $\Pi_{V2} = \{\pi_{23}\}$. The distance resiliency of WES_{22} is therefore the expected number of steps completed by π_{23}, given by the function λ. As only a single step-task mapping μ_2 exists, the expected number of steps completed by π_{23} is given to be:

$$\begin{aligned}
\lambda(\mu_2, \pi_{23}, (Z_2 \setminus \{\perp_x\}), \theta_{22}) &= \theta(x_1, \pi_{23}(\mu_2(x_1))) + \theta(x_1, \pi_{23}(\mu_2(x_1)))\theta(x_2, \pi_{23}(\mu_2(x_2))) \\
&= \theta(x_1, \pi_{23}(t_1)) + \theta(x_1, \pi_{23}(t_1))\theta(x_2, \pi_{23}(t_2)) \\
&= \theta(x_1, u_2) + \theta(x_1, u_2)\theta(x_2, u_1) \\
&= 0.6 + 0.6 * 0.8 \\
&= 1.08
\end{aligned}$$

Therefore the distance resiliency of WES_{22} is $\Gamma_D(WES_{22}) = 1.08$. Figure 3.9 shows how the distance resiliency is generated for the workflow execution specification $WES_2 = (WS_2, ((Z_2, \prec_2), \theta_{22}))$ using an MDP_W $(\mathbb{S}, \mathbb{A}, \mathbf{p}, \mathbf{r}_D)$. All states are the same as for generating the quantitative resiliency for WES_2 in the previous section. The transition function $\mathbf{r}_D$ associates a reward of 1 to all transitions from a successful step completion state. In this case, only states s_6 and s_{11} are such states as step x_1 and x_2 are completed successfully in s_6

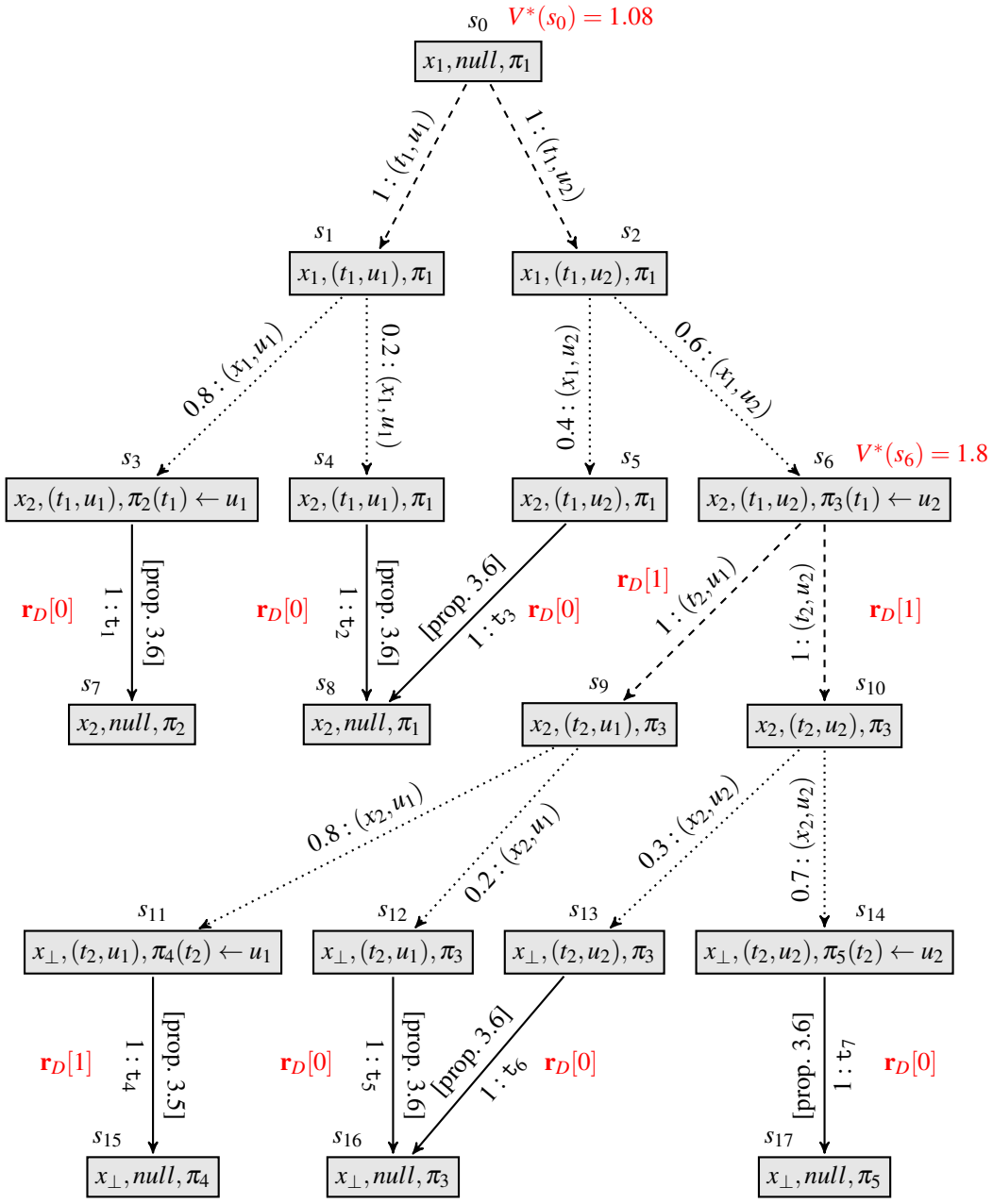


Fig. 3.9 Distance resiliency calculation for workflow execution specification WES_{22} using an MDP_W, where $\mathbf{r}_D$ is the reward received in a termination state s and $V^*(s)$ is the maximum expected reward in state s where non-deterministic choice exists. State s_{15} is the only successful termination state.

and s_{11} respectively. For instance, the transitions from state s_6 to state s_9 and state s_{10} are associated a reward of 1, such that $\mathbf{r}_D(s_6,(t_2,u_1),s_9) = 1$ and $\mathbf{r}_D(s_6,(t_2,u_2),s_{10}) = 1$. Both Property 3.1 and Property 3.2 defined in Section 3.2.1 hold in this case for state s_6.

Figure 3.9 shows the maximum expected rewards at states s_0 and s_6 where a non-deterministic choice exists. Starting at the termination states s_{15}, s_{16}, and s_{17}, all transition rewards are collected up in an inductive manner to give the maximum expected reward at state s_6 to be $V^*(s_6) = 1.8$, where V^* is the optimal value function of the MDP_W. Similarly, collecting all transition rewards starting from the termination states s_7 and s_8, and those already collected at s_6, gives the maximum expected reward at the initial state s_0 to be $V^*(s_0) = 1.08$. As the distance resiliency metric $\Gamma_D = V^*(s_0)$, it follows the distance resiliency of WES_{22} is $\Gamma_Q(WES_{22}) = 1.08$.

3.3 Related Work

To support our approach a number of works appearing in the literature have used Markov decision processes (MDPs) for optimised decision making within the workflow and security domains. In [51], Doshi et. al. consider the automatic composition of Web services to form flexible and efficient workflows. They argue that current planning algorithms do not consider the uncertain behaviour of real-world Web services and the dynamic environments in which they operate. The authors consider the generation of robust and adaptive workflows as a decision problem and model the abstracted composition process as an MDP. Solving the MDP provides a *policy* optimally balancing the rewards of completing the workflow with the risks imposed by the Web services and their operational environment. Periodical Bayesian model learning is suggested as a way to update knowledge about the environment in order to gradually improve the quality of those workflows composed in the future.

A very similar approach to dynamic Web service composition is defined by Gao et. al in [68] who use MDPs to optimise overall quality of service (QoS). This work considers more complex compositions to [51] such as parallel and conditional execution. It is assumed in this work that Web service QoS data can be obtained from service registries. Two algorithms to solve the MDP based on value iteration are also presented. In both of these works, each Web service is selected in order to optimise workflow factors such as cost, time and availability making the problem similar in nature to the one of optimally assigning workflows which is the problem we consider. However they do not consider having to satisfy a security policy when selecting services, nor how certain Web service selections may prevent future sections of a workflow from being composed (i.e., the WSP). It is therefore assumed a workflow can

always be composed (and completed) making them insufficient for the analysis of security policy impact on workflow resiliency.

The assignment of workflows using MDPs is also adopted by Espinosa et. al. in [58] such that selecting a human to perform task is based on assessing their behavioural impact on the overall financial revenue generated by the workflow. Behavioural impact is gained from using a balanced scorecard system which records a human's performance profile in terms of specific skills, experience, salary etc. This system allows a comparison to be performed between different employees in terms of their economical benefit/cost. For example a choice may exist between assigning a current employee or hiring one or more agency staff. The more expensive option may complete the task in a quicker time thus offering more gain in the long term. The motivation for this approach is to integrate it into the common workflow description language Business Process Model and Notation (BPMN) [136] which currently lacks the capability to model workflow goals and strategies. Again, the satisfaction of a security policy is not considered when selecting employees for task assignments.

In terms of security, quantitative access control using partially-observable MDPs is presented by Martinelli et al. in [122] which under uncertainty, aims to optimise the decision process for a sequence of probabilistic access requests. An access request may be denied if by solving the MDP it is predicted that one or more access requests may be denied later on in the sequence (e.g. due to a separation of duty). Access requests are not however assessed within the confines of a structured workflow (i.e., correct ordering) nor does this work consider the WSP and resiliency problems *per se*. It does however consider the satisfaction of a security policy and provides inspiration for the solution defined in this chapter for quantifying the resiliency of a workflow. For instance, probabilistic access requests can be viewed as being loosely equivalent to the probabilistic availability of users to perform workflow tasks.

3.4 Summary

In general, metrics are standards of measurement, commonly used by organisations, by which efficiency, performance, progress, or quality of a plan, process, or product can be assessed. In Chapter 2 we defined an execution workflow specification which consists of a workflow specification and execution specification. The former describes the workflow itself in terms of tasks, users, and security policy; the latter exposes the notion of user availability and defines a forecast of predicted availability of users during the execution of a workflow. We also defined the satisfiability and resiliency of a workflow before proposing quantitative measures for these concepts in the form of *workflow metrics*. We suggest a CISO can use

these metrics to assess the satisfiability and resiliency of a workflow, and undertake fine grained impact analysis of modifications to a workflow's security policy by generating a metric with and without the modification, and observing the degree of change.

In this chapter we considered the execution of a workflow to be a stepwise decision making problem consisting of discrete steps where an assignment decision and event is actioned. Assignment decisions represent the selection of a task and a user, whilst assignment events represent the assignment of a task's execution to a user. The choice of which assignment decisions to execute are non-deterministic and must be resolved by a workflow management system. Event success is probabilistic as the availability of users is considered to be probabilistic. We showed how the execution of a workflow under these conditions is partly controlled by the workflow management system resolving assignment decisions, and partly random due to the uncertainty introduced by the probabilistic availability of users.

We described the general concept of decision making processes before introducing a Markov decision process (MDP) which is an analytical tool for modelling and studying stepwise decision problems. Solving an MDP exposes the optimal decision policy which resolves all non-deterministic choices by defining an optimal sequence of process actions that maximise the chance of reaching some process termination state. We showed how a workflow execution process can be modelled as an MDP, which we call a workflow Markov decision process (MDP_W). We then define two reward functions, one for generating quantitative satisfiability and resiliency of a workflow, the other for generating distance resiliency. Reward functions place motivational rewards on specific process transitions, and solving the MDP_W consists of finding the expected maximum reward in the initial state of the process for a given reward function. Using this single MDP_W framework we show how the satisfiability and resiliency metrics can be generated by changing the reward function. In Chapter 4 we show how these metrics can be computer generated by encoding the MDP_W in the probabilistic model checker PRISM, and how the impact of a security policy on the completion of a workflow can be analysed.

Chapter 4

Computer Generated Metrics

Generating workflow metrics manually can soon become complex, time consuming, and error prone even for small scale workflows like the examples used in this thesis to illustrate our approach. Applying mathematical-based techniques directly may not be practical nor fully understood by a Chief Information Security Officer (CISO) when generating metrics for analysing the impact security policies have on workflow completion. A logical next step is to provide a more accessible and efficient way of generating metrics by using an automated tool that supports our approach. In Chapter 2 we gave a formal definition of a security constrained workflow and described it to be satisfiable if the execution of its tasks could be assigned to users without violating security constraints. We then extended the definition to expose the notion of users being unavailable during workflow execution and described a workflow as being resilient if it could be completed with unavailable users while still satisfying all security constraints. In Chapter 3 we modelled the process of assigning the execution of tasks to users under uncertainty as a Markov decision process (MDP), and described how solving a workflow Markov decision process (MDP_W) could generate quantitative measures of workflow satisfiability and resiliency.

In this chapter we focus on the concept of probabilistic model checking and introduce the probabilistic model checking tool PRISM, which is suitable for modelling and solving Markov decision processes. We give a systematic encoding of an MDP_W in PRISM using its high level modelling language before describing how PRISM is used to verify properties in the encoded process model and generate the workflow metrics we require. Using PRISM we generate metrics for an example workflow execution specification, and show how a CISO can use these metrics to analyse the potential impact different security policy modifications have on the workflow's completion. We also give an indication of the main computational overheads that are encountered when using PRISM for generating workflow metrics.

4.1 PRISM

In this section we introduce the probabilistic model checking tool PRISM [104]. Using its high level, state based modelling language we encode an MDP_W defined in Chapter 3 and use PRISM to automatically generate the quantitative workflow metrics we require. We first give a brief overview of probabilistic model checking before describing the PRISM model checking tool, its modelling language, and its building and verification of process models.

4.1.1 Probabilistic Model Checking

To rule out errors in a system one needs to try all possible executions of that system, which is often not feasible in practice. Instead, a common technique is to perform some method of formal verification. Formal verification is the application of rigorous, mathematical-based techniques to verify the existence of properties in a system and prove that a system satisfies its specification. Formal verification can take many forms, for instance, manual proof, automated theorem proving, static analysis, and model checking. Model checking is a common technique for automatically verifying the existence of properties in finite-state systems, modelled for example, as Markov decision processes [104]. Model checking has a number of advantages over traditional approaches that are based on simulation, testing, and deductive reasoning. In particular, model checking is automatic and can be quite efficient. Also, if the system design contains an error, model checking will produce a counterexample that can be used to pinpoint the source of the error.

Once a system property whose existence needs verifying is specified, an automatic model checking tool can be used to exhaustively check whether that property holds in the system. For instance, one may want to verify the absence of deadlocks in critical systems, or, in the case of the workflow satisfiability problem, verifying that a workflow can be satisfied. Many properties other than the correctness of a system may be important. For instance a system designer may need to guarantee the safety, reliability, performance, dependability, resource usage, and security of a system. Many model checking tools are in existence, many of which have been used successfully in practice to verify real industrial designs[1].

Some systems are inherently probabilistic, such as the workflows we consider in the first part of this thesis. Probabilistic model checking is a formal verification technique for systems which contain uncertainty, for instance, the probabilistic availability of resources such as users, needed for a system to complete successfully. Analysis of these systems can quantify

[1] http://www.prismmodelchecker.org/other-tools.php

the existence of properties such as levels of resource usage, quality of service, rates of failure, satisfiability and resiliency.

4.1.2 Model Checker

PRISM is a free, open-source probabilistic model checking tool, which enables the formal modelling and analysis of models that exhibit probabilistic behaviour [104]. PRISM has been used to analyse systems in many different domains, including communication and security protocols, biological systems and many others including workflow, some of which are discussed in Section 4.4.2. PRISM has also been heavily used in the field of systems and security research, and features in over 500 peer reviewed publications[2]. PRISM becomes an intuitive choice of tool for our needs as it can model both probabilistic and non-deterministic choice, and gives an efficient way to automatically solve a Markov decision process whilst providing analysis data regarding computation overheads.

PRISM can build and analyse many types of probabilistic model, including discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs), probabilistic automata (PAs), probabilistic timed automata (PTAs), and Markov decision processes which we use in Chapter 3 to model the execution process of a security constrained workflow coming with probabilistic user availability. The PRISM developers, who are active researchers at Oxford University at the time of writing, give no definitive answer to the size of models in terms of state space, the PRISM model checking tool can handle. Several factors can affect the performance of PRISM including its use of data structures based on binary decision diagrams (BDDs) which make performance unpredictable [131]. Other factors include the model type and model properties being verified, and the engine used for verification. The default PRISM engine is the *hybrid* engine which can reportedly handle models of up to 10^8 states on a 'typical' personal computer [103]. To put this into perspective, the example MDP_W processes we have introduced in Chapters 2 and 3 have between 12 and 719 states. PRISM engines are discussed in Section 4.1.4.

Probabilistic models can be encoded in PRISM using the PRISM language which is a high-level, state-based modelling language. PRISM provides support for automatic analysis with a wide range of quantitative properties which can be specified for such models, for example, the maximum probability of reaching a failure state, the probability of terminating successfully, or the expected maximum reward for reaching any one of a number of states holding a particular property. PRISM provides a number of techniques for solving probabilistic models, such

[2] http://www.prismmodelchecker.org/bib.php

as Markov decision processes, including dynamic programming (e.g. value iteration) [83]. Value iteration begins at the 'end' states of a process and then works backward to determine a sequence of optimal actions, and refines an estimate of the optimal value function V^*. Roughly speaking, PRISM proceeds by first considering the last time a non-deterministic decision might be made and trying all available decisions at that time. Using this information, PRISM can determine what to do at the second-to-last time a non-deterministic decision might be made. This process continues backwards until PRISM has determined the best action for every possible situation at every point in time.

4.1.3 Modelling Language

A PRISM encoded process model constitutes a number of interactive modules that contain one or more local *variables* and *commands* as follows:

> **module** *name*
> *variables*
> *commands*
> **endmodule**

The values of a module's local variables define the module's state, while the values of every variable across all modules constitutes the global state of the model. Each variable is defined with a *name*, a *type* restricted to either a finite range of integers or to a Boolean, and an initial *value*:

> *name* : *type* **init** *value*

The initial state of a process model is therefore defined by the initial values of all variables. The behaviour of a module is described through a set of local commands. Each command contains a *guard* and one or more *updates*, taking the form:

> [*label*] *guard* $\rightarrow$*update*$_1$ & ... & *update*$_n$;

A guard is a predicate over both local variables and those contained in other modules. If a command's guard equates to true, the guard's corresponding updates take place assigning one or more local variables with new values. Updating variables is equivalent to a state transition causing the model to move from its current state to a new state. Labelling commands across modules with a common label allows those commands to be synchronised such that a transition consists of all these commands operating in parallel. Such a transition will only occur when the guards of all its constituent commands equate to true. A guard of the form *true* is an empty guard that always equates to true; commands with such a guard are used to update local variables regardless of the variable's current value. These commands can

still be synchronised with others through labelling. A module containing several commands whose guards all equate to true will make a non-deterministic choice over which command to perform. PRISM allows probabilistic variable updates with commands of the form:

> [*label*] *guard* →$prob_1$: $update_1$ + ... + $prob_n$: $update_n$;

This states the probability of $update_1$ is $prob_1$ and so forth such that the sum of $prob_1$ to $prob_n$ is 1. Only one of the updates will take place with its given probability assuming the guard is true. For example, a Boolean variable available has a probability of 0.75 to be true and 0.25 to be false with a command of the form:

> [*label*] *guard* →0.75 :(available'=**true**) + 0.25 :(available'=**false**);

Guards placed on commands and probabilistic variable updates essentially form the process transition function $\mathbf{p}(s,a,s')$. Named expressions or *formulas* can be included in a PRISM process model to avoid the duplication of code and reduce the state space. Essentially a formula's expression can be substituted by its name where ever the expression would be expected.

> **formula** *name* = *expression*;

One or more reward functions may be included in a PRISM process model which return a reward for reaching a particular state, or executing a particular transition comprised of synchronised commands. We are concerned with the later which has the form:

> **rewards** "*name*"
> [*label*] *guard* : *reward*;
> **endrewards**

The "*name*" of the function is simply used to distinguish it amongst multiple reward functions, and can be removed if only one reward function is present in a process model. The label matches the label of the command in the model for which the reward should be returned. The guard is used to restrict when the reward is returned for a given command and can be set to **true** if the reward is to be returned every time the given command executes. The reward itself may be a single value or a formula combining values and variables within the PRISM model.

4.1.4 Model Building and Verification

In order to analyse a PRISM process model, it is necessary to identify which properties need to be evaluated by the PRISM model checker. PRISM's property specification language incorporates several probabilistic temporal logics, including probabilistic computation tree

logic (PCTL), continuous stochastic logic (CSL), linear time logic (LTL), and PCTL*, which incorporates both PCTL and LTL [104]. PRISM supports two quantitative properties for non-deterministic models Pmin=? [path property] and Pmax=? [path property] which return the minimum and maximum probability of satisfying a path property respectively. Reward-based properties using the R operator are also supported which relate to the expected value of a reward function. In order to calculate the maximum reward within a process model, the following verification property is encoded in a PRISM properties file:

```
R{"name"}max=? [ F p ]
```

Path properties include F, which is the *eventually* operator, such that F p is true if the statement p eventually holds in the process. In this case R{"*name*"}max =? asks PRISM to verify the maximum expected reward returned by the reward function "*name*" when the path property F p is true. In order to verify the property p is satisfied, PRISM first builds the process model by converting a process encoding into a Markov decision process. To do this PRISM computes the set of all reachable states from the initial state of the process model and the transition matrix which represents the model. The transition matrix represents the process transition function $\mathbf{p} : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \rightarrow [0,1]$, and contains the probabilities for all transitions, from state to state, that are possible in the model. A non-zero entry at row i, column j in the transition matrix represents the probability of going from state s_i to state s_j in the process.

Verifying the existence of properties in a Markov decision process using PRISM is based on exhaustive search and numerical solution [66]. The numerical result returned by PRISM is the expected maximum reward at the initial state of the Markov decision process. The model verification functionality in PRISM is implemented by four main engines *MTBDD*, *sparse*, *hybrid*, and *explicit*, all of which essentially perform the same calculations[3]. The first three engines are symbolic meaning the transition matrix is represented using data structures such as binary decision diagrams (BDDs) and multi-terminal BDDs (MTBDDs) [131]. PRISM starts with a symbolic representation of the initial state before performing fixed-point iteration. In each iteration the states are computed which can be reached within one transition step. The fourth engine, *explicit*, uses explicit state data structures to represent the transition matrix, such as sparse matrices where most entries are zero [185].

[3]http://www.prismmodelchecker.org/manual/ConfiguringPRISM/ComputationEngines

4.2 Encoding Workflow Markov Decision Processes

In this section we provide a systematic encoding in the PRISM modelling language of a workflow Markov Decision Process (MDP_W) described in Section 3.2. We do so using the workflow execution specification $WES_1 = (WS_1, ES_1)$ defined in Section 2.3.3, where WS_1 is a workflow specification and ES_1 is an execution specification. A state of the MDP_W is defined in Section 3.2 to be $s = (x, (t,u), \pi)$, where x is a workflow step, (t,u) is an assignment decision, and π is a plan. The full encoding is given in Appendix A.1.

4.2.1 Workflow Specification

We now show how a workflow specification can be encoding in PRISM using the workflow specification $WS_1 = ((T_1, <_1), U_1, (A_1, S_1, B_1))$ defined in Section 2.1.4, where $(T_1, <_1)$ is a task schema, U_1 is a set of users, and (A_1, S_1, B_1).

Task Schema

Given the task schema $(T_1, <_1)$, where $T_1 = \{t_1, t_2, t_3, t_4, t_5\}$ is a set of tasks, we consider a task variable `t` which can take the values 0 to 5 to indicate the current task $t \in T_1$, such that:

```
t : [0..5] init 0;
```

If `t=0` then no task is currently selected, meaning in the process state $s = (x, (t,u), \pi)$, the selected task t in the assignment decision (t,u) is null. Next, given the ordering of tasks $<_1 = \{(t_1,t_2), (t_1,t_4), (t_2,t_3), (t_3,t_5), (t_4,t_5)\}$ the task schema $(T_1, <_1)$ is encoded as:

```
module task_schema
t : [0..5] init 0;
[d] decision & !t1 → (t'=1);
[d] decision & t1 & !t2 → (t'=2);
[d] decision & t1 & !t4 → (t'=4);
[d] decision & t2 & !t3 → (t'=3);
[d] decision & t3 & t4 & !t5 → (t'=5);
[t] true → (t'=0);
endmodule
```

A command labelled `[d]` selects a task t as part of an assignment decision, and assigns it to `t` thus becoming the new current task. The formula `decision` must hold for any `[d]` command to execute and we encode `decision` in Section 4.2.3. The formulas `t1`, ... ,`t5` are true if the respective task t_i has already been selected. We encode these formulas in Section 4.2.4. Note, when t_1 has been selected, the non-deterministic choice between t_2 and t_4 exists if neither

have been selected. Similarly, if t_1 and t_2 have been selected, the non-deterministic choice exists between t_3 and t_4 if neither have been selected. Finally, t_5 can only be selected if the both t_3 and t_4 have been selected. The command labelled [t] is a termination action and sets the current task t to null.

Users

Given the set of users $U_1 = \{u_1, u_2, u_3, u_4\}$, we consider a user variable u which can take the values 0 to 4 to indicate the current user $u \in U_1$, such that:

```
u : [0..4] init 0;
```

If u=0 then no user is currently selected, meaning in the process state $s = (x, (t, u), \pi)$, the selected user u in the assignment decision (t, u) is null. The task of users U_1 is encoded as:

```
module users
u : [0..4] init 0;
[d] true → (u'=1);
[d] true → (u'=2);
[d] true → (u'=3);
[d] true → (u'=4);
[t] true → (u'=0);
endmodule
```

A command labelled [d] selects a user u as part of an assignment decision, and assigns it to u thus becoming the new current user. Note how these commands are synchronised with the task selection commands to form an assignment decision (t, u). As there is no guard on these commands, a non-deterministic choice exists between selecting any user $u \in U_1$. The command labelled [t] is a termination action and sets the current user u to null.

Security Policy

The workflow security policy (A_1, S_1, B_1), is defined to be:

$$A_1 = \{(t_1,u_1),(t_1,u_2),(t_2,u_2),(t_2,u_3),(t_3,u_1),(t_3,u_2),(t_4,u_2),(t_4,u_4),(t_5,u_1),(t_5,u_4)\}$$
$$S_1 = \{(t_2,t_4),(t_2,t_3),(t_3,t_4),(t_4,t_5)\}$$
$$B_1 = \{(t_1,t_3)\}$$

The set of authorisation constraints A_1 is encoded in PRISM as a set of formulas as follows:

```
formula a1 = u=1 & (t=1 | t=3 | t=5);
formula a2 = u=2 & (t=1 | t=2 | t=3 | t=4);
```

```
formula a3 = u=3 & (t=2);
formula a4 = u=4 & (t=4 | t=5);
```

A formula exists for each user, for instance a1 is true if the current user is u_1 and the current task is either t_1, t_3 or t_5. If a1 is true then u_1 is authorised to be assigned the execution of the current task t as (t_1,u_1), (t_3,u_1), and (t_5,u_1) are all in A_1. The satisfaction of all authorisation constraints in A_1 is encoded as:

```
formula a = (u=0 & x=1) | a1 | a2 | a3 | a4;
```

The condition (u=0 & x=1) states the formula a is always true in the initial state $s = (x_1, null, \pi)$. The set of separation of duty constraints S_1 are encoded as a set of PRISM modules. For instance (t_2,t_3) is encoded as:

```
module sod1
us1 : [0..4] init 0;
fs1 : bool init false;
[e] (t=2 | t=3) & us1=0 →(us1'=u);
[e] (t=2 | t=3) & us1!=0 & u=us1 →(fs1'=true);
[e] (t!=2 & t!=3) | (us1!=0 & u!=us1) →true;
[t] true → (us1'=0)&(fs1'=true);
endmodule
```

When the execution of either t_2 or t_3 is assigned a user u first, the variable us1 = u. The Boolean fs1 becomes true to indicate a policy violation, if the execution of the second task is also assigned to u. Commands are labelled [e] in this case as they can execute in parallel to an assignment event described in Section 4.2.3. The command labelled [t] is a termination action and sets the current user us1 to null and fs1 to true. Given $|S_1| = 4$, the satisfaction of all separation of duty constraints in S_1 is encoded as:

```
formula s = !fs1 & !fs2 & !fs3 & !fs4;
```

The set of binding of duty constraints S_1 are encoded in a similar way. For instance (t_1,t_3) is encoded as:

```
module bod1
ub1 : [0..4] init 0;
fb1 : bool init false;
[e] (t=1 | t=3) & ub1=0 →(ub1'=u);
[e] (t=1 | t=3) & ub1!=0 & u!=ub1 →(fb1'=true);
[e] (t!=1 & t!=3) | (ub1!=0 & u=ub1) →true;
[t] true → (ub1'=0)&(fb1'=true);
endmodule
```

Given $|B_1| = 1$, the satisfaction of all binding of duty constraints in B_1 is encoded as:

```
formula b = !fb1;
```

It follows that the satisfaction of the security policy (A_1, S_1, B_1) is encoded as the following formula:

```
formula valid = a & s & b;
```

The formula is named valid as a plan π in a state $s = (x, a, \pi)$ is valid if the formulae a, s, and b all hold true.

4.2.2 Execution Specification

We now show how an execution specification can be encoding in PRISM using the workflow specification $ES_1 = ((Z_1, <_1), \theta_1)$ defined in Section 2.3.1, where $(Z_1, \prec_1)$ is an execution schema and θ_1 is an availability forecast.

Execution Schema

Given the execution schema $(Z_1, \prec_1)$, where $Z_1 = \{x_1, x_2, x_3, x_4, x_5, x_\perp\}$ is a set of workflow steps, we consider a step variable x which can take the values -1 to 5 to indicate the current step $x \in Z_1$, such that:

```
x: [−1..5] init 1;
```

The step variable is initialised to 1 as the process starts at step x_1. Next, given the ordering of steps $\prec_1 = (x_1, x_2, x_3, x_4, x_5, x_\perp)$, the execution schema $(Z_1, \prec_1)$ is encoded as:

```
module execution_schema
x : [−1..5] init 1;
[e] x < 5 → (x'=x+1);
[e] x = 5 → (x'=−1);
endmodule
```

When x=−1 the current workflow step is the termination step $x_\perp$.

Availability Forecast

Given the availability forecast $\theta_1 : (Z \setminus \{x_\perp\}) \times U \to [0,1]$, whose values are shown in Table 2.2, we first consider a set of variable pairs tx and ux for each execution step $x \in Z \setminus \{x_\perp\}$, which can take the values 0 to 5, and 0 to 4 respectively, such that:

```
tx1 : [0..5] init 0;
ux1 : [0..4] init 0;
```

The variable tx1 is the task whose execution was assigned to a user at step x_1, while the variable ux1 is the user assigned the execution of a task at step x_1. We then encode the availability of each user $u \in U$ to form the availability forecast θ_1 as follows:

```
[e] event & x=1 & u=1 →0.8:(tx1'=t)&(ux1'=u)+0.2:(tx1'=0);
[e] event & x=2 & u=1 →0.8:(tx2'=t)&(ux2'=u)+0.2:(tx2'=0);
[e] event & x=3 & u=1 →0.8:(tx3'=t)&(ux3'=u)+0.2:(tx3'=0);
[e] event & x=4 & u=1 →0.6:(tx4'=t)&(ux4'=u)+0.4:(tx4'=0);
[e] event & x=5 & u=1 →0.8:(tx5'=t)&(ux5'=u)+0.2:(tx5'=0);
```

These commands are the availability of user u_1 at each execution step $x \in Z \setminus \{x_\perp\}$. Each command labelled [e] is a probabilistic assignment event. For instance, if the current step is x_1 and the current user is u_1, then tx1 becomes the current task t and ux1 becomes u_1 with the probability $\theta_1(x_1,u_1) = 0.8$. This indicates the assignment event, assigning the execution of t to u_1 has succeeded. Otherwise the assignment event fails and tx1 becomes null. In this case the number of assigned tasks is one less than the number of completed steps, forcing the process to terminate. We discuss the encoding of this in Section 4.2.3. The formula event must hold for any [e] command to execute and we encode event in Section 4.2.3. All tx, ux, and availability forecast commands [e] are encoded within a single PRISM module as shown in Appendix A.1.

4.2.3 Process Actions

We now show how the properties defined in Section 3.2.1 are encoded, which if satisfied, allow either an assignment action, assignment decision, or a termination action to be executed.

Assignment Decisions

Two properties were defined in Section 3.2.1 which must be satisfied by a state $s = (x,a,\pi)$ for an assignment decision action to be executed in s. The first property we consider, Property 3.2, is defined to be:

$$\forall t'' \in T, t'' < t', \exists u \in U \Rightarrow \pi(t'') = u$$

This states task ordering must be respected and is satisfied by the task_schema module described in Section 4.2.1. The second property, Property 3.1, is defined to be:

$$x_i \neq x_\perp \wedge \pi \text{ is valid} \wedge |R| = i-1 \wedge (t \in R \vee R = \emptyset)$$

Here $\pi : R \rightarrow U$ is a plan, where $R \subseteq T$ is a set of tasks. To encode this property we first encode the formula:

```
formula null = t=0 & u=0;
```

Which only holds true in both the initial state of the process, where $R = \emptyset$, and any termination state $s = (x, null, \pi)$. Next we encode the formula:

```
formula tinR = (x=2&tx1=t) | (x=3&tx2=t) | (x=4&tx3=t) | (x=5&tx4=t) |(x=-1&tx5=t);
```

The formula tinR holds true if the execution of the current task t has been assigned to a user such that $t \in R$. Finally we encode Property 3.1 as the formula decision :

```
formula decision = x!=-1 & valid & sp=x-1 & (x=1 & null | tinR);
```

Given a state $s = (x, a, \pi)$, the formula valid indicates the plan π is valid, and is encoded in Section 4.2.1. The formula sp is equivalent to $|R|$, and is defined in Section 4.2.4. Finally, x=1 & null allows an assignment decision to execute in the initial state of the process.

Assignment Events

Two properties were defined in Section 3.2.1 which must be satisfied by a state $s = (x, (t, u), \pi)$ for an assignment event action to be executed in s. The first property we consider, Property 3.4, is defined to be:

$$x_j = x_i \wedge u_q = u_p$$

This states task workflow step x must equal the current workflow step x, and the user u must equal the current user u. This property holds true by virtue of an assignment decision command [d], described in Section 4.2.1, which moves the process to state s from a state $s' = (x, (t, u'), \pi)$. The command [d] changes the current user u from u' to u, but does not change the step x in state s'. The second property, Property 3.3, is defined to be:

$$x_i \neq x_\perp \wedge |R| = i - 1 \wedge t \notin R$$

We encode Property 3.3 as the formula event:

```
formula event = x!=-1 & sp=x-1 & !tinR;
```

The formula sp is equivalent to $|R|$, and is defined in Section 4.2.4, and the formula tinR is defined in the previous section.

Termination Actions

Two properties were defined in Section 3.2.1, either of which must be satisfied by a state $s = (x,(t,u),\pi)$ for a termination action to be executed in s. The first property we consider, Property 3.5, is defined to be:

$$\pi \text{ is valid} \wedge R = T$$

Here $\pi : R \to U$ is a plan, where $R \subseteq T$ is a set of tasks. To indicate a plan π is complete we encode the formula:

```
formula complete = tx1!=0 & tx2!=0 & tx3!=0 & tx4!=0 & tx5!=0;
```

We then encode Property 3.5 as the command:

```
[t] valid & complete →true;
```

Given a state $s = (x,a,\pi)$, the formula valid defined in Section 4.2.1, indicated the plan π is valid. The second property, Property 3.6, is defined to be:

$$(x_i \neq x_\perp \wedge \pi \text{ is valid} \wedge |R| < i-1) \vee (x_i = x_\perp \wedge R \neq T) \vee (x_i \neq x_1 \wedge \pi \text{ is invalid})$$

Property 3.6 is then encoded as a command:

```
[t] (x!=−1 & sp!=x−1 & valid) | (x=−1 & !complete) | (x!=1 & !valid) →true;
```

The formula sp is equivalent to $|R|$, and is defined in Section 4.2.4. Both termination commands [t] are encoded within a single PRISM module as shown in Appendix A.1. To determine whether the process is in a termination state $s = (x_i, null, \pi)$, where the workflow step x_i is not the first workflow step x_1, we encode the formula:

```
formula terminated = x!=1 & null;
```

4.2.4 Plan

Given a state $s = (x,a,\pi)$, the combination of all tx and ux variables introduced in Section 4.2.2 forms the plan π, by indicating the task t selected at each step x, and the user u assigned the execution of t. Understanding whether the execution of each task $t \in T$ has been assigned is captured by the following formulae:

```
formula t1 = tx1=1 | tx2=1 | tx3=1 | tx4=1 | tx5=1;
formula t2 = tx1=2 | tx2=2 | tx3=2 | tx4=2 | tx5=2;
formula t3 = tx1=3 | tx2=3 | tx3=3 | tx4=3 | tx5=3;
```

```
formula t4 = tx1=4 | tx2=4 | tx3=4 | tx4=4 | tx5=4;
formula t5 = tx1=5 | tx2=5 | tx3=5 | tx4=5 | tx5=5;
```

These formula are used to satisfy task ordering when selecting tasks, as shown in Section 4.2.1. Given $\pi : R \rightarrow U$ is a plan, where $R \subseteq T$ is a set of tasks, the size of a plan is equivalent to $|R|$, and encoded by the formula sp:

```
formula sp = (tx1/tx1) + (tx2/tx2) + (tx3/tx3) + (tx4/tx4) + (tx5/tx5);
```

4.2.5 Reward Functions

We now show how the reward functions defined in Section 3.2.2 can be encoded in PRISM. The first reward function $\mathbf{r}_Q$ is used to maximise the MDP_W optimal value function V^* in order to generate the quantitative satisfiability and resiliency metric for a workflow:

$$\mathbf{r}_Q((x_i,a,\pi),a',(x_i,a'',\pi)) = \begin{cases} 1 & \text{if } \pi \text{ is valid} \wedge R = T \wedge a' \in \mathbb{T} \\ 0 & \text{otherwise} \end{cases}$$

This is encoded in PRISM as:

```
rewards "quantitative"
[t] complete & valid : 1;
endrewards
```

A reward of 1 is returned when, from a state $s = (x,a,\pi)$ where the plan π is complete and valid, the process transitions to a termination state by executing a termination action in state s. Note, the PRISM encoded reward function is synchronised to return a reward of 1 when a termination command [t] executes. The second reward function $\mathbf{r}_D$ is used to maximise the MDP_W optimal value function V^* in order to generate the distance resiliency metric for a workflow:

$$\mathbf{r}_D((x_i,a,\pi),a',(x_i,a'',\pi)) = \begin{cases} 1 & \text{if } \pi \text{ is valid} \wedge R = T \wedge a' \in \mathbb{T} \\ 1 & \text{if } i \neq 1 \wedge a' \in \mathbb{D} \\ 0 & \text{otherwise} \end{cases}$$

This is encoded in PRISM as:

```
rewards "distance"
[t] complete & valid : 1;
[d] x > 1 : 1;
endrewards
```

A reward of 1 is returned when, from a state $s = (x, a, \pi)$ where the plan π is complete and valid, the process transitions to a termination state by executing a termination action in state s. A reward of 1 is also returned when, from a state $s = (x, a, \pi)$ where the step x is not the first workflow step x_1, the process transitions to a state s' by executing an assignment decision in state s. Note, the PRISM encoded reward function is synchronised to return a reward of 1 when either a termination command `[t]` or assignment decision `[d]` executes.

4.3 Workflow Analysis

In this section we carry out analysis of the workflow execution specifications defined in Chapters 2 and 3 using the probabilistic model checker PRISM. We define the verification properties needed for PRISM to automatically analyse the MDP_W models and generate workflow satisfiability and resiliency metrics. We then illustrate how a Chief Information Security Officer (CISO) can analyse the impact of a security policy on workflow completion by making arbitrary changes to the security policy of one of the example workflow execution specifications. We then give an example of the computational overheads encountered when generating workflow metrics with PRISM.

4.3.1 Verification Properties

In this section we define two reward-based properties needed for PRISM to automatically analyse the MDP_W models and generate workflow satisfiability and resiliency metrics. The first property verifies the maximum reward in the initial process state s_0 expected when the process eventually terminates, that is the maximum expected reward return by the reward function $\mathbf{r}_Q$. The maximum expected reward is equivalent to the value returned by the optimal value function $V^*(s_o)$ of the MDP_W $(\mathbb{S}, \mathbb{A}, \mathbf{p}, \mathbf{r}_Q)$. This reward-based property is encoded in PRISM as:

```
R{"quantitative"}max=? [F terminated]
```

The property asks PRISM to verify the maximum reward, using the `"quantitative"` reward function, expected when the process terminates. The path property `terminated` is defined in Section 4.2.3. In this case a reward of 1 is received when the process terminates successfully. It is possible to encode an equivalent quantitative property in PRISM using the formulae `complete` and `valid` defined in Sections 4.2.3 and 4.2.1 respectively:

```
Pmax=? [F complete & valid]
```

This property asks PRISM to verify the maximal probability of eventually reaching a state $s = (x, a, \pi)$ where π is a complete and valid plan. The second property verifies the maximum reward in the initial process state s_0 expected when the process eventually terminates, that is the maximum expected reward return by the reward function $\mathbf{r}_D$. The maximum expected reward is equivalent to the value returned by the optimal value function $V^*(s_o)$ of the MDP$_W$ $(\mathbb{S}, \mathbb{A}, \mathbf{p}, \mathbf{r}_D)$. This reward-based property is encoded in PRISM as:

```
R{"distance"}max=? [F terminated]
```

The property asks PRISM to verify the maximum reward, using the "distance" reward function, expected when the process terminates. In this case a reward of 1 is received when each workflow step is completed successfully, and when the process terminates successfully.

4.3.2 Security Impact Analysis

In this section we generate workflow metrics for the workflow execution specifications defined in Chapters 2 and 3. We then show by example how a CISO could use workflow metrics to analyse the potential impact of modifying a workflow security policy.

Analysing Satisfiability and Resiliency

When analysing the potential impact of a policy modification on workflow completion, it is intuitive to begin by generating the quantitative satisfiability of a workflow execution specification. As discussed in Section 2.3.4, a workflow execution specification can be satisfiable and not resilient, but it cannot be resilient and unsatisfiable. We generate the quantitative satisfiability Γ_S for the workflow execution specifications WES_{21} and WES_{22} defined in Section 3.2.3, and WES_1 defined in Section 2.3.3. Generating quantitative satisfiability requires a workflow execution specification $WES = (WS, ((Z, \prec), \theta))$ to come with a full availability forecast θ, that is all users have a probabilistic availability of 1 for every execution step. Supplying WES_{22} and WES_1 with a full availability forecast, the quantitative satisfiability of all three workflow execution specifications is 1. The workflow execution specifications WES_{21} already comes with a full availability forecast making WES_{21} and WES_{22} equivalent in this case.

Next the quantitative resiliency Γ_Q is generating for WES_{21}, WES_{22}, and WES_1 with their original availability forecasts, that is full in the case of WES_{21} and probabilistic for both WES_{22}, and WES_1. The full PRISM encodings for all three workflow execution specifications are listed in Appendices A.1 to A.3, while the PRISM state diagrams of the MDP$_W$ for both WES_{21} and WES_{22} are shown in Appendix A.4. Note that the two state

Table 4.1 Workflow metrics for workflow execution specifications WES_{21} coming with a full availability forecast, and WES_{22} and WES_1, both coming with a probabilistic availability forecast.

Workflow Metric	WES_{21}	WES_{22}	WES_1
Quantitative satisfiability Γ_S	1.00	1.00	1.00
Quantitative resiliency Γ_Q	1.00	0.48	0.41
Distance resiliency Γ_D	2.00	1.08	3.03

diagrams are equivalent to the two MDP_W manual calculation diagrams shown in Figures 3.7 and 3.8 respectively. The quantitative resiliency values generated using PRISM are shown in Table 4.1. Note, because WES_{21} comes with a full availability forecast and $\Gamma_S(WES_{21}) = 1$, its quantitative resiliency is also 1. Note also, $\Gamma_S(WES_{22}) = 0.48$, which matches the value manually calculated for WES_{22} in Section 3.2.3.

Next the distance resiliency Γ_D is generating for WES_{21}, WES_{22}, and WES_1 using PRISM, which are shown in Table 4.1. Note, because WES_{21} comes with a full availability forecast and $\Gamma_S(WES_{21}) = 1$, its distance resiliency is 2, which is the number of tasks in WES_{21}. Note also, $\Gamma_D(WES_{22}) = 1.08$, which matches the value manually calculated for WES_{22} in Section 3.2.3.

Analysing Policy Modifications

We now consider a scenario where a CISO has reviewed current organisational security requirements and identified six potential security policy modifications, $\mathbf{m_1}, \ldots, \mathbf{m_6}$, for workflow execution specification $WES_1 = ((T_1, <_1), U_1, (A_1, S_1, B_1))$. Given the security policy (A_1, S_1, B_1) the the five potential modifications are:

- $\mathbf{m_1}$: $A_1 \cup \{(t_5, u_3)\} \rightarrow$ a user is newly qualified to execute a task
- $\mathbf{m_2}$: $S_1 \cup \{(t_3, t_5)\} \rightarrow$ a guideline update suggests a new separation of duty
- $\mathbf{m_3}$: $A_1 \setminus \{(t_5, u_1)\} \rightarrow$ a change in schedule means user can no longer execute task
- $\mathbf{m_4}$: $B_1 \cup \{(t_2, t_5)\} \rightarrow$ a CISO wants to achieve compliance with a security standard
- $\mathbf{m_5}$: $A_1 \setminus \{(t_1, u_1), (t_1, u_2)\} \rightarrow$ a change in technology means users are no longer qualified a for task
- $\mathbf{m_6}$: $S_1 \cup \{(t_1, t_2)\} \rightarrow$ a new security threat has emerged between tasks

Table 4.2 Workflow metrics for workflow execution specification WES_1, where $\mathbf{m_i}$ is a modification applied to the security policy of WES_1.

Workflow Metric	WES_1	$\mathbf{m_1}$	$\mathbf{m_2}$	$\mathbf{m_3}$	$\mathbf{m_4}$	$\mathbf{m_5}$	$\mathbf{m_6}$
Quantitative satisfiability Γ_S	1.00	1.00	1.00	1.00	0.00	0.00	1.00
Quantitative resiliency Γ_Q	0.31	0.36	0.05	0.00	0.00	0.00	0.31
Distance resiliency Γ_D	2.93	2.98	2.61	2.61	2.23	0.00	2.93

Each policy modification has been applied to WES_1 and the workflow metrics generated using PRISM. In order to generate the quantitative satisfiability of WES_1, it is supplied with a full availability forecast. The generated metrics are shown in Table 4.2, where column WES_1 gives the values before any security policy modification, and column $\mathbf{m_i}$ gives the values for WES_1 with policy modification $\mathbf{m_i}$. Note that if WES_1 has a level of quantitative resiliency, that is $0 < \Gamma_Q(WES_1) \leq 1$, then WES_1 is also satisfiable, such that $\Gamma_S(WES_1) = 1$. On the other hand, if WES_1 is not satisfiable, that is $\Gamma_S(WES_1) = 0$, then WES_1 has 0 quantitative resiliency, such that $\Gamma_Q(WES_1) = 0$. It follows that, given any workflow execution specification, if $\Gamma_S(WES) = 0$ and the availability forecast of WES is full, then $\Gamma_Q(WES) = 1$.

Following modifications $\mathbf{m_1}$ and $\mathbf{m_2}$, WES_1 still maintains a level of resiliency, albeit impacted by the modifications. In the case of $\mathbf{m_1}$, which authorises a user to execute a task, and arguably weakens security, the quantitative and distance resiliency of WES_1 increases, as one might expect. In the case of $\mathbf{m_2}$, which adds a separation of duty, and arguably strengthens security, the quantitative and distance resiliency of WES_1 decreases, as one might again expect. Following modification $\mathbf{m_3}$, the quantitative resiliency of WES_1 is 0, that is, $\Gamma_Q(WES_1) = 0$, yet WES_1 is still satisfiable. This shows there exists at least one complete and valid plan π, but none are feasible. Interestingly, WES_1 still maintains a level of distance resiliency, albeit reduced, and indicates 2.61 tasks can be expected to be completed before the security policy causes WES_1 to become deadlocked. With modification $\mathbf{m_4}$, WES_1 is no longer satisfiable, such that $\Gamma_S(WES_1) = 0$, which means the quantitative resiliency of WES_1 is also 0 in this case. Even in this situation, WES_1 still maintains a level of distance resiliency which indicates the security policy is expected to cause WES_1 to become deadlocked after 2.23 execution steps. Applying modification $\mathbf{m_5}$ shows a situation where the security policy causes WES_1 to have 0 quantitative satisfiability and resiliency, and 0 distance resiliency. This indicates the security policy causes WES_1 to deadlock at the first execution step, which is clearly the case as modification $\mathbf{m_5}$ removes all users authorised to execute the first task t_1. Finally, modification $\mathbf{m_6}$ highlights a situation where a security constraint can be added, a

separation of duty constraint in this case, and have no impact on the quantitative and distance resiliency of a workflow.

4.3.3 Computational Overheads

In this section we give an indication of the main computational overheads encountered when using PRISM to generate workflow metrics. We do so by generating workflow metrics for the workflow execution specifications WES_1, defined in Chapter 2, and WES_{21} and WES_{22} defined in Chapter 3. In doing so we consult the analysis logs provided by PRISM, and the execution platform's file storage system regarding computation overheads. The workflow metrics are generated by executing PRISM on a standard MacBook Pro laptop incorporating a 2.7Ghz Intel Core i5 processor, 16GB RAM and an OS X 10.12.5 operating system. To take into account any influence the computing platform may have on timing overheads, each metric is generated 20 times and the average taken. The workflow metrics are generated using an unmodified version 4.3.1 of the PRISM model checker running the sparse engine. The available computational data allows us to consider the following overheads when generating workflow metrics:

- **Model build time** : time taken by PRISM to construct the MDP_W model
- **Verification time** : time to by PRISM to generate workflow metric
- **States** : total number of reachable states in the MDP_W model
- **Transitions** : total number of transitions between reachable states in the MDP_W model
- **Memory** : total memory allocated to PRISM while generating workflow metric
- **File size** : size of file containing PRISM encoding of MDP_W
- **Size on disk** : storage space allocated by execution platform for PRISM encoding

The computation overheads for generating the metrics for WES_1, WES_{21}, and WES_{22} are shown in Table 4.3. Note when generating the same metric for the same workflow execution specification, certain overheads such as model build time and verification time may fluctuate depending on the execution platform. Other overheads such as the number of states and transitions remain fixed as these are dependent on the MDP_W. An MDP_W model need only be built once, that is, each time it is loaded into PRISM for verification. Verification time however must be accepted each time a metric is generated from the pre-built MDP_W model.

As one would expect, build times, memory requirements, the number of states, and the transitions between states increases as the 'size' of a workflow increases. For instance WES_{21} consists of 2 tasks, 2 users, and 1 separation of duty constraint, while WES_1 consists of 5 tasks, 4 users, and 5 separation and binding of duty constraints. However, with just a

Table 4.3 Computation overheads when using PRISM to generate workflow metrics for workflow execution specifications WES_1, WES_{21}, and WES_{22}.

Overhead	WES_{21}	WES_{22}	WES_1
Model build time (s)	0.002	0.003	0.206
Verification time (s)	0.001	0.001	0.039
Total runtime (s)	0.003	0.004	0.245
States	12	18	719
Transitions	14	24	1107
Memory (KB)	0.70	1.10	44.60
File size (KB)	1.92	1.93	5.42
Size on disk (KB)	4.00	4.00	8.00

small increase in tasks, users, and security constraints, the number of states and transitions increases significantly. This would indicate computation overheads are strongly linked to the 'complexity' of a workflow, in terms of the number of tasks, users and security constraints. Workflow complexity can therefore be measured by the number of states and transitions in the corresponding MDP_W. Note however WES_{21} and WES_{22} are equivalent in terms of tasks, users, and security constraints; the difference lays with the availability forecast, which is full for WES_{21} and probabilistic for WES_{22} (Section 2.3.1). The fact that users may be unavailable or not increases the number of states and transitions, and therefore adds to the complexity of a workflow. We consider workflow complexity further in Chapter 5 and show by continually adding security constraints to a workflow's security policy, the number of states can actually start to decrease after a certain point.

We now consider the six security policy modifications, $\mathbf{m_1}, \ldots, \mathbf{m_6}$, introduced in Section 4.3.2 and add them in turn to workflow execution specification WES_1. We then generate workflow metrics for WES_1 with each policy modification and record the computation overheads shown in Table 4.4. For modifications $\mathbf{m_1}, \ldots, \mathbf{m_4}$ the number of states, 719, remains the same as WES_1 without any policy modifications despite the changes in the quantitative resiliency values shown in Table 4.2. This unchanging number of states can be attributed to the distance resiliency which remains at between 2.98 and 2.23 after each policy modification $\mathbf{m_1}, \ldots, \mathbf{m_4}$. Instead, it is the number of transitions which is affected by adding or removing policy constraints. It is only when distance resiliency is 0, following policy modification $\mathbf{m_5}$, that both states and transitions reduce significantly. Memory usage and file size gives a sense of resource cost when making a security policy modification. For example, modification $\mathbf{m_1}$ which adds an authorisation constraint 'costs' a file size increase of 0.01KB but reduces

Table 4.4 Computation overheads when using PRISM to generate workflow metrics for workflow execution specification WES_1 before and after security policy modifications $\mathbf{m_1}, \ldots, \mathbf{m_6}$.

Overhead	WES_1	$\mathbf{m_1}$	$\mathbf{m_2}$	$\mathbf{m_3}$	$\mathbf{m_4}$	$\mathbf{m_5}$	$\mathbf{m_6}$
Model build time (s)	0.206	0.165	0.221	0.143	0.172	0.092	0.141
Verification time (s)	0.039	0.037	0.038	0.043	0.038	0.002	0.024
Total runtime (s)	0.245	0.202	0.259	0.186	0.210	0.094	0.165
States	719	719	719	719	719	18	652
Transitions	1107	1095	1107	1119	1107	25	1003
Memory (KB)	44.60	44.50	44.50	44.70	44.50	1.10	40.50
File Size (KB)	4.58	4.59	4.81	4.57	4.82	4.57	4.82
Size on Disk (KB)	8.00	8.00	8.00	8.00	8.00	8.00	8.00

memory usage of PRISM by 0.1KB when generating metrics. Modification $\mathbf{m_2}$ which adds a separation of duty constraint 'costs' a file size increase of 0.23KB while also decreasing memory usage of PRISM by 0.1KB. Modification $\mathbf{m_6}$ had no impact on the resiliency of WES_1, yet interestingly, the number of states and transitions is reduced, along with the total runtime, and the amount of memory allocated to PRISM for generated a workflow metric.

4.4 Related Work

In this section we introduce current work related to model checking and the probabilistic model checker PRISM.

4.4.1 Model Checking

Several works appear in the literature using model checking to verify properties of security constrained workflows. In [3], Armando et al., present a formal definition based on the action language *C* [70], of workflows and security policies consisting of an RBAC access control model, separations of duty, and delegation constraints. Automatic analysis is provided by the Causal Calculator (CCALC) [167], to verify whether the workflow can reach completion given a set of resources, essentially providing a solution to the workflow satisfiability problem. A resource allocation plan (akin to a user-task assignment) can be extracted using the minimal number of resources in a similar way to the graph colouring technique used by Kohler and Schaad in [99]. Armando and Ponta in [4, 5] use model checking to verify authorisation requirements are correctly defined in a workflow specification. This work

considers separating the encoding of the workflow and the access control policy within the model checker. A number of detailed authorisation and data dependency requirements are considered and encoded in their framework allowing fine-grained access control policies to be expressed. Automated model checking is used to ensure firstly that workflow security properties are present in a BPMN workflow description, and secondly a resource allocation plan can be found under those security properties as considered in [3]. Although these work consider the workflow satisfiability problem and start to think in terms of how different security constraints (similar to policy modifications) impact the satisfiability of a workflow they do not consider probabilistic availability of resources or workflow resiliency.

Quantitative analysis of security impact on workflow performance using model checking techniques is considered by He et al., in [76]. The authors consider hybrid workflows consisting of both human and computer based tasks annotated with synthetic performance data. A rich set of security constraints is considered including RBAC, separations and bindings of duty, cardinality constraints and delegation through role hierarchies. The timed Colour Petri-Net (TCPN) formalism [90], is applied to model workflow execution and authorisation allowing various workflow performance metrics to be computed using a Petri-net based model checker. Metrics including authorisation overheads similar to those generated in the work by Parkin et al., [141], and performance based metrics such as resource utilisation and workflow success rate are generated in order to identify unacceptable overheads and workflow bottlenecks caused by the security constraints as discussed by Basin et al, in [12]. The unavailability or users, and therefore the generation of resiliency metrics for a workflow however is not considered. Methods to improve workflow performance such as modifying security constraints are also outlined but techniques to manage the risk of poor performance when security constraints cannot be modified are not.

4.4.2 PRISM

In support of our choice of tool, PRISM has been used for optimisation and quantitative verification in several works appearing in the literature. Notably, Calinescu et al., in [25] use PRISM for model checking and quantitative verification of software that can self-adapt in response to changing system objectives and the environment in which it exists. The motivation for this approach is to prevent software changes that would lead to errors in the newly adapted and deployed software. Therefore model checking and quantitative verification is performed at runtime to predict and identify violations of the PRISM encoded system requirements (akin to identifying security policy violations), optimise steps to avoid

requirement violations (similar to optimising workflow task assignments), and provide proof of system requirement satisfiability. PRISM is also used in a similar approach by Calinescu et al., in [26] who incorporate it into a framework for dynamically composing service-based workflow systems at runtime in order to provide self-adapting, complex and adaptive functionality in accordance with workloads and other environmental factors. This is similar to the quality of service optimisation problem considered by Gao et al., [68] who also use MDPs in their approach as mentioned above. Using PRISM in this case enables automatic optimised service composition inline with quality of service requirements formalised and encoded into the model checker.

In terms of workflow PRISM has been used by Herbert and Sharp in [77] as part of a framework to model workflows and carry out quantitative analysis on them. The core of this work is the definition of an algorithm to automatically translate workflows expressed in BPMN to an equivalent MDP encoding in PRISM in order to perform direct analysis of several quantitative workflow properties. A general description is provided on specifying analysis properties in PRISM in order to verify whether particular states of interest can be reached, or checking the occurrence and correct ordering of certain events modelled in the state of the MDP encoding. A high-level outline is provided on encoding workflow control patterns in PRISM such as sequential task ordering, parallel task execution and choice points but other workflow aspects such as encoding security policies, users, probabilistic availability, and task assignment is not considered, nor is the verification of workflow resiliency.

4.5 Summary

Manual generation of workflow metrics by solving a workflow Markov decision process (MDP_W) can be a complex, time consuming and error prone process even for small scale workflows like the examples we consider in the first part of this thesis. For instance, the MDP_W for workflow execution specification WES_1 defined in Section 2.3.3, coming with 5 tasks, 4 users and 5 separation and binding of duty constraints, has 719 states. The application of a Markov decision process approach may not be practical, nor fully understood, by a Chief Information Security Officer (CISO) when generating metrics to analyse the potential impact of security policy modifications. In this chapter we have taken the first steps in providing a more accessible and efficient way of generating metrics by using PRISM, which is a tool for modelling and automatically verifying properties in probabilistic models such as Markov decision processes.

PRISM is a free, open-source tool developed and managed by active researchers at Oxford University, and features in over 500 peer reviewed publications at the time of writing. We have given an introduction to the concept of probabilistic model checking, before describing the PRISM model checker, the PRISM high level state-based modelling language, and the building and property verification of models using PRISM. We then outlined a systematic way of encoding an MDP_W in PRISM before describing how the tool can be used to generate quantitative satisfiability and resiliency metrics. Using PRISM we generate metrics for example workflow execution specifications and showed the results match those of manual calculations undertaken in Chapter 3. Next we showed how a CISO could use these metrics to analyse the potential impact different security policy modifications have on a workflow's completion. By example, we highlighted the resiliency of a workflow may be reduced or increased by a policy modification, yet the workflow still maintain a level of quantitative resiliency. We also showed a workflow can have 0 quantitative resiliency but still be satisfiable, and a workflow that has 0 quantitative satisfiability and resiliency can still have a level of distance resiliency. Furthermore, we showed a workflow that is unsatisfiable has 0 quantitative resiliency. We then gave an indication of the main computational overheads encountered when using PRISM to generate workflow metrics. These include the size of an MDP_W model in terms of states and transitions, the amount of memory allocated to PRISM for generating metrics, and model building and verification times. We highlighted that some overheads are dependant on the execution platform and may fluctuate, while others are dependant on the MDP_W model and are fixed.

Having shown how satisfiability and resiliency metrics can be generated for a workflow execution specification we have not so far considered cases where security policy impact analysis indicates the resiliency of a workflow is expected to fall to an unacceptable level. This may be an issue for workflows that cannot be made acceptably resilient but must still be executed. In Chapter 5 we outline two techniques that use resiliency metrics to help manage the risk of potential workflow failure. The first technique considers generating resiliency metrics at runtime, ensuring the execution of each task is assigned to users in a way that maximises resiliency in accordance with the current user availability forecast. We utilise the observation that constraints can be added to a security policy without affecting the resiliency yet reduce metric computation time. The second technique considers workflows with choice and introduces new resiliency metrics called *expected resiliency* and *resiliency variance*. We then show how these metrics can be used to help form mitigation strategies for overcoming situations where user unavailability would otherwise force a workflow to become deadlocked.

Chapter 5

Workflow Risk Management

In some cases, the quantitative resiliency of a workflow may fall below an acceptable level of risk when analysing the impact of security policy modifications. This can be of particular concern when policy modifications are enforced by regulation updates, especially for workflows in critical domains such as healthcare where the tolerance for workflow failure is small. In Chapter 2, the quantitative resiliency of a security constrained workflow was shown to indicate the success rate of a workflow, that is the expected probability a workflow can complete whilst satisfying all security constraints under the uncertainty of user availability. In Chapter 3, the process of assigning the execution of tasks to users who may be unavailable was modelled as a Markov decision process (MDP). Solving a workflow Markov decision process (MDP_W) with appropriate rewards functions was shown to generate quantitative measures of workflow satisfiability and resiliency. In Chapter 4, an encoding was given of an MDP_W in the probabilistic model checker PRISM which is used to automatically generate quantitative workflow metrics for a Chief Information Security Officer (CISO) to analyse the potential impact of security policy modifications.

This chapter explores two techniques showing how calculating quantitative resiliency can help manage the risk of workflow failure. The first technique is aligned with reducing failure risk by considering the computation of resiliency at runtime to ensure the execution of tasks maximises the quantitative resiliency of the workflow in accordance with a current prediction of user availability. We show by adding artificial constraints to a security policy, the resiliency verification time can be reduced without impacting the resiliency value. The second technique is aligned with failure risk acceptance by considering the formation of mitigation strategies, in particular calculating resiliency metrics to help form strategies for workflows containing choice. We introduce a new resiliency variance metric which can give a prediction of failure risk for such workflows.

5.1 Risk Reduction

Maximising the resiliency of a workflow at runtime would require its quantitative resiliency to be recomputed at each workflow step x, given the current user availability forecast, before assigning the execution of a task t at step x to a user u. Generating resiliency metrics at runtime could ensure the task execution assignments are only granted if the remainder of the workflow has a resiliency above a given resiliency threshold. Generating workflow metrics can be computationally demanding therefore generating quantitative resiliency metrics at runtime has itself an impact on workflow execution time. This means, in some cases, techniques are required that facilitate fast generation of workflow metrics.

In Chapter 4, the probabilistic model checker PRISM was used to automatically generate quantitative resiliency metrics to analyse the resiliency of a workflow execution specification (WES) at design time. Quantitative resiliency metrics are generated assuming an availability forecast, which defines the expected availability of users at each workflow step, does not change during the execution of a workflow. If the execution time is long, say over a number of days, the availability forecast may well change. At runtime, a user is either available or not. As discussed in Section 2.3.4, this means quantitative resiliency gives an expectation of successful workflow completion, not a level of workflow successful completion. A workflow only completes successfully if the execution of all tasks have been assigned to users without violating the workflow security policy, and all users are available to execute their assigned tasks.

A plan π defined in Section 2.2.1, assigns the execution of workflow tasks to users. When the availability forecast does not change at runtime, the optimal complete and valid plan π, that is the plan that maximises the quantitative resiliency of a workflow, does not change at runtime. However, when user availability is dynamic, the optimality of π might change during workflow execution, meaning the new optimal plan might need to be found. In [42], Crampton and Khambhammettu describe two workflow execution models; a workflow-driven execution model (WDEM), where users are automatically assigned the execution of tasks, and user-driven execution model (UDEM), where users initiate requests to be assigned the execution of tasks. The impact of dynamic user availability is slightly different between the two models. With WDEM it may be necessary to continuously compute the optimal valid plan π and so adapt to changes in user availability. With UDEM it may be necessary to ensure a user requesting to execute a task is assigned that task according to the optimal valid plan π, or a plan π' that delivers a level of resiliency which satisfies some threshold

requirement. In the case of either model, quantitative resiliency might need to be recomputed several times at runtime.

In this section, we investigate how a Chief Information Security Officer (CISO) could analyse and modify a workflow security policy to improve the computation time for generating quantitative resiliency metrics at runtime. We first show that adding or removing security components to a security policy has a clear impact on resiliency computation time, that is, the computation time can either increase or decrease. We then propose a methodology to help a CISO compute a set of security constraints that can be artificially added to a security policy, without impacting the resiliency value, but which significantly decreases resiliency metric computation time.

5.1.1 Empirical Assessment of Policy Modifications

In this section we provide an empirical assessment of quantitative resiliency computation time to help understanding of how it can be improved at runtime. In doing so we investigate the impact upon computation time of adding security constraints to a workflow security policy. First we consider $WES_3 = (((T_3, <_3), U_3, (A_3, S_3, B_3)), ((Z_3, \prec_3), \theta_3))$ to be a workflow execution specification consisting of 10 tasks which are to be executed sequentially, and 5 users. For reference, the workflow specification $WS_3 = ((T_3, <_3), U_3, (A_3, S_3, B_3))$ and execution specification $ES_3 = ((Z_3, \prec_3), \theta_3)$ is defined in (Appendix B.1). We only consider adding separation of duty constraints to the security policy (A_3, S_3, B_3), which is sufficient to show the impact adding constraints has on resiliency computation time.

The workflow execution specification WES_3 coming with 10 tasks, means the maximum number of separation of duty constraints S_3 can contain is 45. Given a discrete variable k, such that $0 \leq k \leq 45$, we generate 100 random security policies (A'_3, S'_3, B'_3) under the following constraints. First, the set of authorisation constraints A'_3 permits between 2 and 5 users in U_3 to be assigned the execution of each task $t \in T_3$, such that $20 \leq \left|A'_3\right| \leq 50$. Next the set of separation of duty constraints S'_3 contains k constraints, more precisely $\left|S'_3\right| = k$, and the set of binding of duty constraints B'_3 contains 0 constraints, that is $B'_3 = \emptyset$. A total of 4600 security policies are generated denoted by the set F.

Each security policy $(A'_3, S'_3, B'_3) \in F$ is applied to the workflow execution specification WES_3 such that $WES'_3 = (((T_3, <_3), U_3, (A'_3, S'_3, B'_3)), ES_3)$. We the generate the quantitative resiliency metric for each WES'_3 by following the process described in Chapter 4, that is WES'_3 is encoding as a workflow Markov decision process (MDP_W) in the probabilistic model checker PRISM which automatically generates the quantitative resiliency of WES'_3

Table 5.1 Impact results of modifications to the security policy of workflow execution specification WES_3, where each column $\mathbf{i} \sim \mathbf{j}$ shows the average impact of adding between i and j separation of duty constraints.

	0	**1 ∼ 5**	**6 ∼ 10**	**11 ∼ 15**	**16 ∼ 20**
Quantitative resiliency Γ_Q	0.582	0.580	0.557	0.528	0.505
# 0 resilient WES'_3	0	0	0	1	0
Model build time (s)	0.56	2.83	16.12	25.91	21.72
Verification time (s)	0.11	0.38	1.56	2.24	1.80
Total runtime (s)	0.67	3.21	17.68	28.15	23.52
States	3893	58246	346992	600287	522850
Transitions	73249	758351	3352889	4754705	3649065

	21 ∼ 25	**26 ∼ 30**	**31 ∼ 35**	**36 ∼ 40**	**41 ∼ 45**
Quantitative resiliency Γ_Q	0.460	0.349	0.153	0.009	0.000
# 0 resilient WES'_3	11	90	305	488	500
Model build time (s)	13.81	7.52	4.38	2.55	1.78
Verification time (s)	1.08	0.52	0.20	0.07	0.04
Total runtime (s)	14.89	8.04	4.58	2.62	1.82
States	332259	171627	89361	47140	29387
Transitions	2171394	1090709	561534	294596	182751

by solving the corresponding MDP_W. We use an unmodified version 4.2.1 of the PRISM model checker using the explicit engine which is suitable for solving MDP_W models with a potentially very large state space, only a fraction of which is actually reachable.

For efficient generation of each workflow execution specification WES'_3 we have implemented a Python-based tool which automatically generates each WES'_3 and the MDP_W for WES'_3 in the PRISM modelling language. Each MDP_W is then automatically inputted to PRISM which in turn generates both the quantitative resiliency metrics we require, and a log of computational overheads including MDP_W state space and computation time. PRISM is executed using a standard Lenovo laptop incorporating a 2.40Ghz i7-4500U Intel processor, 8GB RAM and Windows 8 operating system. To take into account any influence the computing platform may have on the results, each analysis is repeated 20 times and the averages taken.

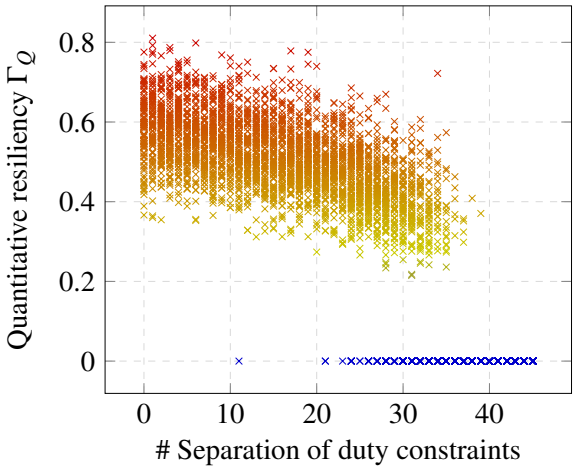


Fig. 5.1 Quantitative resiliency analysis of workflow execution specification WES_3 using PRISM, where each plot represents a set Y of WES_3 instances whose security policy contains the same number of separation of duty constraints.

Quantitative Resiliency Analysis

Table 5.1 shows the results of generating the quantitative resiliency metric for WES'_3 with each security policy $(A'_3, S'_3, B'_3) \in F$. The column **0** gives the average computational values of the 100 instances of WES'_3 coming with the security policy (A'_3, S'_3, B'_3) where $|S'_3| = 0$. The columns $\mathbf{i} \sim \mathbf{j}$ give the average computational values of the 500 instances of WES'_3 coming with the security policy (A'_3, S'_3, B'_3) where $i \leq |S'_3| \leq j$. The row (# 0 resilient WES'_3) indicates the number of WES'_3 instances appearing in each column that have 0 quantitative resiliency. We provide a graph plotting the quantitative resiliency of each WES'_3 instance against the number of separation of duty constraints k, in Figure 5.1. Each plot represents a set Y of WES'_3 instances such that for all $WES'_3 = (((T_3, <_3), U_3, (A'_3, S'_3, B'_3)), ES_3)$ and $WES''_3 = (((T_3, <_3), U_3, (A''_3, S''_3, B''_3)), ES_3)$ in Y, $\Gamma_Q(WES'_3) = \Gamma_Q(WES''_3)$ and $|S'_3| = |S''_3|$, where Γ_Q is the quantitative resiliency metric function defined in Section 2.3.4.

Figure 5.1 shows the quantitative resiliency in general, steadily reduces with an incremental introduction of separation of duty constraints. For example, when $|S'_3| = \emptyset$, each WES'_3 instance has quantitative resiliency of between 0.4 and 0.8. Instances of WES'_3 when $|S'_3| = 20$ have quantitative resiliency of between 0.3 and 0.7. When $0 \leq |S'_3| \leq 10$, all instances of WES'_3 are resilient to some degree, and, up to the point where $|S'_3| = 20$, all except 1 instance of WES'_3 have some level of quantitative resiliency. The data in Figure 5.1 and Table 5.1 also indicate 11 instances of WES'_3 have 0 quantitative resiliency when $21 \leq |S'_3| \leq 25$, and 90 instances have 0 quantitative resiliency when $26 \leq |S'_3| \leq 30$. Once $|S'_3| \geq 40$ no instance of WES'_3 is resilient, however some instances of WES'_3 have a

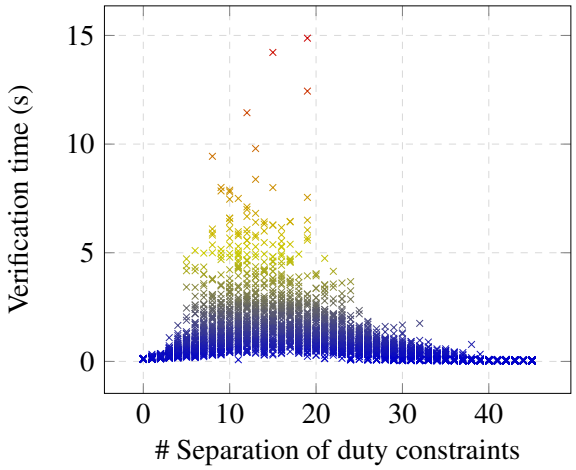


Fig. 5.2 Quantitative resiliency verification time of workflow execution specification WES_3 using PRISM, where each plot represents a set Y of WES_3 instances whose security policy contains the same number of separation of duty constraints.

level of quantitative resiliency when $\left|S_3'\right| = 39$ out of a possible 45 constraints. The results indicate that some separation of duty constraints can be added or removed with no effect on quantitative resiliency. For instance if a line was drawn on Figure 5.1 horizontally at 0.5 resiliency we can observe particular sets of constraints can be added of size 0 up until roughly 35 while a level of 0.5 quantitative resiliency is maintained.

Computation Time Analysis

Next we provide a graph plotting the time PRISM takes to verify the quantitative resiliency of each WES_3' instance against the number of separation of duty constraints k. The graph is shown in Figure 5.2. As before, each plot represents a set Y of WES_3' instances such that for all workflow execution specifications $WES_3' = (((T_3, <_3), U_3, (A_3', S_3', B_3')), ES_3)$ and $WES_3'' = (((T_3, <_3), U_3, (A_3'', S_3'', B_3'')), ES_3)$ in Y, $\Gamma_Q(WES_3') = \Gamma_Q(WES_3'')$ and $\left|S_3'\right| = \left|S_3''\right|$. The graph indicates the verification time increases in general, and then begins to decrease despite a continual incremental introduction of separation of duty constraints. The times measured are of course somewhat dependant on the efficiency of PRISM and the execution platform. The maximum average verification time is 2.24 seconds when $11 \leq \left|S_3'\right| \leq 15$. When $S_3' = \emptyset$ and $41 \leq \left|S_3'\right| \leq 45$ the average verification time is 0.11 and 0.04 seconds respectively.

The latter result can intuitively be attributed to the 0 quantitative resiliency value when all 45 constraints are applied. However, even when $26 \leq \left|S_3'\right| \leq 30$ where the quantitative resiliency of WES_3' is 0.348, the verification time is lower at 0.52 seconds than the verification

Table 5.2 Quantitative resiliency before and after a single separation of duty constraint is added to the security policy of workflow execution specification WES_4, and quantitative resiliency verification times using PRISM.

		$+(t_2,t_3)$	$+(t_2,t_5)$	$+(t_3,t_5)$	$+(t_1,t_4)$
	$\mathbf{p_4}$	$\mathbf{p_{41}}$	$\mathbf{p_{42}}$	$\mathbf{p_{43}}$	$\mathbf{p_{44}}$
Quantitative resiliency Γ_Q	0.512	0.479	0.512	0.512	0.512
Verification time (s)	0.110	0.109	0.141	0.110	0.063

time when $11 \leq \left|S'_3\right| \leq 15$. This result would indicate instances of WES'_3 are at their most complex in terms of verification time approximately when $26 \leq \left|S'_3\right| \leq 30$. By observing the number of states and transitions for the MDP_W of each WES'_3 instance, the verification time can be put into context. PRISM takes on average 2.24 seconds to verify the quantitative resiliency of an MDP_W with 600287 states and 4.75 million transitions. As one would expect, verification time appears to be tightly coupled to the size of an MDP_W model. This means to reduce verification time we must look to reduce the size of an MDP_W model without altering the quantitative resiliency. The results indicate that in some cases that separation of duty constraints can be added to, or removed from, a workflow without impacting quantitative resiliency.

5.1.2 Reducing Computation Time

In this section we provide a methodology to calculate a set of *dummy* security policy constraints (e.g., redundant separation of duty constraints or unused authorisation constraints), in order to reduce quantitative resiliency verification time. In Section 5.1.1 we showed that in some cases, separation of duty constraints could be added to, or removed from, a workflow security policy without impacting quantitative resiliency. We are not in a position to say which constraints should be removed as this may weaken the security policy. Therefore we only consider strengthening the policy, in other words adding separation of duty constraints and removing authorisation constraints, which in effect, can be removed after verifying the quantitative resiliency of a worfkflow without any loss of security.

Adding Separations of Duty

We now consider $WES_4 = (((T_4, <_4), U_4, (A_4, S_4, B_4)), ES_4)$ to be a workflow execution specification whose workflow specification $WS_4 = ((T_4, <_4), U_4, (A_4, S_4, B_4))$ and execution specification ES_4 are defined in Appendix B.2. For compactness we write p_4 for the security

Table 5.3 Quantitative resiliency before and after a single authorisation constraint is removed from the security policy of workflow execution specification WES_4, and quantitative resiliency verification times using PRISM.

		$-(t_4,u_4)$	$-(t_5,u_4)$	$-(t_4,u_2)$	$-(t_1,u_1)$
	$\mathbf{p_4}$	$\mathbf{p_{45}}$	$\mathbf{p_{46}}$	$\mathbf{p_{47}}$	$\mathbf{p_{48}}$
Quantitative resiliency Γ_Q	0.512	0.395	0.512	0.512	0.512
Verification time (s)	0.063	0.047	0.121	0.110	0.062

policy (A_4,S_4,B_4). The quantitative resiliency of WES_4 is verified as 0.512 in a time of 0.110 seconds, based on the average of 10 resiliency calculations using PRISM. We now add a new separation of duty constraint (t_2,t_3) to give a new policy $p_{41} = (A_{41},S_{41},B_{41})$ where $A_{41} = A_4$, $S_{41} = S_4 \cup \{(t_2,t_3)\}$, and $B_{41} = B_4$. The quantitative resiliency of WES_4 coming with p_{41} is now verified to be 0.479 in an average time of 0.109 seconds. In other words, the verification time has reduced by 0.001 seconds but with a loss of 0.033 quantitative resiliency.

We now consider adding alternative separation of duty constraints (t_2,t_5),(t_3,t_5) and (t_1,t_4) to p_4 to give new policies p_{42}, p_{43} and p_{44} respectively. The quantitative resiliency and average verification times are given in Table 5.2 where $+(t,t')$ denotes the addition of a separation of duty constraint to p_4, whilst $-(t,u)$ denotes the removal of an authorisation constraint from p_4. The addition of (t_2,t_5) to p_4 (p_{42}) results in no loss to quantitative resiliency but increases the average verification time by 0.031 seconds. Adding (t_3,t_5) to p_4 (p_{43}) results in no loss to quantitative resiliency nor any reduction of average verification time. However, adding (t_1,t_4) to p_4 (p_{44}) results in no loss to quantitative resiliency yet a reduction to the average verification time of 0.047 seconds.

Removing Authorisation Constraints

We now consider removing an authorisation constraint (t_4,u_4) to give a new policy $p_{45} = (A_{45},S_{45},B_{45})$ where $A_{45} = A_4 \setminus \{(t_4,u_4)\}$, $S_{45} = S_4$, and $B_{45} = B_4$. The quantitative resiliency of WES_4 coming with p_{45} is computed to be 0.395 at an average verification time of 0.047 seconds. In other words, the verification time has reduced by 0.063 seconds but with a loss of 0.117 to quantitative resiliency.

Next we consider removing alternative authorisation constraints (t_5,u_4),(t_4,u_2) and (t_1,u_1) from p_4 to give new policies p_{46}, p_{47} and p_{48} respectively. The quantitative resiliency values and average verification times are given in Table 5.3. The removal of (t_5,u_4) from p_4 (p_{46}) results in no loss to quantitative resiliency but increases the average verification time by 0.011 seconds to 0.121 seconds. Removing (t_4,u_2) from p_4 (p_{47}) results in no loss

Table 5.4 Quantitative resiliency before and after respectively adding and removing a set of separation of duty and authorisation constraints to and from the security policy of workflow execution specification WES_5, where each constraint set contains up to three constraints.

	WES_5	$+s_1$	$+s_2$	$+s_3$	$-a_1$	$-a_2$	$-a_3$
Quantitative Resiliency Γ_Q	0.640	0.634	0.628	0.622	0.625	0.605	0.580
Verification time (s)	6.534	4.285	3.557	3.295	5.098	3.833	1.765

to quantitative resiliency nor any reduction of average verification time. However, removing (t_1, u_1) from p_4 (p_{48}) results in no loss to quantitative resiliency yet reduces the average verification time by 0.048 seconds to 0.062 seconds. These results indicate that a selective addition of separation of duty constraints, or removal of authorisation constraints can reduce quantitative resiliency verification time without impacting the quantitative resiliency value.

Calculating Dummy Security Constraints

With the aid of a larger workflow execution specification WES_5, we provide a method of calculating an optimal set of *dummy* security policy constraints that minimises quantitative resiliency verification time without any reduction to the quantitative resiliency value. For clarity we calculate two optimal sets, one of redundant separation of duty constraints that can be added to the policy, and one of authorisation constraints that can be removed. Our method could easily be modified to calculate a single set of optimal dummy constraints composed of separation and binding of duty constraints, and authorisation constraints.

The workflow execution specification $WES_5 = (((T_5, <_5), U_5, (A_5, S_5, B_5)), ES_5)$ comes with 10 tasks, 5 users, and a security policy $p_5 = (A_5, S_5, B_5)$ composed of 29 authorisation constraints, 15 separation of duty constraints, and 0 binding of duty constraints. The workflow specification $WS_5 = ((T_5, <_5), U_5, (A_5, S_5, B_5))$ and execution specification ES_5 are defined in Appendix B.3. We also use the same computing platform and PRISM model checker set-up as described in Section 5.1.1. The quantitative resiliency of WES_5 is calculated as 0.640 with an average verification time of 6.53 seconds.

Dummy Separations of Duty

A Python encoded tool has been implemented which, given WES_5, calculates all possible sets of separation of duty constraints that can be added to the security policy $p_5 = (A_5, S_5, B_5)$. In the case of WES_5, the maximum number of separation of duty constraints is 45 meaning up to 30 separation of duty constraints can be added. All possible sets containing between 1 and

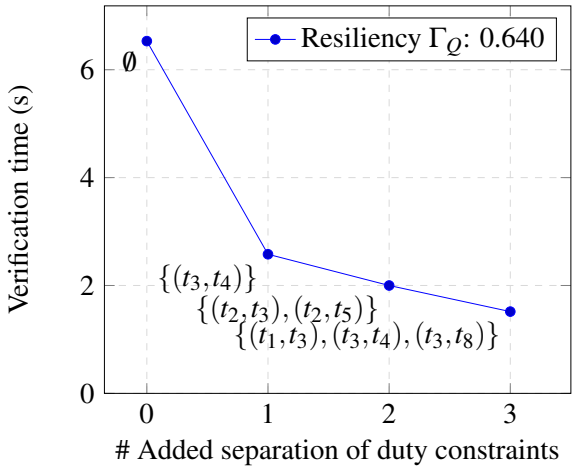


Fig. 5.3 Sets of up to three separation of duty constraints which can be added to the security policy of workflow execution specification WES_5 to optimally reduce quantitative resiliency verification time without impacting the resiliency value.

30 separation of duty constraints not appearing in S_5 are therefore computed. Each of these constraint sets are automatically added in turn by the Python tool to S_5, the resulting MDP_W is generated, and inputted to PRISM for quantitative resiliency verification. Results in terms of quantitative resiliency, verification time, and constraint set added to S_5 are logged.

The results of this analysis step are given in Table 5.4, where a column $+s_i$ gives the average quantitative resiliency and verification time for all i sized sets of separation of duty constraints added to the security policy S_5. For instance the average quantitative resiliency is 0.622 when adding any 2 separation of duty constraints not in S_5. Similarly, a column $-a_i$ gives the average quantitative resiliency and verification time for all i sized sets of authorisation constraints removed from the security policy S_5. For instance the average quantitative resiliency is 0.605 when removing any 2 authorisation constraints from S_5. For compactness we only show the impact on verification time of up to 3 additional separation of duty constraints and the removal of up to 3 authorisation constraints. In general, adding arbitrary separation of duty constraints in an incremental fashion is shown to reduce the resiliency verification time but this comes with a reduction in the quantitative resiliency value.

Finding a set of dummy constraints that reduces quantitative resiliency verification time without reducing the quantitative resiliency value is found by performing an automatic double sort on the analysis result set, first by the quantitative resiliency value (largest to smallest) and then by verification time (smallest to largest). The set of dummy separation of duty constraints that does not change the quantitative resiliency yet gives the lowest

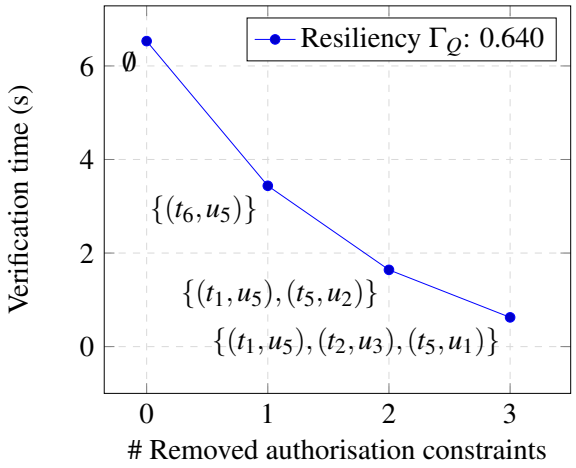


Fig. 5.4 Sets of up to three authorisation constraints which can be removed from the security policy of workflow execution specification WES_5 to optimally reduce quantitative resiliency verification time without impacting the resiliency value.

verification time for each i additional separation of duty constraints is shown in Figure 5.3. For example, adding the single constraint (t_3,t_4) minimises the verification time on average to 2.58 seconds. Adding the three separation of duty constraints $\{(t_1,t_3),(t_3,t_4),(t_3,t_8)\}$ minimises the verification time to 1.52 seconds, reducing the original verification time for WES_5 by 5.01 seconds without impacting its quantitative resiliency value.

Dummy Authorisation Constraints

Similar to adding separation of duty constraints, the results in Table 5.4 show that in general, removing arbitrary authorisation constraints in an incremental way reduces quantitative resiliency verification time but with a reduction in the quantitative resiliency value. The set of removable authorisation constraints shown to give the lowest computation time for i authorisation constraints removed from S_5 is given in Figure 5.4. For example, removing the single authorisation constraint (t_6,u_5) minimises the verification time to 3.44 seconds. Notice removing the three authorisation constraints $\{(t_1,u_5),(t_2,u_3),(t_5,u_1)\}$ minimises the verification time to 0.63 seconds, thus reducing the original computation time by 5.90 seconds without impacting the quantitative resiliency value.

5.2 Risk Acceptance

In practice, it is unlikely a security constrained workflow will be fully resilient meaning a risk of workflow failure may need to be accepted in some cases. When analysing the

impact of security policy modifications, a Chief Information Security Officer (CISO) may find the resiliency of a workflow would reduce to an unacceptable level. In cases where workflows must complete, early termination, due for instance to the unavailability of users, may bring heavy operational penalties in terms of monetary costs, lost productivity and reduced reputation. Workflows that become deadlocked due to user unavailability are typically managed by performing mitigation actions which facilitate a completable workflow, often essential in healthcare and other critical domains where workflow failure tolerance is small. For example, it may be that authorising a security override (e.g. break glass [145]) has less long-term impact to an organisation than allowing a workflow to fail. Elucidating permitted mitigation actions to manage the risk of workflow failure due to user unavailability, forms a workflow mitigation strategy. Calculating the quantitative resiliency of a workflow can aid the formation of such plans. In Chapters 2 to 4, quantitative resiliency has only been considered for workflows with sequential and parallel task ordering, that is workflow execution specifications WES coming with a task schema $(T, <)$ where all $t \in T$ must be executed according to the order defined by the partial order $<$. Calculating the quantitative resiliency for workflows of this form provides a singular comprehensible indicator of workflow failure risk for a CISO. For instance, a workflow with quantitative resiliency 0.75 has a failure risk of 0.25. Low failure risk (i.e. high quantitative resiliency) would imply an infrequent need to perform any mitigation actions. This could favour a mitigation strategy consisting of short-term, low cost actions such as a security constraint emergency override. High failure risk (i.e. low quantitative resiliency) would suggest a broader strategy including more permanent yet costly mitigation actions such as staff training and raising expected user unavailability by cancelling vacations, or paying overtime for example.

This section considers workflows with choice meaning two or more unique execution paths exist that can be taken at runtime to complete a workflow, and where each path may come with a different quantitative resiliency value. Understanding failure risk and mitigation strategy requirements of such workflows can be much more complex, especially when a workflow contains several unique execution paths. For example, an instance of an insurance claim workflow will typically take one execution path if a claim is accepted, and another path if a claim is rejected. When a claim is accepted it may take one execution path if the cost of a claim is above i for more investigation, and another path if the cost is i or less. Taking the resiliency average, or *expected resiliency* alone may be a misleading indicator of failure risk, especially when a workflow contains execution paths of both very high and very low quantitative resiliency. We therefore introduce *resiliency variance*, a new metric for workflows with choice that indicates overall resiliency variability, or volatility, from the

resiliency average. In business terms, volatility is typically viewed as a measure of risk; a variance metric helps determine the risk an investor might take on when purchasing a specific asset [45]. We show how a workflow with choice can be reduced to a set of workflows without choice which allows the techniques described in Chapters 2 to 4 to be used and show how the resiliency variance of the workflow with choice is calculated. We then discuss how resiliency variance could provide a CISO with an indicator of failure risk taken on when modifying a security policy for a workflow with choice. We then discuss how this could also aid forming a suitable workflow mitigation strategy. For example, a workflow with high expected quantitative resiliency and low variance indicates low failure risk and mitigation cost, while high variance would suggest a much higher failure risk and mitigation cost. As the focus here is on completing a workflow successfully we do not consider resiliency distance in this section.

5.2.1 Workflow with Choice

Our definition of a workflow specification $WS = ((T, <), U, (A, S, B))$ (Definition 4) does not allow for choice, that is, having exclusive task orderings according to the evaluation of some choice points. A choice point can be equated to an `if/else` statement in computer programming terms. For instance, a workflow managing a purchasing process might mean executing different tasks based on the cost of a purchase. In this section, we give an inductive definition for a workflow specification with choice inspired from [95], and show how it can be reduced into a definition compatible with Definition 4.

Task Structure with Choice

A *task structure* is built upon two sets: the set of tasks T and a set C of atomic choices. Intuitively, the latter represents the different choice points where a workflow can branch. The set TSC of task structures with choice is then defined inductively:

- Given a single task $t \in T$, t also belongs to TSC;
- Given two task structures $ts_1 \in TSC$ and $ts_2 \in TSC$, $ts_1 \rightarrow ts_2$ also belongs to TSC, and corresponds to the sequential execution of ts_1 followed by ts_2;
- Given two task structures $ts_1 \in TSC$ and $ts_2 \in TSC$, $ts_1 \wedge ts_2$ also belongs to TSC, and corresponds to the parallel ordering ts_1 and ts_2;

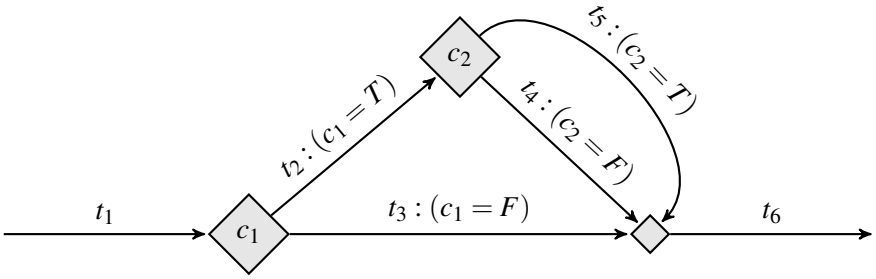


Fig. 5.5 Task structure ts_6 coming with choices of exclusive task execution, where diamond nodes c_i are choice points and the empty diamond node indicates the end of a choice.

- Given a choice $c \in C$ and two task structures $ts_1 \in TSC$ and $ts_2 \in TSC$, $c : ts_1 \,?\, ts_2$ also belongs to TSC, and corresponds to the task structure ts_1 if c evaluates to true, and to ts_2 otherwise.

To illustrate the concepts presented in this section, we define a set $T_6 = \{t_1, t_2, t_3, t_4, t_5, t_6\}$ of tasks, a set $C_6 = \{c_1, c_2\}$ of atomic choices, and a task structure ts_6 such that:

$$ts_6 = t_1 \rightarrow [c_1 : [t_2 \rightarrow [c_2 : t_5 \,?\, t_4]] \,?\, t_3] \rightarrow t_6$$

For the sake of simplicity, we do not consider any parallel composition in this example. A graphical representation of ts_6 is provided in Figure 5.5 where choice points are represented as diamond nodes. In order to represent the end of a choice, we use the empty diamond node, and in this particular example, both choices c_1 and c_2 finish at the same point. The directed arcs represent the ordering of task execution. A directed arc labelled $t_i : (c_j = T)$ indicates task t_i is executed if choice point c_j is true, and $t_i : (c_j = F)$ indicates task t_i is executed if choice point c_j is false. Three exclusive sequences of task execution exist; (t_1, t_2, t_4, t_6), (t_1, t_2, t_5, t_6), and (t_1, t_3, t_6). It is worth pointing out that in the graphical notation used in Figure 5.5, the choice nodes corresponds to or-nodes and the empty diamond node to a merge coordinator in [177].

Task Structure Reduction

At runtime, the choices in a task structure ts are resolved, and only the corresponding task sequence is executed. We adopt here an approach where it is unknown how each choice is going to be resolved at runtime, and therefore consider beforehand all possible task sequences. Intuitively, we want to reduce a task structure with choice to one without choice, for which all tasks should be executed in order to use the techniques described in in Chapters 2 to 4. Hence,

we write TS for the subset of TSC corresponding to task structures without choice, and we model the reduction process through the function $red : TSC \times \wp(C) \to TSC$, such that, given a task structure ts and a set of choices $\gamma \subseteq C$, $red(ts, \gamma)$ corresponds to the reduction of ts where each choice in γ is evaluated as true, and any other choice as false. More formally:

$$\begin{aligned} red(t, \gamma) &= t \\ red(ts_1 \to ts_2, \gamma) &= red(ts_1, \gamma) \to red(ts_2, \gamma) \\ red(ts_1 \wedge ts_2, \gamma) &= red(ts_1, \gamma) \wedge red(ts_2, \gamma) \\ red(c : ts_1 \,?\, ts_2, \gamma) &= \begin{cases} red(ts_1, \gamma) & \text{if } c \in \gamma \\ red(ts_2, \gamma) & \text{otherwise} \end{cases} \end{aligned}$$

All possible instances without choice of a task structure ts with choice is defined to be:

$$ins(ts) = \{ts' \in TS \mid \exists \gamma \subseteq C \; red(ts, \gamma) = ts'\}$$

A task structure ts without choice can be converted to a set of tasks with a partial ordering, thus allowing us to reuse the existing definition of a task schema $(T, <)$ (Definition 1). Given a task structure ts, we first write $\tau(ts)$ for the set of tasks appearing in ts (which can be straightforwardly defined by induction over ts). We then define the function $ord : TS \to \wp(T \times T)$, which, given a task structure without choice ts, returns the ordering relation $<$ over the tasks in ts.

$$\begin{aligned} ord(t) &= \emptyset \\ ord(ts_1 \wedge ts_2) &= ord(ts_1) \cup ord(ts_2) \\ ord(ts_1 \to ts_2) &= \{(t_1, t_2) \mid t_1 \in \tau(ts_1) \wedge t_2 \in \tau(ts_2)\} \cup ord(ts_1) \cup ord(ts_2) \end{aligned}$$

The possible instances of task structure ts_6 are therefore:

- $t_1 \to t_2 \to t_5 \to t_6$ (corresponding to $\gamma = \{c_1, c_2\}$)
- $t_1 \to t_2 \to t_4 \to t_6$ (corresponding to $\gamma = \{c_1\}$)
- $t_1 \to t_3 \to t_6$ (corresponding to $\gamma = \{c_2\}$ and $\gamma = \emptyset$)

Since these instances do not contain any parallel structure, the ordering for each instance is simply the total ordering of the tasks following the sequence. Therefore, given a workflow specification with choice, denoted as $WSC = (ts, U, (A, S, B))$, and a set of choices $\gamma \subseteq C$, we abuse the notation and write $red(WSC, \gamma)$ for the workflow specification $WS = (red(WSC, \gamma), U, (A', S', B'))$, where the security policy (A', S', B') corresponds to the secu-

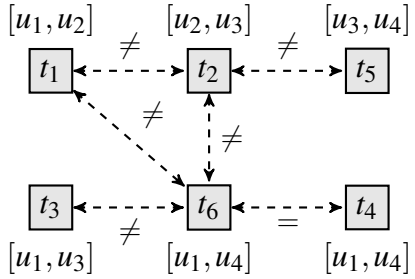


Fig. 5.6 Illustration of workflow security policy (A_6,S_6,B_6), where $\neq$ indicates a separation of duty between tasks, $=$ indicates a binding of duty between tasks, and $\{u_i,\ldots,u_n\}$ indicates the users authorised to execute a task.

rity policy (A,S,B) restricted to the tasks appearing in $red(WSC,\gamma)$. Similarly, we write $ins(WSC)$ for the set of workflow specifications WS such that there exists $\gamma \subseteq C$ satisfying $WS = red(WSC,\gamma)$.

We now consider a set of users $U_6 = \{u_1,u_2,u_3,u_4\}$ and a security policy (A_6,S_6,B_6) that defines the following sets of authorisation, separation of duty, and binding of duty constraints:

- $A_6 = \{(u_1,t_1),(u_2,t_1),(u_2,t_2),(u_3,t_2),(u_1,t_3),(u_3,t_3),(u_1,t_4),(u_4,t_4),(u_3,t_5),(u_4,t_5),(u_1,t_6),(u_4,t_6)\}$
- $S_6 = \{(t_1,t_2),(t_1,t_6),(t_2,t_5),(t_2,t_6),(t_3,t_6\}$
- $B_6 = \{(t_4,t_6)\}$

Figure 5.6 illustrates the security policy (A_6,S_6,B_6), where arcs labelled '$\neq$' and '=' signify the constraints given in S_6 and B_6 respectively. A label $[u_m,...,u_n]$ states the users that are authorised by A_6 to be assigned the execution of task t_i.

5.2.2 Resiliency with Choice

Given a workflow specification $WSC = (ts,U,(A,S,B))$, we need to assign the execution of tasks in ts to users in U in order to execute them, while respecting the security policy (A,S,B). If ts contains some choice elements, it is not strictly necessary to assign the execution of all tasks, only those that will be chosen at runtime. However, as mentioned above, we assume here that we have no control over the choices, and therefore we cannot know beforehand which subset of tasks must be assigned. Hence, we reduce the problem assigning the execution of tasks for a workflow with choice to considering the assigning the execution of tasks in all possible instances of the workflow without choice, thanks to the function *red*.

Table 5.5 User availability data sets D_1 and D_2, from which availability forecasts can be directly defined, and used to generate a set of quantitative resiliency values for the workflow specification with choice WSC_6.

	$\mathbf{D_1}$				$\mathbf{D_2}$			
	u_1	u_2	u_3	u_4	u_1	u_2	u_3	u_4
x_1	0.95	0.90	0.96	0.94	0.95	0.90	0.96	0.94
x_2	0.99	1.00	0.90	0.97	0.99	1.00	0.90	0.97
x_3	0.40	0.77	0.99	0.30	0.40	0.77	0.70	0.45
x_4	0.40	0.88	0.89	0.80	0.40	0.88	0.89	0.80

We now consider adapting the quantitative resiliency measure for a workflow without choice to a resiliency measure for a workflow with choice using the aid of an example. We consider a workflow specification with choice $WSC_6 = (ts_6, U_6, (A_6, S_6, B_6))$ such that $ins(WSC_6) = \{WS_{61}, WS_{62}, WS_{63}\}$ where $WS_{61} = ((T_{61}, <_{61}), U_{61}, (A_{61}, S_{61}, B_{61}))$ is defined to be:

- $T_{61} = \{t_1, t_2, t_5, t_6\}$ and $<_{61} = \{t_1, t_2), (t_2, t_5), (t_5, t_6)\}$
- $U_{61} = \{u_1, u_2, u_3, u_4\}$
- $A_{61} = \{(t_1, u_1), (t_1, u_2), (t_2, u_2), (t_2, u_3), (t_5, u_3), (t_5, u_4), (t_6, u_1), (t_6, u_4)\}$
- $S_{61} = \{(t_1, t_2), (t_1, t_6), (t_2, t_5), (t_2, t_6)\}$ and $B_{61} = \emptyset$

Next, workflow specification $WS_{62} = ((T_{62}, <_{62}), U_{62}, (A_{62}, S_{62}, B_{62}))$ is defined to be:

- $T_{62} = \{t_1, t_2, t_4, t_6\}$ and $<_{62} = \{t_1, t_2), (t_2, t_4), (t_4, t_6)\}$
- $U_{62} = \{u_1, u_2, u_3, u_4\}$
- $A_{62} = \{(t_1, u_1), (t_1, u_2), (t_2, u_2), (t_2, u_3), (t_4, u_1), (t_4, u_4), (t_6, u_1), (t_6, u_4)\}$
- $S_{62} = \{(t_1, t_2), (t_1, t_6), (t_2, t_6)\}$ and $B_{62} = \{t_4, t_6\}$

Finally, workflow specification $WS_{63} = ((T_{63}, <_{63}), U_{63}, (A_{63}, S_{61}, B_{63}))$ is defined to be:

- $T_{63} = \{t_1, t_3, t_6\}$ and $<_{62} = \{t_1, t_3), (t_3, t_6)\}$
- $U_{63} = \{u_1, u_2, u_3, u_4\}$
- $A_{63} = \{(t_1, u_1), (t_1, u_2), (t_3, u_1), (t_3, u_3), (t_6, u_1), (t_6, u_4)\}$
- $S_{63} = \{(t_1, t_6), (t_3, t_6)\}$ and $B_{63} = \emptyset$

We now consider two data sets D_1 and D_2 shown in Table 5.5, predicting the availability of users when executing WSC_6. We assume that D_2 is the result of some rescheduling for users u_4 and u_4. A table entry $x_i \times u_j$ is the expected probability of user u_j being available at execution step x_i. Execution specifications $ES = ((Z, \prec), \theta)$ can be extracted directly from D_1 and D_2 which are compatible with the workflow specifications WS_{61}, WS_{62}, and WS_{63}. The compatibility of a workflow and execution specification is defined in Section 2.3.2. For

Table 5.6 Quantitative resiliency values for each execution path of workflow specification with choice WSC_6 using user availability data sets D_1 and D_2, along with the expected resiliency and resiliency variance of WSC_6.

	$\mathbf{D_1}$				$\mathbf{D_2}$			
	WSC_6	WS_{61}	WS_{62}	WS_{63}	WSC_6	WS_{61}	WS_{62}	WS_{63}
Resiliency Γ_Q	0.75	0.75	0.23	0.32	0.53	0.53	0.34	0.42
$expR(w_1)$	0.43	-	-	-	0.43	-	-	-
$varR(w_1)$	0.051	-	-	-	0.006	-	-	-

instance, in the case of WS_{61} and WS_{62}, the set of workflow steps Z in a compatible execution schema will be $\{x_1, x_2, x_3, x_4, x_\perp\}$, while in the case of WS_{63}, $Z = \{x_1, x_2, x_3, x_\perp\}$.

Combining WS_{61}, WS_{62}, and WS_{63} with compatible execution schemas generates three workflow execution schemas $WES = (WS, ES)$ and allows the quantitative resiliency of each WES to be generated using the techniques described in Chapters 3 and 4. Using these techniques we automatically generate the quantitative resiliency of WS_{61}, WS_{62}, and WS_{63} shown in Table 5.6. We have created a Python-based tool which implements the function *red*, and given a workflow specification with choice WSC, generates the set of workflow specifications without choice $ins(WSC)$ before encoding the MDP_W for each in the PRISM modelling language for automatic verification. Note that WS_{61} returns the highest quantitative resiliency value of all $WS \in ins(WSC_6)$ under D_1 and D_2.

Resiliency Extrema

Finding the minimal quantitative resiliency for a workflow specification with choice WSC indicates which $WS \in ins(WSC)$ will give the lowest success rate for WSC assuming an execution specification ES. This can be interpreted as the worst case, or the instance in $ins(WSC)$ with the highest failure risk. On first glance this indicates which parts of WSC need the most attention in terms of mitigation. For instance, in our example, WSC_6 is most likely to fail when WS_{62} is executed at it gives the minimal quantitative resiliency, 0.23 and 0.34 under D_1 and D_2 respectively. Imagine now that under D_1, WS_{62} has a low probability of execution, for example, 0.01, or 1 execution in 100 instances whereas WS_{61} with 0.75 quantitative resiliency has a high execution probability, for example, 0.80, or 80 in 100 cases. In general, the resiliency for WSC_6 will therefore be much higher meaning a costly mitigating strategy for the infrequent, low resiliency case may not be cost effective.

A bound on the expected success rate can be placed on WSC by calculating both the maximal and minimal resiliency for WSC. In our example under D_1, WSC_6 has a large

bound with an expected completion rate of between 0.23 (WS_{62}) and 0.75 (WS_{61}). Under D_2, WSC_6 has a much smaller bound such that the expected completion rate is between 0.34 (WS_{62}) and 0.53 (WS_{61}). The resiliency bound can be a useful resiliency measure when all $WS \in ins(WSC)$ have an equiprobable chance of being executed. If however under D_1, WS_{61} has a low execution probability of 0.01 whilst WS_{62} has a high execution probability of 0.8 then in general the resiliency achieved will tend towards the minimal value of 0.23. Placing a bounds on the resiliency in this case becomes a misleading measure of resiliency to a CISO.

Resiliency Distribution

Given a workflow specification with choice WSC, calculating the resiliency for every possible instance $WS \in ins(WSC)$ coming with an execution specification ES provides the full resiliency distribution for WSC. This can enable the CISO to identify instances of low resiliency, and therefore those needing more extensive mitigation. A tolerance threshold for resiliency may exist for WSC, deemed acceptable when every instance WS has a resiliency equal to or more than the threshold, in other words the probability that every WS meets the threshold is 1. Assume for the sake of example, a resiliency threshold of 0.30 and for simplicity, an equiprobable execution model for all $WS \in ins(WSC_6)$ where the execution probability of WS is 0.33. A more complex probabilistic model could easily be imagined. Under D_1 the probability of WSC_6 meeting this threshold is therefore 0.66 (unacceptable) as the quantitative resiliency of WS_{61} is below this value, whilst under D_2 the probability is now 1 (acceptable) as all $WS \in ins(WSC_6)$ have a quantitative resiliency above the threshold. Illustrating a comparison of risk failure between a pre and post mitigated workflow to business leaders using resiliency distribution may be complex, especially when they multiple execution paths. It may be more useful for a CISO to provide a singular, easy to understand measure of resiliency for a workflow with choice.

Expected Resiliency

In Section 2.3.4, the quantitative resiliency function was defined to be $\Gamma_Q : \mathbb{WES} \rightarrow [0,1]$ which given a workflow execution specification $WES = (WS, ES)$ returns the quantitative resiliency of the workflow specification WS under the execution specification ES. We now consider a probability function $prob : \mathbb{WS} \rightarrow [0,1]$, which given a workflow specification without choice $WS \in ins(WSC)$, returns the probability of WS being executed. The expected resiliency indicates the likely success rate across every instance in a workflow with choice WSC, calculated as the average resiliency of all $WS \in ins(WSC)$. We define the function

$expR : \mathbb{WSC} \times \mathbb{ES} \rightarrow [0,1]$, which given a workflow with choice WSC and an execution specification ES returns the expected resiliency of WSC.

$$expR(WSC, ES) = \sum_{WS \in ins(WSC)} prob(WS)\Gamma_Q((WS, ES))$$

In our example, assuming $prob(WS) = 0.33$ for all $WS \in ins(WSC_6)$, the expected resiliency is 0.43 for WSC_6 under both D_1 and D_2, shown in Table 5.6. This in turn indicates an expected failure rate for WSC_1 of 0.57. Under D_1, with an equiprobable execution model, means the expected resiliency of 0.43 is not assured with every execution of WSC_1. Each time the instance WS_{62} is executed, the probability of WSC_1 terminating successfully is only 0.23. When executing WS_{61} the quantitative resiliency is much higher than the expected value. Clearly in this case the expected resiliency alone gives a misleading measure of resiliency for a workflow with choice, in other words the expected resiliency cannot actually be expected in every case.

Under data set D_2, the expected resiliency is now roughly attained whichever $WS \in ins(WSC_6)$ is executed. In this case the expected resiliency measure alone is arguably enough to indicate the true failure risk of WSC_6. In other words, a resiliency of ≈ 0.43 can be expected with every execution of WSC_6. This remains so even when the probabilistic execution model for all $WS \in ins(WSC_6)$ is not equally weighted.

Resiliency Variance

The resiliency variance is a measure of how spread a distribution is, or the variability from the expected resiliency of all instances in a workflow with choice WSC. A resiliency variance value of zero indicates that the resiliency of all $WS \in ins(WSC)$ are identical such that the expected resiliency alone will give a true indicator of risk failure. All resiliency variances that are non-zero will be positive. A large variance indicates that instances are far from the mean and each other in terms of resiliency, whilst a small variance indicates the opposite. The resiliency variance can give a prediction of volatility or failure risk to a workflow designer taken on when implementing a particular workflow with choice. To quantify the resiliency variance measure we define a function $varR : \mathbb{WSC} \times \mathbb{ES} \rightarrow \mathbb{R}$, which given a workflow specification with choice WSC and an execution specification ES, returns the resiliency variance of WSC.

$$varR(WSC, ES) = \sum_{WS \in ins(WSC)} prob(WS)(\Gamma_Q(WS) - expR(WSC))^2$$

The resiliency variance for WSC_6, calculated using availability data sets D_1 and D_2, from which execution specifications can be defined directly, is given in Table Table 5.6. An equiprobable execution model is again used for simplicity. Under D_1 a resiliency variance of 0.051 is calculated, equivalent to a large standard deviation of 0.23 ($\sqrt{varR(WSC_6)}$). Under D_2 the resiliency variance has been reduced to 0.006, equivalent to a much smaller standard deviation of 0.08 from the expected resiliency. Clearly this indicates in this case that all instances of WSC_6 under D_2 have a probability of terminating successfully close to the expected resiliency of 0.43.

The former case (D_1) indicates that instances in WSC_6 can have a large spread in terms of resiliency despite having the same expected resiliency as the latter case (D_2) coming with a small spread, or variance. Under D_1, the results show that instances exist in WSC_6 with much lower and higher probabilities of terminating successfully than the expected resiliency for WSC_6. The workflow specification with choice WSC_1 can be considered volatile or high risk as it has a high risk of failing if one such instance with low quantitative resiliency is executed. Coupled with expected resiliency, resiliency variance can provide an easy to understand measure of workflow risk failure and allow CISOs to quickly compare similar complex workflows, before and after a policy modification, to help them predict a suitable mitigation strategy.

5.2.3 Mitigation Strategy

In this section we give an overview of the main techniques discussed in the literature that could be implemented within a workflow mitigation strategy to overcome situations when a workflow deadlocks. These mitigation actions are categorised into two classes, long-term actions and emergency actions.

Long-term Actions

Long-term actions can help raise the resiliency of a workflow by providing a secure solution that does not involve having to violate the security policy or change the task structure [149, 151]. Long-term actions can also often provide a more permanent solution to parts of a workflow that commonly becomes blocked. Long-term actions arguably take time and can be expensive in monetary terms to complete, yet the long-term benefits can be high. Those actions of interest include:

- **suspension** : a workflow is suspended until a user becomes available. This can be appropriate if deadlines are not important or there is some assurance of future

availability. Essentially a task is assigned to a user and executed when the user becomes available.

- **escalation**: the probability of a valid user being available for a task is increased. A user may be asked to return from vacation or come in on their day off, or they may need to suspend another task they are currently executing.

- **training** : a user's capabilities are raised to an acceptable level before authorising their execution of a task.

- **change policy** [11, 12] : security constraints are removed or modified (e.g. reallocating roles) which can take time and may need to be done multiple times if a workflow is to complete. Changes may not be possible due to legal requirements or impractical if users do not have the correct skills.

Emergency Actions

Emergency actions can help raise the quantitative resiliency of a workflow by overriding the security policy or changing the task structure. Such actions provide a quick-fix to a workflow that becomes blocked but do not offer any permanent solution to parts of a workflow that commonly becomes deadlocked. A less secure solution is provided than long-term actions that may also impact the output quality of the workflow if the task structure is indeed changed. Emergency actions are arguably quick and cheap in monetary terms to complete, yet the long-term benefits can be low. A distinction is made between overriding which implies some control is in place over who and how policies can be broken while violation is unsolicited. Those actions of interest include:

- **delegation** [55, 67, 102]: if user is unavailable they may delegate a task assignment to a peer or subordinate who would not normally be authorised to perform the task. This overrides the authorisation constraints but can result in lower standards and higher risk.

- **break glass** [120]: certain users are given the right to override a security constraint to gain privileges when the assigned user is unavailable, set up with special accounts. Justification is typically sought after access is granted.

- **skipping**: a task is bypassed and executed at a later time, although out of sequence. This is similar to suspension although other tasks are executed while waiting for a user to become available.

- **forward execution**: the workflow instance is rolled back [53] until another execution path can be taken which bypasses the deadlocked state.

Strategy Selection

Implementing a suitable mitigation strategy is important to reduce a workflow's chance of failure, especially one with both a high expected success rate and rigid security constraints. Ultimately a favourable mitigation strategy will give a high expected resiliency and a low resiliency variance. Clearly we are not in a position to state which and when particular mitigation actions should be implemented as part of a mitigation strategy as this is highly context dependant. We do however offer some discussion on this matter and show how the resiliency measures for a workflow with choice discussed in Section 5.2.2 could be useful in this regard. It may be the case that a mitigation strategy can consist of only long-term actions, especially where security is paramount and no emergency actions are permitted. Alternatively, finishing a workflow in a timely manner may be the priority meaning a mitigation strategy consists of only emergency actions. A third option is a mitigation strategy consisting of both long-term and emergency actions that is fully comprehensive and means the most appropriate option is always available.

Although long-term mitigation actions can be costly in both time and monetary terms, it may be the case that such actions need only be performed once. For instance, training a staff member once for a particular task means they can perform the task in all future executions when necessary. Implementing long-term mitigation actions for all instances of low resiliency would seem a sensible option however if some or all low resiliency instances have a very low probability of execution, this approach may not be cost effective. Emergency actions alone may be acceptable. If on the other hand emergency actions are implemented for an instance with a high probability of execution yet low resiliency it is likely that these often less secure actions will need to be performed multiple times. Long-term actions may be more appropriate here. Using the minimum resiliency of a workflow with choice may lead to over mitigation, especially if the lowest resiliency instances are infrequently executed. Using the maximum resiliency may produce the opposite effect such that a workflow is under mitigated. Workflows with high resiliency variance and low resiliency variance can have the same measure of expected resiliency meaning this measure alone may be misleading. The expected resiliency and resiliency variance together can inform mitigation strategy choice as follows:

- **high resiliency and high variance** : a combination of both action types with a higher proportion of emergency actions
- **low resiliency and high variance** : a combination of both action types with a higher proportion of long-term actions
- **high resiliency and low variance** : emergency actions
- **low resiliency and low variance** : long-term actions

5.3 Summary

In some cases, the quantitative resiliency of a workflow may fall below an acceptable level of risk when analysing the impact of security policy modifications. This can be of particular concern when policy modifications are enforced by regulation updates, especially in critical workflow domains such as healthcare that have a small tolerance for workflow deadlock or failure. We have outlined two techniques showing how calculating quantitative resiliency can help manage the risk of workflow failure caused by users becoming unavailable. The first technique is aligned with reducing failure risk and has considered the computation of quantitative resiliency at runtime to ensure the assignment of task executions to users maximises the quantitative resiliency of the workflow in accordance with the current prediction of user availability. We have shown that adding or removing security constraints to the security policy of a workflow has a clear impact on the resiliency computation time, which can increase or decrease respectively. We have then proposed an approach for a Chief Information Security Officer (CISO) to analyse and modify a security policy to improve the computation time for quantitative resiliency at runtime by adding a set of dummy, or artificial security constraints that do not impact the resiliency value. Our observations show the gain in time can be significant, for instance in our example the computation time reduces from 6.53 seconds to 0.63 seconds.

The second technique is aligned with failure risk acceptance and has considered the formation of mitigation strategies, in particular calculating resiliency metrics to help form strategies for workflows with choice that can have a different resiliency value for each possible execution path. We have shown that taking the resiliency average, or expected resiliency alone can be a misleading indicator of workflow failure risk, especially when a workflow contains paths of both very high and very low quantitative resiliency. We have introduced a new metric for workflows with choice called *resiliency variance* that indicates

overall resiliency variability, or volatility, from the resiliency average. We have shown how a workflow with choice can be reduced to set of workflows without choice whose quantitative resiliency can be generated using the techniques described in Chapters 3 and 4. We then showed how resiliency variance is calculated before discussing how resiliency variance can provide a CISO with an indicator of the workflow failure risk taken on when modifying a security policy for a workflow with choice. Discussion has been given on how resiliency variance could be useful for predicting a suitable mitigation strategy containing actions that should be taken to avoid a workflow failing due to user unavailability. For instance, a workflow with high expected resiliency and low resiliency variance would indicate a low risk of failure and a mitigation strategy containing low-cost emergency style mitigation actions.

Part II

Ontologies

Chapter 6

Security Ontology

Part 1 of this thesis considered quantitative techniques for analysing the potential impact that security policy modifications have on workflow completion. These techniques provided metrics for analysing workflow security at the process level by calculating a workflow's resiliency, that is the probability of completion, under the uncertainty of user availability, without violating security constraints. This analysis considered tasks to be abstract such that if the execution of a task could be assigned to a user, that task would be completed. In practice, workflow tasks are often complex and come with many different security policies regulating system access, password usage, network access, usb stick usage, and many other concepts necessary to complete tasks. The second part of this thesis therefore considers the impact security policy modifications may have at the task level. Due to the complexities of many workflow tasks we consider facilitating qualitative security analysis, using the knowledge and experience of security and human behavioural experts.

This chapter provides the foundations for two bespoke tools we have created for information security and human factors experts to easily record and incorporate their knowledge within the formal structure of a security ontology. An ontology is a description of concepts, their properties, and their interrelationships that exist within a particular domain. We suggest bespoke ontology development tools are needed for knowledge holders to construct an ontology first-hand, and thereby build a shared and agreed knowledge base of information security and its relationships to human behaviour in the workplace. The resulting knowledge base could be used by Chief Information Security Officers (CISOs) to analyse the potential impact modifying a security policy has on workflow completion, and the identification of behavioural controls to manage that impact. The chapter begins by describing the management of organisational knowledge and how it can be formalised in an ontology to form a knowledge base. It then introduces the knowledge stakeholders whose knowledge would be

incorporated into a security ontology, and a brief overview of human factors related to an information security setting. It then describes current ontology development tools before discussing current information security ontologies appearing in the literature. The security ontology which forms the underlying knowledge base structure for the tools we introduce is outlined, before presenting the results of consultations carried out with two CISOs from which we have extracted requirements for two security ontology development tools.

6.1 Knowledge Management

The second part of this thesis focuses on the provision of tools which allow security and human behavioural experts to record and incorporate their knowledge within the structure of a security ontology. Formalising security related knowledge would create an agreed security knowledge base whose content could be used by a Chief Information Security Officer (CISO) to conduct qualitative analysis on the potential impact security policy modifications have on workflow completion. As a first step to creating bespoke security ontology development tools we describe the management of knowledge and how it can be formalised in an ontology.

6.1.1 Organisational Knowledge

Knowledge is a major organisational resource [89]. It can be considered as comprising the experience and insights embodied in individuals, or organisational procedures and processes. Its key components are concepts (e.g. events or objects) and the relationships between those concepts. Davenport and Prusak describe knowledge as:

> "A fluid mix of framed experiences, values, contextual information, and expert insight that provides a framework for evaluating and incorporating new experiences and information. It originates and is applied in the minds of knowers. In organisations, it often becomes embedded not only in documents or repositories but also in organisational routines, processes, practices, and norms." [46]

Knowledge can potentially reside in the minds of experts or within the external sources they use such as organisation statistics (business intelligence), experimental research data or online materials. The capturing and sharing of distributed knowledge can potentially bring vast improvement to an organisations's performance if that knowledge is sound, meaningful and actionable [44]. Organisations will acquire insight into how to adapt to changing environments, avoid errors and solve problems more efficiently. By recording this knowledge

in the form of a knowledge base, organisations will gain the valuable experience and insights of experts and retain it for future use. Knowledge recording must allow experts to capture their knowledge directly and structure that knowledge in a useful and meaningful way.

For knowledge to be valuable it must be correct in a way that allows organisations to manage current situations efficiently and plan effectively for their future [84]. Correct knowledge can be considered to be the consensual knowledge of the contributing experts. To acquire correct knowledge the method of capturing and recording expert knowledge must include a mechanism for reaching consensus. The discipline of knowledge management deals with the management of organisational knowledge and encompasses its capture, recording and reuse. Defined by Skyrme:

> "Knowledge management is the explicit and systematic management of vital knowledge and its associated processes of creating, gathering, organising, diffusion, use and exploitation. It requires turning personal knowledge into corporate knowledge that can be widely shared throughout an organisation and appropriately applied." [157]

6.1.2 Formalising Knowledge

To construct a knowledge base and present its content in a useful and meaningful way that content must be structured and formalised. One method is to record and incorporate knowledge within the structure of an ontology [71, 72]. An ontology is an interrelated set of terms and concepts that are used to describe and model a particular domain of interest. Ontologies present a common understanding of a domain through formal descriptions of its concepts and relationships. This reduces ambiguity and supports consistent treatment of an ontology's content. Providing explicit semantics to ontology data allows its interpretation and reasoning by both humans and software. Explicit semantics also support the sharing, use and reuse of ontology content. Through sharing a common understanding of information and knowledge, humans and systems can communicate and interoperate effectively within a particular domain.

For an ontology to assign meaning to data it is necessary to define information via representational building blocks and the relationships within and between those blocks. These building blocks are 'classes', 'individuals' and 'properties' [135]. Classes describe concepts in the domain with each class representing a group of instances, or individuals with similar characteristics. Each individual is an instance of a particular class. Classes themselves may have sub-classes whose individuals are more specific than those of the

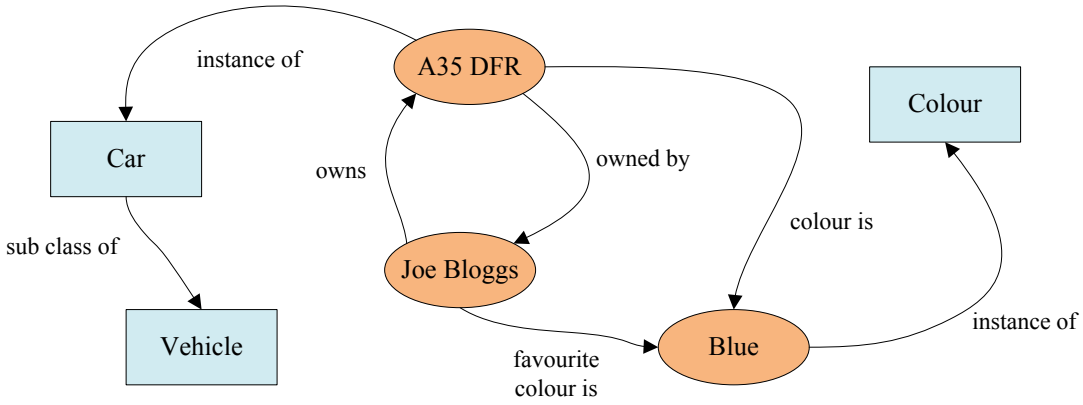


Fig. 6.1 Partial ontology example where rectangles are ontology classes, ovals are instances of a class, and directed arcs are relationships between instances and classes.

super-class. Properties provide attributes and values to individuals by defining relationships between those individuals. Restrictions on a property's domain and range can be stated to restrict which properties an individual can or must have. An ontology is created by defining classes (arranged in a hierarchal structure) and properties [135]. A knowledge base can be considered to be an instance of a particular ontology that contains individual instances of its classes, related by its given properties.

Figure 6.1 shows a simple ontology example which illustrates how an ontology is structured and its concepts related. Each individual (ovals) is an instance of a class (boxes), for instance *Blue* is an instance of the *Colour* class and *Joe Bloggs* could be an instance of a *Person* class. The *Car* class is a sub-class of the *Vehicle* super class, identified by the directional arrow between the two classes. Only one individual of the *Car* class is shown in this example, defined as the car's unique number plate *A35 DFR*, but typically the *Car* class will have multiple individuals. Many of the property relationships (arrows) would typically have a range and domain restriction. For example, the relationship *owns* would be a many-to-many mapping as every car can potentially have multiple owners and a person can own multiple cars.

6.1.3 Knowledge Stakeholders

We assume a scenario where CISOs need formalised knowledge of human factors in information security to analyse the impact of changing a security policy and identify behavioural controls to manage that impact. It is then necessary to obtain this knowledge and record it within an ontology. CISOs may also propose additional content for inclusion in the structure

of a foundation ontology with regard to how information security issues are represented (see Section 6.3.2). It is assumed CISOs have knowledge of information security issues and some personal management experience of human behavioural factors, but no knowledge of ontology construction.

Human factors researchers, in the context of this work, investigate human-behavioural factors in an information security environment. They may, directly or indirectly, also examine how an organisation's information security policies affect employee behaviour. A human factors researcher would analyse the content of the security ontology for research purposes, and potentially submit refined or additional content concerning human-behavioural factors. This content is then used by a CISO while analysing the impact of changing a policy. It is assumed that human factors researchers have knowledge of human-behavioural factors in information security, and a limited knowledge of security management issues, but no knowledge of ontology construction.

Ontology experts (IT technicians and ontology researchers) would assist CISOs and Human Factors Researchers by constructing, modifying and maintaining a security ontology. They take the content given by a CISO or human factors researcher (i.e. knowledge or procedural details) and enter it into the ontology. The ontology content may then be formatted for presentation during the information security management process, or used by ontology experts in further (manual or automated) processing. An ontology expert can also make changes to the ontology structure on the instruction of a CISO or human factors researcher. It is assumed an ontology expert has knowledge of ontology construction and tools but little knowledge of information security issues and/or human-behavioural factors. As such they are able to maintain the ontology but do not have the capacity to reliably populate it. For these reasons CISOs, human factors researchers and ontology experts must be provided with a shared tool that facilitates the direct capture and re-use of information security knowledge.

6.1.4 Human Factors Knowledge

Workflow tasks are commonly performed by humans [69], whose behaviour when processing information is constrained by information security policies. Information security is often viewed as an added burden by users who must follow security procedures and abide by security rules in order to carry out their assigned tasks [10]. The way users react and behave when following these security protocols can affect task completion times and workflow success rates. For instance a user forgetting their password, or not having the correct permissions to access a critical piece of information can slow task completion. Users may of

course simply exhibit accidental, non-malicious and unpreventable behaviour in reaction to security policies [126, 153]. To the CISO, security controls that may seem straightforward and obvious can be challenging to those users who are not technically minded, even in the case of the simplest security procedures. Security can therefore be viewed as too complex, unusable and the cause of 'unnecessary' delays, for instance a complex security policy may necessitate regular resets of a user's password when too many unsuccessful attempts have been made [142]. Users may adopt a blasé attitude to information security (i.e. doesn't apply to them), they may not grasp its importance, they may be oblivious to it, or they may feel they have no choice but to ignore it to complete their assigned tasks [134]. Security work arounds are a common symptom of misaligned security controls such as missing permissions to access needed information. Bypassing a security policy puts information at risk which ultimately can affect the success rate of a workflow, for instance information may be left open to unauthorised access or modification [87]. Another common reaction to security is the unauthorised use of instant computing resources (e.g., the cloud) and the latest technologies 'needed' by users to manage their ever increasing workloads and meet their targets. Organisations are often accused of being slow in assessing and authorising these technologies meaning users often go ahead and use them anyway [119]. These various impacts on human behaviour may not be fully understood by a CISO, nor behavioural controls that can manage this impact when changing a workflow security policy.

It is therefore important a CISO considers human behavioural factors when analysing the potential impact of modifications to a security policy on workflow success rate. This impact may be unmeasurable in many cases but may be observed and studied which can over time tease out the typical behaviour or norms of users when interacting with particular security procedures and rules. It is these norms and their alignment with information security that need to be captured and formalised in an ontology using tailored development tools to enable meaningful policy impact analysis. Social norms are unwritten rules about how humans behave [164]. Behaviour which follows these norms is called conformity, and most of the time norms provide a useful way to understand and predict what humans will do in particular situations. Humans conform to social norms because it provides order, guidance and understanding within an environment, making them often necessary for humans to operate [125]. Norms defining appropriate behaviour exist for every environment and have been observed to change when humans move from one environment to another [164].

A large amount of human factors and behavioural research in the information security domain appears in the literature. Notably, Bartsch and Sasse in [10] present a study they conducted within a large organisation, of 118 user experiences with information access

controls and their subsequent (self-reported) behaviour. The study highlighted the fact that user issues with security policies are reported as anecdotes and organisations, more so CISOs are often reluctant to carry out comprehensive reviews of the problems, nor analyse the impact of security on productivity. From the user responses the authors were able to develop Personas which represent groups of users in terms of behavioural norms and their daily problems which are to a large extent caused by policy modifications. Similar user studies have been conducted to establish user experiences with security policies and subsequent user behaviours, specifically in relation to information sharing [184] and access control [14, 144]. In [96], Kirlappos et al. identify a security behaviour called shadow security where users bypass organisational security policies in order to complete tasks yet form their own security measures to counteract the risks they may cause. This in essence reflects the working compromise user find between security and 'getting the job done'. A number of shadow security practices are described following a study conducted with users within an organisation. Work has been conducted on characterising and formalising user security behaviours within a knowledge base. Alfawaz et al. in [2] have conducted a number of case studies and an extensive literature review to characterise user behaviour towards information security controls. From their findings they suggest how user behaviour can change over time due to influential factors such as technology, the social environment, and self-interest which would suggest an ontology of information security and human factors knowledge would need continual review. Stanton et al. define a taxonomy of user security behaviour ranging from malicious behaviour meaning a user will intentionally violate a policy to beneficial behaviour meaning a user is strongly security minded [159]. The authors taxonomy was formed following 110 individual user interviews and a survey of over 1000 users self-reporting behaviour relating to passwords.

6.2 Ontology Development

6.2.1 Using Development Tools

Currently the construction and modification of an ontology aligning information security and human behavioural factors would require the use of an existing ontology development tool. Ontologies are typically expressed in such tools through formal XML based ontology languages which assign machine-understandable semantics to the user submitted content. Common examples are the well supported Web Ontology Language (OWL) [170] and the Darpa Agent Markup Language plus Ontology Interface Layer (DAML+OIL) [82]. The tools

themselves typically come in two forms, graphical and text based. Graphical tools allow the user to enter their content by constructing an ontology diagram using a palette of shapes and connectors provided by the tool (e.g. SemTalk [64] and GrOWL [101]). Alternatively, textual editors (e.g. OntoWiki [7] and Protégé [133]) permit content to be entered manually and arranged in a hierarchal structure. Both forms of editor automatically convert the informal content into a formal ontology and store it as an ontology file that can be analysed using the same, or another language supporting tool. Although different in form, both types of development tool require the same information from the user. Assuming the ontology content has been gathered, the user must define its overall structure in the tool being used. Concept types (classes) and relationship types (properties), including any range or domain restrictions must be created by the user before the content itself can be inputted. As the content is entered, each concept (individual) must be separately defined by stating its content, class type, properties and relationships to other concepts.

Due to its complex nature this process assumes familiarity with ontology technologies. To create or modify an information security ontology, knowledge is required of ontology creation, including the use of ontology development tools; ontology structure and language; and the ontology content itself. Most CISOs and human factors researchers do not have the expertise to create an ontology directly using existing tools as such tools are complex and aimed at ontology experts, not those who hold the knowledge to populate those ontologies. This of course holds true for many domains of interest, however this thesis focuses on the information security domain and the provision of bespoke ontology development tools for security experts. As such an information security domain expert may be unable to develop ontology content themselves, and would require either the assistance of an ontology expert or a dedicated ontology development tool that hides ontology complexity. Due to the potential lack of expertise in ontology creation an information security domain expert may either convey their knowledge to an ontology expert, or interact with an ontology expert directly within a shared ontology development scenario. The ontology expert will proceed to enter the content into an ontology development tool. However, manual entry can be error prone especially if the ontology expert does not fully understand the nuances of the content and its structure as they enter it into the ontology. Problems still exist if it is assumed that the ontology expert is familiar with the structure of the ontology, and that the ontology structure is adequately defined before entry of knowledge data. The process can prove time-consuming as there is a need to ensure the availability of the ontology expert and relevant domain experts. There is also a need to ensure that each individual concept is unambiguously

defined within the constructs of the chosen ontology development tool. This may slow the knowledge-capturing process.

6.2.2 Collaborative Ontology Development

For an ontology of information security and human behavioural factors knowledge to be a comprehensive and correct its development process should allow knowledge contributors to cooperate and reach consensus [162]. The creation of an information security ontology is far too large a project for any one individual or small team to carry out effectively. Development by such people also restricts the contributed knowledge which may be specialised, out of date or culturally specific leading to a restricted and stagnant knowledge base. To solve this, the ontology must be constructed by multiple experts to provide a larger and more accurate repository of information security knowledge. There are potentially vast amounts of information security knowledge held globally so collaboration must be an integral part of the ontology's development [142]. Collaboration will allow CISOs and human factors researchers to capture, integrate, publish and share their knowledge with peers and colleagues within the information security domain. Through collaboration these domain experts can potentially comment, criticise and peer review other content with the ultimate aim of reaching consensus. Through the involvement of multiple experts, a larger and more accurate knowledge base can be created. The collaboration process must include mechanisms for both synchronous and asynchronous development and communication. This will allow valuable contribution from all experts regardless of location or time-zone. Furthermore, a deeper discussion of information security is promoted between domain experts leading to an accepted base of knowledge for the whole community.

For an ontology of information security knowledge to be useful the knowledge it contains must be accepted and trusted by both its providers and users. Different terminology can potentially be used for the same concept or the same terminology for different concepts. Incorporating a consensus mechanism in the knowledge base development process means every term will be clear and unambiguous to all parties involved. Agreement is realised when every term has the same meaning to all domain experts allowing effective communication amongst themselves through their shared vocabulary. Consensus can be considered a form of collaborative decision making in which all contributors have an equal say when discussing and accepting a concept's meaning. Technology can allow CISOs and human factors experts to successfully share in the creation of a knowledge base. The Web is a natural platform for collaboration and knowledge sharing by distributing the development process and the

resulting knowledge base to the entire information security community. Knowledge holders will have on demand access to the latest version of the knowledge base anywhere in the world. Web hosting also eliminates the time and cost associated with deploying client side applications and the need to upgrade them.

6.2.3 Existing Development Tools

Collaborative ontology development tools allow the successful capture and integration of ontology content from a wide variety of sources which is too much for one single person to process. A number of tools are already available, for example Web-Protégé [172] which, based in a Web environment, allows simultaneous editing on the same ontology file and for users to see those changes immediately. The ontologies themselves are listed in the tool's interface and are available for any registered user to view and edit. Web-Protégé offers form based content entry while presenting that content and its structure in a textual format. Ontology content is organised into class, property and individual hierarchies displayed with other structural data in separate tabbed panes whose level of complexity is appropriate for an ontology expert Although the integration of widespread knowledge is simplified with Web-Protégé, an understanding of ontology construction and language is still required. OntoWiki [7] is a collaborative Web application for the development of ontologies that serves to acquire the knowledge of users while in effect hiding the actual ontology development. OntoWiki is similar to existing Wiki systems and regards ontologies as information maps, built from nodes which represent concepts in a visual and intuitive way. All content supplied by registered users can be commented on, rated, and its popularity and provenance viewed. This tool to decrease the entrance barrier for domain experts to capture their knowledge and collaboratively develop ontologies however is not tailored specifically to information security domain experts. COE (Collaborative Ontology Environment) [75] is a further collaborative tool that attempts to abstract away the ontology development by building on the rapid construction capabilities of CmapTools [27] and its concept mapping system to represent domain knowledge. Concept maps however are meant for communication to humans and not machine readable. This application translates concept maps to the machine readable OWL language. COE combines an OWL ontology viewing and editing environment which displays ontologies as concept maps. Related concepts may also be located in any Web based ontologies and incorporated into the ontology being developed allowing knowledge from wide spread sources to be captured. Even though OntoWiki and COE are aimed more at domain experts rather than ontology creators they still remain relatively complex, require a

certain amount of initial training; and are generic in nature and not designed specifically for security ontology creation unlike our proposed tool.

Visualisation of an ontology during its construction or modification is of great advantage to the user and eases these processes immensely. There are a number of visual ontology creation tools using OWL as a base language, for instance GrOWL [101] and OWL-S Editor [57] which illustrate the ontology in a UML format. SemTalk [64] uses the functionality of Microsoft's graphical application Visio's to create and modify ontologies graphically, again in a UML format, translating ontologies automatically to an OWL ontology file. SemTalk still remains complex however and is again aimed at the ontology creator, not the domain expert and is designed for ontology creation covering any domain. In general we make the observation that currently available ontology development tools are relatively complex, are generic in nature and require a substantial amount of initial training and configuration before the knowledge capture process can begin.

6.3 Information Security Ontology

The diverse knowledge relating to information security issues and human behavioural factors can be organised and related in the form of a knowledge base by using an ontology. Such an ontology will allow clear and effective communication within the security community and inform (and thereby improve) a CISO's ability to analyse and communicate the predicted impact of changing a security policy in terms of typical user behaviours, and identify controls that can be used to manage those behaviours. Some work already exists in the literature in the field of security ontologies. In [147], Raskin et al., highlight the suitability of using ontologies to describe the information security domain. Donner in [50], discusses the need for an ontology to describe the most important security concepts and their interrelationships. The author suggests such an ontology is needed to provide explicit meaning to the current, vaguely defined terminology and allow clear and effective communication between colleagues and their clients. Vulnerabilities to information security systems and the information itself would be classified allowing the detection of possible threats and their countermeasures. The discussion ends with the proposal that the ontology should be developed in a collaborative manner in order to create a robust body of information security knowledge.

6.3.1 Current Security Ontologies

The capture of security knowledge in an ontology has been shown to be viable through a number of studies. A systematic survey and comparison of approaches applying the concept of ontology to reuse, communicate and share information security knowledge has been carried out by Blanco et al., in [19] and Souag et al., in [158]. In the former, the main proposals in some 28 different works have been identified and compared in terms of security concepts and their relationships. The authors suggest that in general the ontologies analysed do not define all possibilities of the information security domain, but instead focus on specific areas of it. They also suggest that a limited number of security concepts can be defined and that many, although common to each ontology are defined in different ways. This would suggest a more collaborative approach is needed within a community in order to remove ambiguity from ontology construction.

In [78], Herzog et al., suggest an ontology of information security needs to contain four main concepts: 'assets', 'threats', 'vulnerabilities' and 'countermeasures'. An example ontology has been developed in OWL consisting of some 88 threats, 79 assets, and 133 countermeasures. This ontology structure is extended by Fenz et al., in [62] who incorporate the ISO27001 guideline [21] with a security ontology that considers the physical aspects of information security management. Also proposed here is the 'Ontoworks' framework to access, visualise and reason about the content of the ontology. The main purpose of their work is to allow organisations to audit security polices and assess whether they adhere to the ISO27001 guidelines. Similarly, Ekelhart et al., in [61] incorporate domain knowledge, organisation assets and the German IT Grundschutz Manual guidelines [60] into a security ontology. The purpose of this work is to allow organisations to determine the set of controls they need to put in place to obtain certification for specific information security management standards, however it does not consider controls suitable to manage the impact of human behaviour when interacting with security controls.

Human behavioural aspects in security have been considered by Obrst et al., who define a systematic approach for developing an ontology of the cyber security domain expressed in OWL [137]. This work focuses primarily on describing threats and impacts on IT infrastructure by malware but includes human-oriented concepts such as the human users interacting with affected systems, the actors infecting those systems with malware, and the actor's capabilities to infect those systems. The ontology can be used to extract infrastructure vulnerabilities and detect malware attacks and controls to mitigate those attacks. In terms of human behavioural factors the focus is on the attacker trying to maliciously infect a system rather than the users and how they behave when interacting with common security controls.

Schiavone et al., in [155] suggest implementing workable security polices requires the whole organisation to be described as a complex system within an ontology. The ontology suggested incorporates concepts describing an organisation at the business level such as 'capabilities', 'mission', 'governance', 'resources', and 'values'. By aligning these with information security concepts and introducing business metrics, the ontology can be analysed in terms of how best to configure complaint information security policies and controls yet minimise the impact on various business activities. Information security concepts such as 'threats', 'vulnerabilities', and 'countermeasures' are gained from externally sourced security ontologies suggesting the need for these to be constructed by security domain experts.

A more quantitative approach to information security using ontologies has been suggested by Singhal and Wijesekera in [156]. The authors show how an information security ontology with a similar structure to the one proposed by Herzog et al., [78], can be extended to model security metrics such as costs and benefits of particular security controls. The ontology is constructed in OWL within the Protégé ontology development tool [133]. The main motivation for this work is to query the ontology and produce reports on information security spending. In [54], Ekelhart et al., consider a security ontology for low-cost risk management and threat analysis. The ontology is used to evaluate the effectiveness and the cost/benefit ratio of individual security constraints. A tool 'SecOntManager' has been implemented that uses the ontology content to visualise simulated security threats and their impact on business processes in terms of cost and recovery time. This approach could be adapted to enable CISOs to analyse the impact of changing security polices on user behaviour, and how that behaviour goes on to impact the success rate of a workflow. The main limitation however would be generating suitable metrics as user behaviour at the workflow task level is extremely difficult to quantify, suggesting a more qualitative analysis approach is more appropriate.

6.3.2 Foundation Security Ontology

A foundation information security ontology to enable analysis of policy modifications on workflow success rate should define the most important security issues and concepts ('assets', 'vulnerabilities', 'threats', etc.) and the relationships between them. It should also crucially provide a standard model of human behavioural factors and how they relate to the concepts within an enterprise's security policies. For these purposes we use the ontology developed by Parkin, Van Moorsel and Coles [142]. The concepts represented in the ontology are shown in Figure 6.2. Each individual concept has a relationship with one or more other concepts. The objects Chapter, Section, Guideline and Guideline Step represent content from the ISO27002

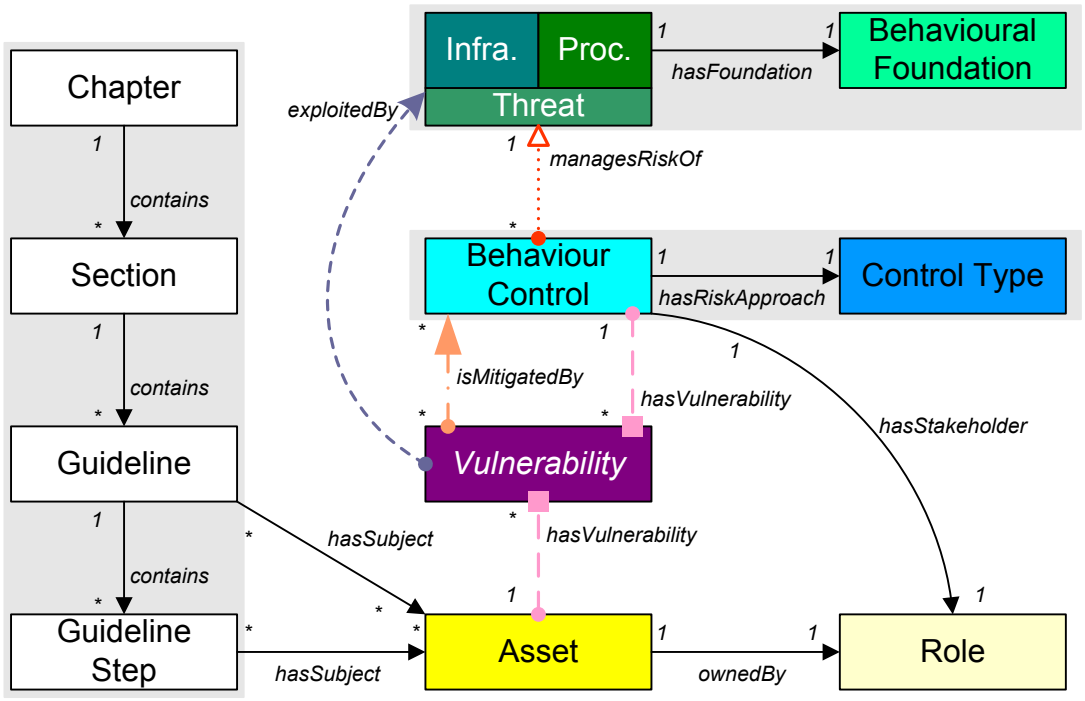


Fig. 6.2 Overview of security ontology which incorporates information security and human behavioural knowledge with the ISO27002 security standard.

standard [22]. An individual Guideline can be associated with a particular information Asset by way of the `hasSubject` relationship. Otherwise if a Guideline has been broken down into more refined Guideline Steps it will be these that are linked to an Asset, e.g., a password. We represent those information Assets identified in a Guideline or Guideline Step that either must be secured or which are crucial to an information security management process. In the ontology an Asset can be `ownedBy` someone that has an identified Role, who is then responsible for its maintenance. For instance a password is owned by a user.

The ontology also represents the security and usability weaknesses of an Asset that may promote or inhibit certain user behaviours. It is with the Vulnerability concept that these human behavioural factors are introduced into the ontology. A Vulnerability may be `exploitedBy` a Threat (e.g., if *"memorisation of password is difficult"* it may follow that *"password is forgotten"*), which renders the Asset unusable or insecure. Note that when a Vulnerability is 'exploited', this may be intentional or accidental. A Threat may be either an Infrastructure Threat or a Procedural Threat. The former represent activities that directly affect security mechanisms, whereas the latter represent security events that impact upon an individual and their behaviour. A Procedural Threat may impact the success rate of a workflow, for instance if an employee forgets a password and is unable to access a system

until it is changed. It is with the Procedural Threat concept that the impact of policy modifications can be introduced into the ontology. For each Procedural Threat we record the Behavioural Foundation, as a means to classify behaviours and indicate the concerns that they raise within an enterprise (e.g. a person's memory capabilities or attitude towards security).

A Vulnerability may be `mitigatedBy` a Behavioural Control. A Behavioural Control represents a procedural activity that can be enacted by a CISO to manage the impact of human behaviour when users interact with organisational security controls. Note a Behavioural Control is similar to the workflow failure risk management approaches discussed in Part 1 of this thesis (Chapter **??**). Each Behavioural Control has a Control Type which indicates the associated risk management approach, such that a Behavioural Control `managesRiskOf` a specific Threat. Approaches to manage the risks imposed by human behaviour can be 'avoidance', 'transfer', 'reduction', or 'acceptance' used appropriately to promote acceptable user working practices when interacting with security. For example, one approach to reduce the risk of forgotten passwords on workflow success rate whilst maintaining the password policy would be to employ more IT help-desk staff.

6.4 CISO Consultations

CISOs use their knowledge and expertise to compose and modify information security policies, which then informs the information security stance of their organisation. We examine the process of information security policy-making and modification through semi-structured discussions with information security managers, through a mixture of surveys and phone interviews. For this we consulted two CISOs; (CISO_1) has a wealth of experience as a former CISO of a large multi-national financial organisation, and (CISO_2) is a CISO at a leading UK University with previous experience at UK regional councils. These discussions help us to understand and identify the requirements of policy makers when using information security knowledge. The consultations also build a picture of how policy makers across the community choose to interact with external bodies and other knowledge holders, identifying barriers to the sharing of knowledge and with this the potential uptake of collaborative ontology development tools. The results of our CISO consultations are organised and presented here based on the responses from the questions in Appendix C.1.

6.4.1 Policy Review Timing

Within the organisations of both CISO_1 and CISO_2 information security policies are reviewed at regular intervals, typically annually. CISO_1 points out that although this is normally the case, reviews may be forced by the emergence of a new threat or with the introduction of a new security technology. With this we may assume that CISOs require access to relevant material for guiding policy modifications to address new threats and security technologies.

6.4.2 Policy Review Resource Gathering

When reviewing and creating policies, CISO_1 refers to the following sources for guidance:

- Payment Card Industry (PCI) [143]
- ISO27001/27002 security standards [21, 22]
- Information Security Forum (ISF) [86]
- International Information Integrity Institute (I-4) [85]

CISO_2 also examines a range of sources for guidance:

- Information Technology Infrastructure Library (ITIL) [8]
- ISO27001/27002 security standards [21, 22]
- The Law Society [169]
- TechRebublic [166]

Although both CISOs work in different sectors, they both refer to the ISO27K family of standards during their work. Having to refer to a number of independent sources implies the need for this information to be cross-referenced in an appropriate manner. Within CISO_1's organisation information gathered from different sources is entered into a software-based policy management tool, which provides facilities to create information security policy database(s). The tool also allows policy content to be separated according to applicable roles (e.g. system development, human resources, auditors, etc). The tool's databases are by no means complete as they provide only information about standards. There remains a lack of coverage of some other kinds of information, such as industry best practices and information for managing human factors within IT security management.

For CISO_2 the majority of gathered information is recorded within the organisation 'as-is'. Some financial and business information is stored as an information base within an Enterprise Resource Planning (ERP) system [138]. Less than 3% of all the gathered information is formalised and managed within a standardised knowledge repository. This contradicts the need to cross-reference various sources of information. Where information is stored, it is managed within individual policy categories. This together with CISO_1's responses suggest that there are various ways to formalise the arrangement of knowledge content into policies.

6.4.3 Policy Creation and Modification

The policy databases in CISO_1's organisation are used to create, modify and publish information security policies internally upon the organisation's intranet. During policy creation or modification, the IS027001 standard acts as an underlying structure [21]. Internal policies are then created, or modified according to this structure, and developed through methods including benchmarking, forums and use of informal networks. Internal policies (e.g. software configuration settings, machine build settings) are then quite distinct from external information security standards (e.g. ISO27001).

For CISO_2, an internal information security policy is created, or modified by examining the organisation's trading environment (e.g. business, legal and common practices). CISO_2 uses these practices to facilitate the formation of action plans or strategies to achieve particular IT security-related goals. Policies are defined, or modified to help realise these strategies and are put into practice upon approval by senior management. When addressing an information security concern with the forming, or modifying of a policy, the order of strategy, policy and practice is always followed, suggesting that policies must be seen to be achieving a goal for the organisation.

6.4.4 Policy Reviews

When reviewing information security policy within CISO_1's organisation, various parties must be consulted both internally and externally (e.g. human resources, legal counsel, etc). In a regulated industry, regulators must be consulted as they may have their own published guidelines (e.g. Financial Conduct Authority [65]). Also, the views of both internal and external auditors must be sought to assist in aligning policy initiatives with industry standards. Once the policy has been decided upon, CISO_1 will talk with technologists to determine whether the proposed solution can be integrated into the organisation's applications and

systems. The proposed policy is also discussed with business people to understand how it would be received within the organisation.

$CISO_2$ also consults both internal and external parties during review of information security policies. These parties include the University Registrar, human resources, internal auditors and external legal counsel specialising in information security. Policies and any modifications are verified by peers and external advisors (e.g. legal counsel) together with additional checks by Jisc certificated legal services [91]. These responses imply a need to communicate knowledge meaningfully to peers and other disciplines.

6.4.5 Policy Justification

For both $CISO_1$ and $CISO_2$ it is the case that before a policy or policy modifications are enacted they must be justified to, and supported by, senior management. In $CISO_1$'s case policies and policy modifications are discussed in terms of risk and what the effects could be to the organisation without such policies and policy modifications being in place. For $CISO_2$ debate around policies and policy modifications is framed in terms of impact and reach including legal requirements, financial considerations, personal data protection and intellectual policy. A further consideration noted by $CISO_2$ (but which is likely applicable for $CISO_1$) is the commercial and reputation risks to the institute without such policies or policy modifications being in place. This implies a need to be able to objectively compare the advantages and disadvantages of particular approaches to information security. An ontology would serve to formalise knowledge of security solutions and expose their comparable qualities in terms of impact.

$CISO_1$ noted that situations can arise where objective evidence relating to the impact of potential policy modifications is lacking (e.g. effects on employee productivity). Here $CISO_1$ may rely upon his judgment and expertise to convey the reasons why a particular change in security procedure needs to take place. However $CISO_1$ notes that there would ideally be evidence at hand to support such an argument, implying a need to identify the objective evidence underpinning expert opinion within security policy management.

6.4.6 Policy Evaluation

Within $CISO_1$'s activities, information security policies and policy modifications are typically evaluated for correctness and effectiveness using metrics, but this is not always possible. When assessing technological solutions it is possible to obtain and analyse output from those systems (e.g. the number of e-mails rejected by anti-spam software). However

where the behaviour of users is involved it is difficult to formulate and measure meaningful metrics. An alternative approach might be to obtain agreements from department managers declaring that security measures will be enacted. These agreements are then reviewed at least annually alongside computer-based user training and forward-looking agreements, e.g. *"I will comply with these measures for the next 12 months"*. Such self-assessments then transfer responsibility for security to individuals.

CISO_2 notes yet another approach to evaluating security modifications, where the security policy evaluation process entails physically observing individuals using human-facing security controls and identifying how those controls are dealt with or how they affect working practices. Within CISO_2's organisation, business processes and workflows are also reviewed to see if current security controls are appropriate or whether they can be modified to effect improvement in success rate. That there are various options for managing the assessment of information security policies within organisations, and that their place within business processes and workflows must also be accounted for suggests a need to consider the merits of different methods for analysing security policy modifications, and equally how appropriate a particular method might be for assessing a range of policy modifications.

6.4.7 Sharing Policy Content

CISO_1 states that formal and informal groups of CISOs regularly meet to discuss security issues and share expertise. With this approach, there is potential for organisations to converge on similar solutions, even to a low level (e.g. password composition rules, password reset intervals, etc.). Sharing of information security management knowledge then occurs amongst known and trusted parties. In the academic sector, institutions tend to reach consensus on approaches to information security through regular interaction and sharing of expertise. CISO_2 also corresponds regularly with other non-academic organisations (e.g. local government). Both CISO_1 and CISO_2 would be hesitant to share policy content that exposes the security stance of their organisation. This suggests that if information security practitioners across different organisations were to share knowledge, there would need to a consideration of how to hide the identity of contributors or otherwise maintain knowledge at an abstract level so as not to betray its source (although policies may tend to be framed at a high, operational level as a matter of course).

CISO_1 would consider using policy impact knowledge from new and untrusted sources once consensus is established amongst peers. Consensus may potentially be reached through successive edits until the content is agreed (i.e. no-one feels it necessary to edit any further).

For CISO_2, anonymously supplied information would be considered once it was proven elsewhere and recommended from a trusted source. If such proof was unavailable the anonymous content would still be considered if it could be tested in an environment of limited impact and in such a way that it could do no harm to the organisation. This suggests a need for details on how supplied knowledge can be enacted and tested.

6.4.8 Core Findings

From the discussions with CISOs a number of similarities are apparent, from which assumptions may be drawn regarding the management of knowledge contributing to the impact of information security policy modifications:

- Information security policies are reviewed at regular intervals or when new security threats or technologies emerge, wherein security modifications are informed by guidance material gathered from a variety of disparate sources.

- Organisations across different sectors may form information security policies and modifications to them, according to the same guidelines (e.g. ISO27001/2). This implies that many organisations have similar information security requirements which could benefit from collaboration.

- Policy modification impact consensus is reached from correspondence and discussion amongst peers alongside the examination of guidelines and regulatory mandates.

- All material that informs policy modifications (be it guidelines, regulatory mandates, technical documentation, etc.) is recorded in-house and arranged within distinct, specialised policies, although this is not necessarily conducted in a centralised manner.

- Policies are reviewed in consultation with many internal and external parties, notably experts in legal, human resource and technical issues. These parties cannot be assumed to be proficient in the use of ontologies or information security.

- There is a need to compare different approaches to analysing information security policy impact and the evaluation of policy modifications, and to be able to provide justification for such modifications.

6.5 Summary

This chapter has described the foundations for tools specifically for Chief Information Security Officers (CISOs) and human factors researchers to create a knowledge base by aligning knowledge of information security with common human behavioural responses in an ontology. An ontology is a powerful concept as its content can be analysed from a number of different perspectives. The constructed knowledge base can be used by CISOs to predict how changing a security policy will impact human behaviour which itself impacts the success rate of a workflow. Furthermore, behavioural controls can be extracted to manage the impact of this behaviour. For instance, a complex password policy may cause delay due to a higher rate of user forgetfulness and managed by employing more IT help-desk staff to quicken the password reset process. We have described the main structural components of an ontology and how it can be used to manage organisational knowledge. An overview of current ontology development has been given which has highlighted current tools are generic in nature, relatively complex to use and require a substantial amount of initial training and configuration before knowledge capture can begin. Furthermore employing an ontology expert in order to use current tools is error prone as they are unlikely to understand the nuances of the ontology content and its alignment. We suggest dedicated ontology development tools are required to allow security domain experts to collaborate and create a knowledge base whilst hiding the details of ontology construction. The foundation ontology aligning security and human behavioural aspects is described which will form the underlying knowledge base structure for the development tools. Discussions with two CISOs are also presented which have highlighted the need for suitable tools to align security related knowledge in order to analyse the potential impact of modifying a security policy.

Chapter 7 describes the requirements and implementation of two ontology development tools. The first is graphical in nature by providing a palette of shapes and connectors that can be used by the user to construct ontology diagrams connecting and aligning knowledge fragments within the structure on the foundation ontology. Once constructed the diagram is automatically translated into the ontology language OWL which can be imported into an appropriate ontology analysis tool. The second development tool is Web-oriented allowing knowledge holders to collaborate and reach consensus over ontology content by providing a number of collaborative features. Knowledge entry is text based, aligned within the same foundation ontology structure as the graphical tool. The Web-oriented tool also provides a platform for analysing ontologies created in both tools in order to assess the impact of security policy modifications and derive suitable behavioural controls to mange that impact.

Both tools come with the foundation ontology pre-configured and provide user guidance to simplify the knowledge capture process.

Chapter 7

Ontology Development Tools

Interrelated information security and human behavioural knowledge held by security domain experts can be recorded and incorporated within the formalised structure of a dedicated security ontology to form a security knowledge base. Chief Information Security Officers (CISOs) can use a knowledge base of security related information for qualitative impact analysis of modifications to a security policy. More precisely, the recorded knowledge and its interrelations provide a CISO with a way to predict how people are likely to behave in response to potential security modifications and how that behaviour could impact the success rate of workflow. The knowledge base can also be used by a CISO to extract suitable behavioural controls to manage the impact of user behaviour when interacting with security controls. Currently ontology development is undertaken using generic ontology development tools which accommodate ontology experts, and not those individuals whose knowledge requires capture. This process is therefore time consuming and error prone, and requires appropriate technical skills. Furthermore, gaining the assistance of an ontology expert to develop the ontology on behalf of the knowledge holders is also error prone as they are unlikely to understand the nuances of the ontology content and its alignment.

This chapter presents details of two dedicated ontology development tools for CISOs and human factors researchers to collaborate and create a knowledge base whilst hiding the details of ontology construction. The requirements of both tools have been extracted from consultations with two CISOs presented in Chapter 6. The first tool is graphical in nature allowing a diagrammatical representation of an ontology to be constructed, the second is Web-oriented facilitating distributed knowledge entry and ontology construction. Both tools simplify the current knowledge capture process by removing the need for expertise in ontology construction and technologies. Furthermore they are intuitive, require no ontology configuration due to pre-configuration, and provide mechanisms to guide users, thereby

reducing the potential for errors. The tools also allow domain experts to develop and extend the ontology, and enterprises to tailor the ontology to their own requirements. Evaluation of the Web-oriented tool has been provided by the same two CISOs consulted previously.

7.1 Graphical Ontology Development Tool

A graphical ontology development tool has been implemented providing a palette of shapes and connectors for the user to construct ontology diagrams according to the structure of the underlying security ontology described in Section 6.3.2. Ontology diagrams once constructed can be automatically translated to the Web Ontology Language (OWL) [170], for further analysis in a suitable tool. The tool has been implemented as a downloadable standalone application imagining a scenario with group-based, face-to-face collaboration. The tool's design removes the need to understand ontology construction techniques allowing collaborating information security domain experts to construct and modify, share, and analyse an ontology at an abstract level. The work presented in this section has been published in [118]. First we outline the requirements for a graphical ontology development tool before giving implementation details in Section 7.1.2.

7.1.1 Tool Requirements

The main requirements of a graphical ontology development tool have been extracted from consultations with two CISOs (Section 6.4), and identified as follows:

- A simple, intuitive graphical tool to create and/or modify a security ontology in diagrammatical form. The tool will combine the unstructured knowledge of domain experts with formalised ontology structure while abstracting away details of ontology content creation and maintenance.
- The ontology editor tool must allow capture and organisation of formalised knowledge relating to familiar information security concepts (e.g. Assets, Vulnerabilities, Threats, Procedural Controls etc). Disparate knowledge fragments may also be interrelated, and users should be able to record these relationships.
- The graphical user interface must not obscure the knowledge that is represented. This may require the tool to include only the bare minimum of ontology editing controls, and to present the ontology to the user in a diagrammatical form. An approach such as providing 'drag and drop' functionality using a pre-defined set of shapes to

represent information and construct ontology diagrams may prove useful in this case. A diagrammatic, formalised representation of an ontology could then be translated to an ontology file automatically and without the need for user participation.

- The 'hidden' ontology structure should be pre-defined, meaning that no initial configuration is required before knowledge capture can begin. A domain expert would then only need to concern themselves with adding new information and connecting it to other concepts.

- It may be necessary to present ontology content to other stakeholders (such as senior management) whenever policy-related knowledge is used to identify necessary changes to a security policy, the behavioural impact of those changes, and behavioural controls to manage that impact.

- Ideally an interactive help system will be in place to aid the user through all aspects of ontology development. Other forms of assistance, such as 'tool tips', could be provided to explain features of the tool as they are being actively used. Dialog boxes and the like may also be used to restrict what the user can do, and thereby minimise the potential for errors.

- There must be mechanisms to minimise errors occurring in the knowledge capture process. This may include restrictions on data properties, and active error notifications during knowledge entry.

7.1.2 Tool Implementation

Figure 7.1 illustrates the graphical ontology development tool's main components. These are as follows:

- **Ontology Editor**: the user's interface to the ontology development tool through which ontology content is entered. A graphical representation of an ontology may be created in the editor's canvas and the corresponding machine readable ontology file created. Diagrams are built up from a set of predefined shapes supplied within the editor interface. Ontology diagrams may be saved and existing diagrams loaded into the editor for analysis and/or modification.

- **Ontology Diagram**: a graphical representation of an ontology constructed in the editor from a set of pre-defined shapes. Ontology diagrams may be saved from the ontology

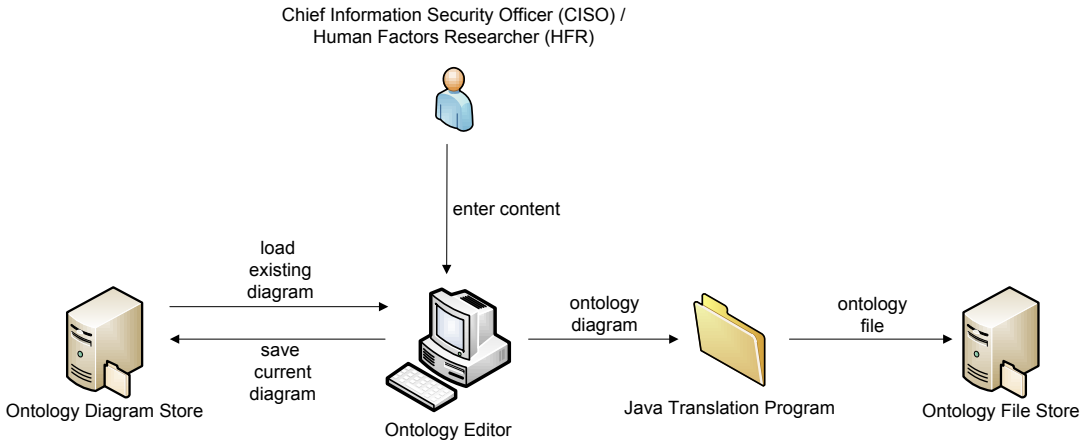


Fig. 7.1 Overview of graphical ontology development tool components which incorporates the ontology editor and two file stores, one for Visio ontology diagrams, the other for OWL ontology files.

editor to the Ontology File Store for future use or ontology diagrams may be loaded from the Ontology File Store into the editor for analysis and/or modification. The Ontology File Store may be located on the user's own machine or held as a centralised organisation database.

- **Ontology File**: a textual representation of an ontology written in OWL. An ontology file may be created from within the editor once an appropriate ontology diagram has been constructed. The creation of the ontology file takes place via the Java Translation Program. Ontology files are saved to the Ontology File Store once created and may be retrieved from the Ontology File Store for analysis using an appropriate application. The Ontology File Store may be located on the user's machine or held as a centralised organisation database.

- **Java Translation Program**: a program which parses an ontology diagram and obtains data on the shapes and their connections contained in that diagram. This data is processed resulting in the creation of a corresponding ontology file. The translation program is activated from the ontology editor interface once an appropriate ontology diagram has been constructed.

Ontology Editor

The ontology editor offers the user a simple graphical interface where they can enter and capture their knowledge in a graphical form. This is in effect a graphical representation of

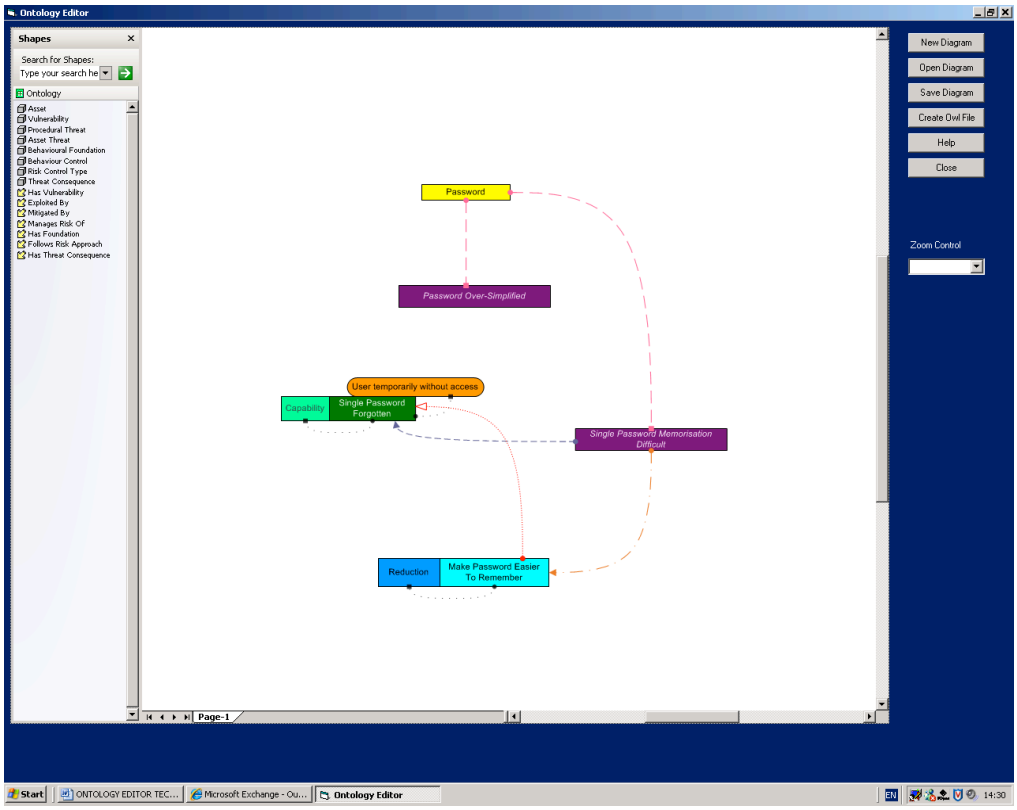


Fig. 7.2 Graphical development tool's interface which allows users to drag and drop populated ontology components and connect them before automatically encoding the ontology in the Web Ontology Language OWL.

the ontology (an ontology diagram). All aspects of the security ontology are pre-defined, and with the integrated help system, diagram construction is simplified and intuitive. Figure 7.2 shows the user interface. To the right of the interface are the available controls (new/open/save diagram, create OWL file, etc.). To translate the current diagram in the editor to the corresponding ontology (OWL) file, the user is only required to input a location to which to save the resulting file, thereby abstracting away any requirement to know about ontology construction.

The main window of the editor interface is where the diagram construction takes place, utilising the Microsoft Visio 2007's drawing control [128]. Microsoft Visio 2007 [130] is a graphical tool for creating a wide variety of diagrams to visualise, explore and communicate complex information. The Visio drawing control allows the full functionality of the Microsoft Visio 2007 drawing surface to be embedded seamlessly into any standalone application written in the Visual Basic 6.0 programming language [129]. The drawing control itself holds a template which consists of all shape data (stencil) and drawing page settings. On

start-up or beginning a new diagram, a master template is loaded into the control offering a blank page. Subsequently, when a diagram is saved the resulting file includes all data from the master template which becomes available when the diagram is reloaded.

On the left of the drawing control is a list of pre-defined shapes available for constructing diagrams. This collection of master shapes is a 'stencil' which is stored as part of the master template. Each 'master' shape in the stencil contains individual data (name, colour, size, etc) and can be re-used by dragging and dropping onto the drawing page. The stencil shapes are pre-defined for the user and consist of boxes (concepts) and arrows (relationships) to coincide with the structure of the underlying foundation ontology described in Section 6.3.2. Other pre-defined elements include the drawing page settings which are stored as part of the master template. These settings dictate not only the page size and orientation but how the shapes behave on the drawing page with regard to lining up and/or connecting to each other (snap and glue). Connecting two boxes with an arrow is straightforward through the use of the Visio auto connect feature which allows a box to be dragged around in the drawing page while still connected to any adjoining boxes. Use of this functionality means that the user does not need to manage how diagrammatic content is arranged within the editor, and is free to concentrate on the development of the content itself.

A number of mechanisms are used to restrict the potential for errors in the ontology population process. When a new box (concept) is added the user is forced to enter that concept's content before they can proceed. Also, when joining boxes, range and domain restrictions are in place on the connecting arrows so only certain boxes can be associated with each other with certain arrows. This removes the possibility of incorrect connections being made. One further error handling feature of the editor is the detection of any unconnected shapes (isolated boxes, unconnected arrows, etc.), which if found to be the case halts creation of the ontology file until the user has resolved any such errors. An integrated help system is in place to aid the user in diagram construction. When a new box (concept) is added to the diagram a dialog box opens explaining the box's type, how it is used and how it may be connected to other boxes in the diagram. An example dialog box is shown in Figure 7.3. Dialog boxes also aid the user through the creation of an ontology file, explaining the input required, e.g. the location to which to save a file. Tool tips are in place for all shapes added into a diagram, the master shapes contained in the stencil and for the user controls. These provide active assistance in explaining the function of the various elements of the editor whenever the user floats the mouse cursor over them. A full help page explaining the editor and its use is also available through the user controls.

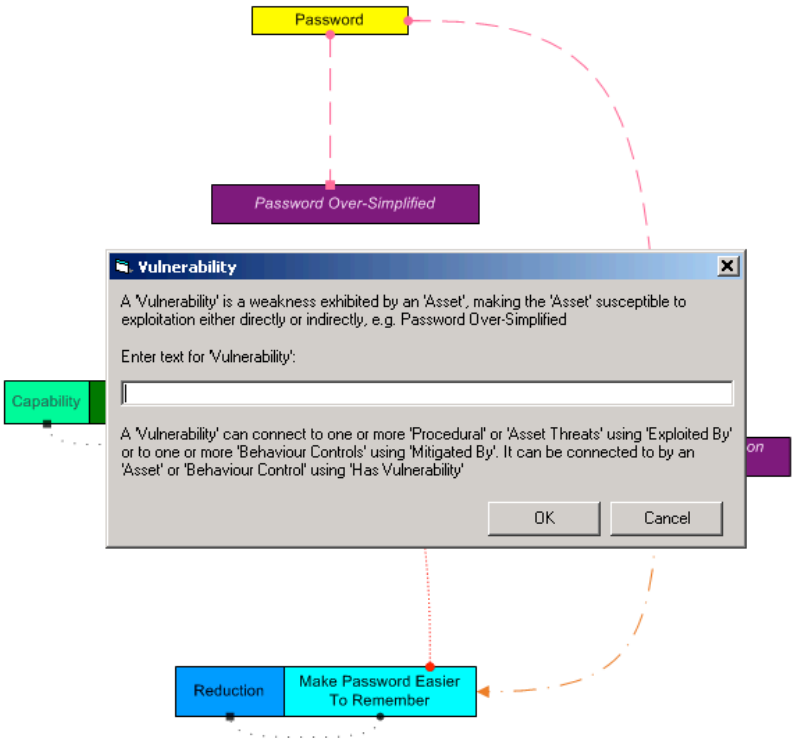


Fig. 7.3 Graphical development tool dialog box for adding properties to a new security ontology 'vulnerability' component which can then be incorporated with pre-existing ontology content.

Ontology Diagram

An Ontology Diagram is a graphical representation of an ontology constructed in the editor from a set of pre-defined shapes. Ontology classes are represented as boxes, each class having its own assigned colour (so as to help the user distinguish between different classes). Each box represents individual ontology concepts and records the content for that concept in textual form. Ontology properties are represented as arrows, each property having its own assigned colour (again to help differentiate between ontology constructs). Each arrow represents a relationship between individual elements from different concept families.

When beginning a new diagram, the generic ontology template is loaded into the editor's drawing control window. The template can be thought of as a blank page which contains the ontology stencil and page settings. Once a diagram is saved as a Visio XML drawing (.vdx), template data is also saved within the resulting diagram file, allowing that file to be re-opened at a later date. Both drawing and template data is used to construct the corresponding ontology file. When a user wishes to create an ontology file from the diagram currently

loaded in the editor, that diagram is saved automatically into a temporary folder ready for further processing. Any files held in the temporary folder are automatically removed when the editor is closed down.

Ontology File

An Ontology File is a textual representation of an Ontology Diagram and is obtained by translating the content of Ontology Diagrams into OWL. The content of the resultant OWL files has the potential to be processed automatically by software programs, thereby providing scope for expert knowledge to be used in various ways. Once created, all ontology files are stored in a user designated Ontology File Store. An example information security ontology file created in the graphical ontology development tool and translated to OWL is presented in Appendix D.

Java Translation Program

The Java Translation Program's purpose is to process a given Ontology Diagram and produce the corresponding Ontology File, in effect translating the representation of the ontology content from a graphical form to a machine-readable OWL format. As this translation is done automatically it removes the need for the user to have any knowledge of ontology construction and the syntax and semantics that is involved in that construction. The translation program also removes the need for a user to manually enter ontology structural information, ontology content and the relationships between that content into a standard ontology editor, e.g. Protégé [133], which can be a lengthy, complicated and error-prone process. The Java Translator Program is written in the SE 7 edition of the Java programming language [93] and is deployed as an executable Java archive on a user's machine. The Java Translator Program requires the following input parameters for operation:

- The filename and location of the Ontology Diagram to be processed.
- The user must specify a URI (Uniform Resource Indicator) which identifies the ontology. This is normally in the form of a web address and can be thought of as the ontology's name.

An overview of the translation process is provided in Figure 7.4. This process is twofold. First an Ontology Diagram file must be processed and the relevant data obtained. Secondly, that data must be transformed into the OWL format before being compiled into an Ontology File. Having been saved in XML format the Ontology Diagram file is passed to the Java

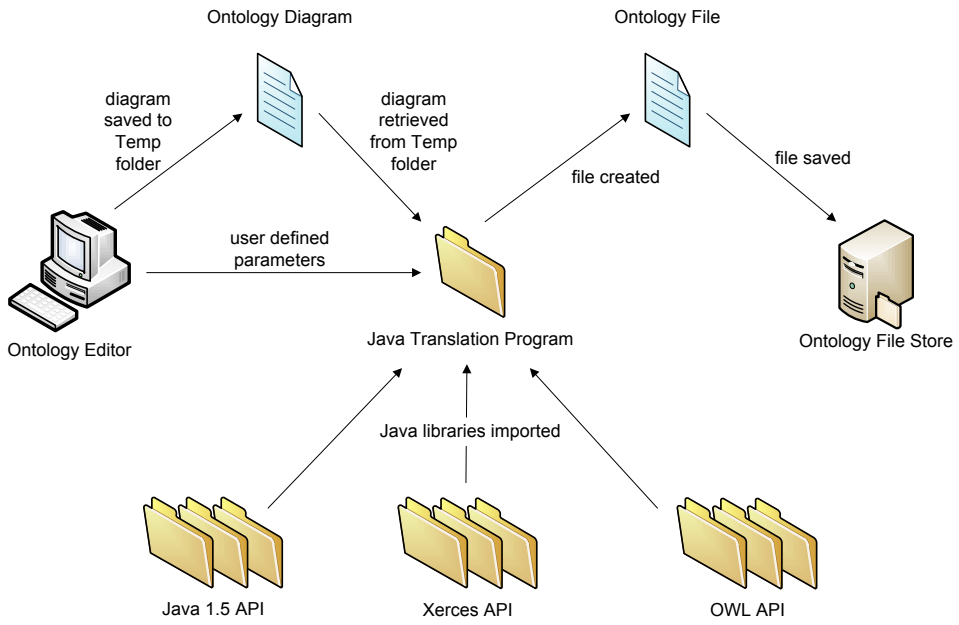


Fig. 7.4 Overview of graphical development tool Java Translation Program's components which imports Java, Xerces, and OWL APIs to automatically encode Ontology Diagrams into an ontology.

Translator Program where it is parsed for validation and the creation of an internal representation for use in subsequent processing. The creation of a Java parser is performed using the Java libraries from the Xerces API [168]. Each shape in the Ontology Diagram, both boxes and arrows, are given a unique name which allows that shape to be uniquely identified during the translation process. For each shape in the Ontology Diagram the following data is retrieved:

- The shape's type: each shape is an instance of a master shape held in the ontology diagram stencil. The shape's type is equivalent to the master shape's name.
- The shape's text: if the shape is a box, the text entered by the user inside that box.
- The shape's connections: if the shape is an arrow, the two boxes it connects.

Once this data has been retrieved and stored in memory the second phase of the translation process commences, namely the creation of the Ontology File. To create the file, libraries from the OWL Java API [81] are used for parsing and writing of OWL ontology files within a Java program. The following aspects of the Ontology File structure are predefined in the Java Translation Program and written directly to the Ontology File:

- Ontology classes: these mirror the boxes in the ontology diagram stencil.
- Ontology data/object properties: these mirror the arrows in the ontology diagram stencil.

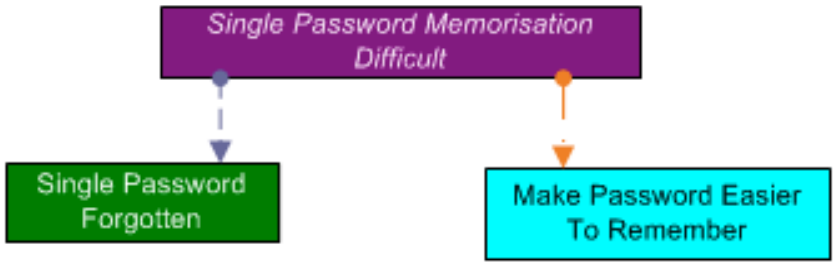
Axioms are added to ontology classes to arrange them into a hierarchal structure within the ontology, and to properties to specify range and domain restrictions. By predefining selected aspects of the ontology file structure the user is not required to understand and enter this information. The data obtained by the Java Translation Program from the Ontology Diagram is used to create instances of the Ontology File classes. An example of the OWL code generated to represent an instance, or individual can be seen in Figure 7.5. The data is translated as follows and written to the Ontology File:

- The shapes type (master shape's name) becomes the individual's type, or owning class. This can be seen in Figure 7.5 as the parent element's name, e.g. `Vulnerability`.
- The shape's text is set as the value of the parent element's `rdf:about` tag, e.g. `SinglePasswordMemorisationDifficult`.

An individual's properties are determined by finding any arrows from the ontology diagram data that have that individual as its starting point. A child element is added to the individual for each property, e.g. `exploitedBy`. The individual at the finishing point of the arrow becomes the value of the child element's `rdf:resource` tag, e.g. `SinglePasswordForgotten`.

7.2 Web-Oriented Ontology Development Tool

A Web-oriented ontology development tool has been implemented providing text based knowledge entry according to the structure of the underlying security ontology described in Section 6.3.2. The Web is a natural platform for collaboration and knowledge-sharing, by distributing the development process and disseminating the resulting knowledge across the information security community. The tool has therefore been implemented by tailoring the open source Web ontology tool Web-Protégé [172]. Using the collaborative tool, CISOs and human factors researchers can potentially submit, comment on, and peer-review submitted knowledge, with the ultimate aim of reaching consensus on a robust body of information security knowledge useful for CISOs to analyse the potential impact of security policy changes. The tool's design has intentionally removed the need to understand ontology




```
<Vulnerability
  rdf:about="#SinglePasswordMemorisationDifficult">
    <mitigatedBy rdf:resource="#MakePasswordEasierToRemember"/>
    <exploitedBy rdf:resource="#SinglePasswordForgotten"/>
</Vulnerability>
```

Fig. 7.5 Section of example Ontology Diagram and corresponding OWL code representing a password vulnerabiltity which is exploited by a threat of being forgotten, and mitigated by a behavioural control making a password easier to remember.

construction from the standard Web-Protégé tool, adapted its collaborative features towards security domain experts, and added user guidance for intuitive knowledge entry. The work presented in this section has been published in [117]. First we outline the requirements for a Web-oriented ontology development tool before giving implementation details in Section 7.1.2.

7.2.1 Tool Requirements

The main requirements of a Web-oriented ontology development tool have been extracted from consultations with two CISOs (Section 6.4). As the tool is Web-based and collaborative in nature it comes with both the requirements described in Section 7.1.1, and the additional requirements listed below:

- The interface should allow collaborative capture of distributed knowledge between disparate parties. There should also be features to allow members of the user community to reach consensus (e.g. by discussing content).

- Users must be able to preserve an appropriate level of anonymity. Users should not be expected to divulge specific organisation security practices.

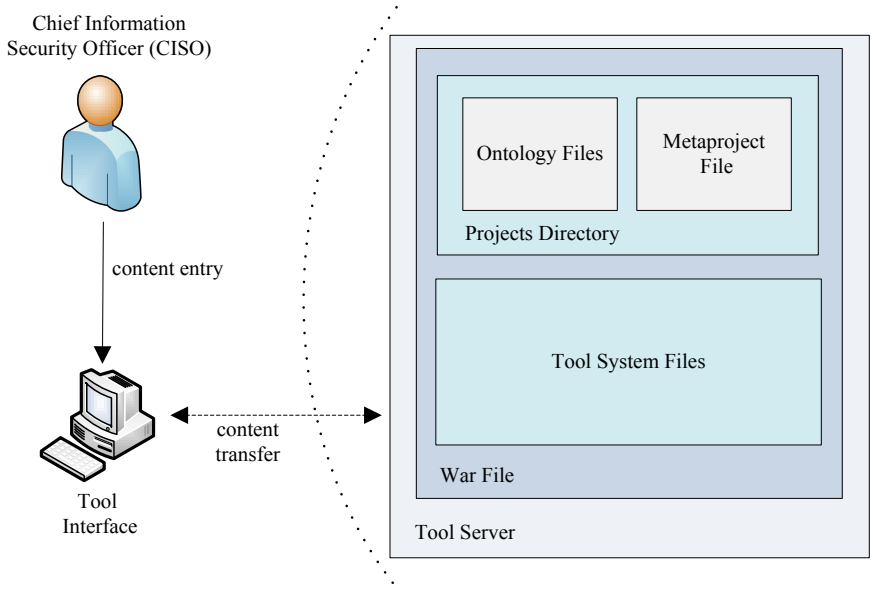


Fig. 7.6 Overview of Web-oriented ontology development tool components which incorporates a Tool Server which houses the tool and allows authorised users to enter content and collaborate remotely.

7.2.2 Tool Implementation

The open-source ontology editor Web-Protégé [172] forms the foundation of the tool. Web-Protégé is a Web-accessible collaborative ontology editor built with a number of collaborative features that can be tailored to meet our requirements. An overview of the components of the collaborative information security ontology editing tool is shown in Figure 7.6. The tool is composed of the client-side Tool Interface and the Tool Server.

- **Tool Interface**: a browser-accessible Web application through which ontology content is viewed and manipulated.
- **Tool Server**: the tool is stored as a Web application archive file (War file) on a centralised Web application server. The server provides remote access to the latest version of the tool.

A War file contains both the tool system files and the ontology files. Having all the necessary files in a single archive allows for simple server deployment. Tool system files within the War file hold the tool's compiled source code and supporting images and html pages. All of the tool's current ontology files are stored in the Projects directory. The Projects directory also contains the tool's Metaproject ontology file, which describes access conditions for the tool's ontologies.

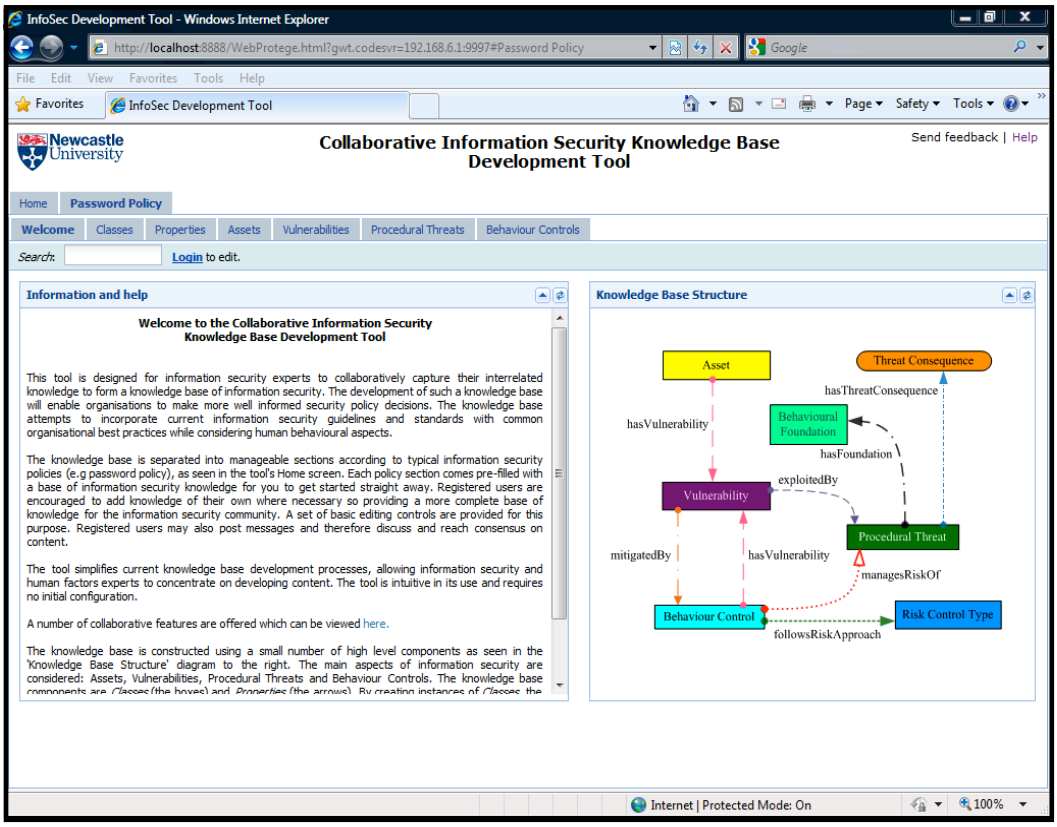


Fig. 7.7 Web-oriented ontology development tool's Welcome page which gives an introduction to the tool, an overview of the underlying security ontology structure, and tabbed pages to access and modify ontology content.

Tool Interface

The Tool Interface allows CISOs and human factors researchers to access ontology content in a manner that abstracts away details of ontology construction. These users are then free to view, add, modify or relate fragments of information security and human factors knowledge to help them:

- View information security policy and human behavioural knowledge and the interdependencies between knowledge fragments.
- Record (and share) knowledge of information security policy, their behavioural impact, and behavioural controls (through use of editing controls).
- Collaboratively refine the knowledge stored within the underlying ontology, using the tool's collaboration features.

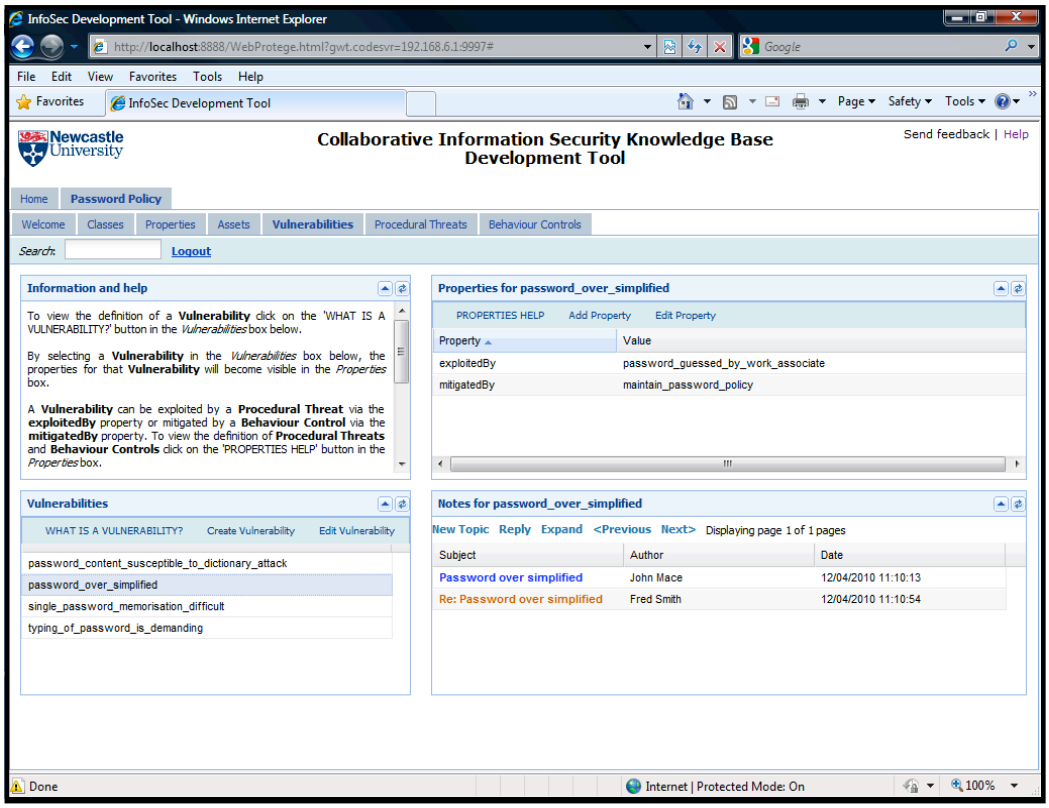


Fig. 7.8 Web-oriented ontology development tool Content page which shows 'vulnerability' compenents incorporated into the security ontology, along with their interrelationships with other content, a help section, and notes posted by users relating to ontology content.

Users are initially presented with a screen offering a list of available ontologies. For demonstration purposes distinct ontologies are provided so as to mirror an organisation's policies (e.g. USB stick policy, password policy, etc.). These ontologies employ the structure described in Section 6.3.2, and can potentially be pre-populated with knowledge content that can be viewed and/or extended with user-supplied knowledge. The main user interface consists of a number of tabbed pages providing users with ontology content, ontology editing controls, and features to advise users in how to interact with the tool. Only authorised users can access editing controls and make changes to the ontology, by providing a registered user ID and password. A 'Welcome' page (Figure 7.7) provides an overview of the tool, its functionality, and the base ontology structure, so as to familiarise new users and act as a reference for returning users. The 'Class' and 'Property' tab pages provide the user with descriptions of the ontology's structural components and how they interact with each other.

The 'Content' pages (Figure 7.8) allow a user to view and edit ontology content. Any modifications made to an ontology are immediately visible to all those users currently access-

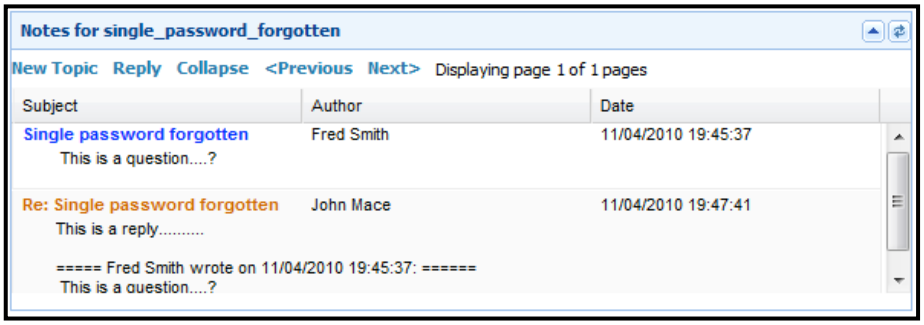


Fig. 7.9 Web-oriented ontology development tool portal which allows users to post notes relating to ontology content.

ing the tool, thereby making it useful in situations that require newly-available knowledge (e.g. when a new threat or technology emerges). Four separate Content pages cover the main classes of the information security ontology structure; Assets, Vulnerabilities, Threats and Behaviour Controls. This allows a user to access ontology content from a range of perspectives rather than be restricted to a particular starting point. Each of the Content pages contains a portlet listing the individual fragments of knowledge associated with the particular class. Selecting one of these individuals will display the properties associated with it (e.g. the Vulnerabilities of a selected Asset). A combination of editing controls and dialog boxes allow a user to add a new class individual or change the text of an existing entry. Warning mechanisms limit the potential for user error and ontology inconsistency (e.g. to prevent addition of an individual with the same text as an existing entry). Users can also connect fragments of knowledge according to the relationships defined in the ontology structure. The tool restricts the user's choice of property connections accordingly by way of restrictive controls, for example a user may only connect an individual Vulnerability to a Threat or Behaviour Control via the `exploitedBy` and `mitigatedBy` properties respectively. Each Content page has an information portal providing an overview of that page's content and relevant instructions on how ontology content may be extended and/or edited.

A 'Notes' portal (Figure 7.9) enables users to annotate, discuss and reach consensus on ontology content by posting messages, akin to a bulletin-board system. Posted messages are linked with specific content and are visible to all users when that content is selected. Authorised users may post new messages or reply to pre-existing messages via an e-mail style dialog box. The tool has the capacity to guide users through aspects of ontology development and exploration, using a variety of help features.

7.2.3 CISO Tool Evaluation

An evaluation has been carried out of the functionality and usability of the Web-oriented ontology development tool by presenting it to the two CISOs consulted previously (see Section 6.4). During evaluation the tool was demonstrated according to a structured demonstration plan, and the chance offered to participants to practice using the tool first-hand. This was immediately followed by a series of structured questions that serve to relate each participant's impression of the tool to the tool's requirements outlined in Section 7.2.1. The structure of the tool evaluation sessions is outlined in Appendix C.2.

As well as evaluating the requirements that were defined for the tool, the evaluation sessions provided the opportunity to centre discussion of collaboration and knowledge-sharing around a tangible tool, eliciting comments about how the tool may be practically applied (along with any issues that have the potential to limit application of this, or a similar, tool). Each complete session was voice-recorded to allow us to capture feedback from the participants throughout the evaluation process.

Knowledge Capture

Overall, the tool was considered as being approachable and easy to navigate, with a clear ontology structure, thereby making progress towards accommodating the needs of target users. CISO_2 stated that *"regular IT security folk will be able to dig into [the tool] and use it"*. The tool's interface was not thought to be confusing but instead very clean, with a *"nice flow round the screen"* (CISO_1). CISO_2 regarded the ontology structure, covering Assets, Vulnerabilities, Procedural Threats and Behaviour Controls, seemed *"very obvious"*.

The notion of separating ontology content, thus making the ontology more manageable, was described by CISO_2 as a very complex area, but that *"splitting the content into policies makes sense"*. However CISO_1 argued that it might be preferable to *"slice and dice the content whatever way the user chooses"*, and that individual users might potentially not recognise policy content that they would otherwise think is relevant to them if it is not appropriately labelled. For example, a security manager might regard *"identification and authentication"* as separate technologies, whereas another manager would regard the same content as *"sign-on"* policy. This problem of potentially unexpected terminology *"makes it less accessible in some ways"*.

With the capturing of content, the tool was described by CISO_2 as generic and *"open enough to do just about anything in information security"*, qualified with examples of issues within networking management, such as firewalls, denial of service attacks, and routing. The

tool can potentially capture knowledge of sophisticated information security issues which can *"become horribly complex but you could end up with areas of specialisation"*.

$CISO_2$ suggested that ontology content could be *"tuned"* according to the user's business sector, for example banking. However, each sector tends to have slight differences in information security needs, and so any content considered irrelevant to that sector would be omitted from the user interface. However, it would depend on *"who takes up the idea [of the tool] first, so making it a while before tailoring content becomes an issue"*. Alternatively, $CISO_1$ suggested that one incentive to use the tool *"would be if you could tag items to make it specific to your organisation and then ... download an extract or have a version of it which is for your own organisation, so that you can ignore stuff which isn't relevant to you"*. $CISO_2$ made a similar comment that keeps content within the tool, suggesting that when users are creating an account they should declare their business sector so that the tool may tailor content to that sector once the user is logged in.

The mechanism which detects whether content is already added was described by $CISO_2$ as good, but the recognition of similar content is not addressed. Not having this feature was considered *"a bad thing"*, as there was potential for multiple entries to describe the same concept. $CISO_1$ built upon this issue, speculating that duplicates might go beyond spelling (e.g. *"pwd"* instead of *"password"*) to similar terms that use natural alternatives (e.g. *"user ID ... logon ID ... username ... credential"*).

Currently the tool relies on users to use the collaborative features to inform others when similar content is already expressed in the ontology. $CISO_2$ stated that this notion *"would probably work"* as the *"community will police itself to a certain extent but if too much policing is needed [the community] will get tired of it"*. $CISO_1$ echoed this - *"what's the incentive ... to edit somebody else's content?"*, *"just because it's easy doesn't give me a carrot to want to do it"* - and suggested a solution of employing an administrator to *"collapse"* duplicates (which potentially defeats the purpose of the tool).

Knowledge as Evidence

In relation to the potential for ontology content to be used in supporting policy modifications and communicating potential impacts, $CISO_1$ speculated that *"you'd want to use this ... because it takes you through a structured way of doing a risk assessment ... but the only way that you'd want to do it is if you can make it specific to your organisation"*, reiterating the idea of *"tagging"* organisation-specific content. $CISO_2$ similarly *"would use [the tool] and happily add content but would need to see the outputs"* before using it within their organisation (*"Would it allow me to make a business case? ... Something that would support*

that would be really useful") before going on to speculate that if the tool was being used to assess threats etc. in a particular business sector then *"it would be good enough as it is"*. CISO_1 echoes similar thoughts: *"what makes it information is the specificity for your situation"*.

CISO_2 noted that an information security manager *"cannot simply say we need to implement this because we are insecure"*. The manager must be able to present a business case to justify modifications to information security policy and their impact to senior management from a range of possible solutions, and *"the tool allows you to do that"*. The business case will state *"we can do this, this and this to get us there and then do calculations to work out the costs"*. A scenario could be imagined where departmental managers (e.g. finance or IT helpdesk) are consulted when making security policy modifications and who use the tool's content to help work out costs: *"at the very least [the tool] helps to identify stakeholders to communicate with"*.

Collaboration and Consensus

CISO_2 considered the collaborative features very useful in discussing knowledge with peers or other stakeholders within an organisation. The ability to add notes to content was described as *"a nice feature"* that *"added richness to the tool"*. When discussing a threat a user could *"post"* a message asking *"how real is this threat?"* while the reply could be "*we've actually had a breach on this"*. The attachment of messages to specific content made *"perfect sense"*.

Regarding the collaborative features, CISO_1 said that *"it's a time-save thing"* so that users are *"not having to reinvent the wheel"* at each organisation. However CISO_1 also speculated that *"within the organisation, that time-saving doesn't mean anything"* unless it is with auditors. For CISO_2 the idea suggested of providing additional mechanisms for reaching consensus, such as a voting or rating system, could be seen as being useful. This would be akin to *"Ebay where the voting on how good a vendor and supplier are is a big deal"*. Such a mechanism promotes sensible levels of discussion while ensuring that content stays appropriate: *"You end up with the community vetting itself which is what you want to do"*. CISO_1 speculated however that users might inevitably just *"grab"* meaningful content, returning to the issue of how to *"incentivise people to comment and use"* the tool.

According to CISO_2 encouragement for users to record and share their knowledge may be brought about by stressing, *"this is your community and your tool to help you. The richer you make it the more powerful it becomes"*. A suggestion was to implement a system where users are asked to review recently added content, perhaps through an e-mail style inbox that highlights content added since their last login. It appears from the evaluation responses that

the need for users to be able to discuss content and be aided in reaching consensus has been achieved, at least in part, via the notes feature. Realistic suggestions for further mechanisms include a voting/rating system enhancement to facilitate content selection.

User Anonymity

CISO_2 noted that the user interface did not raise any privacy or security concerns as the user's organisation is not being identified during content entry. The notion of anonymity is achieved by users logging into the tool through a *username* alias. CISO_2 suggested the tool should *"stress the fact that content can be entered anonymously and content cannot be traced back to an organisation"*. CISO_1 echoed this concern: *" the only drawback would be the level of comfort ... over the ... secrecy around the specificity to my organisation ... how would I know that other people can't see that?"*. The mechanism for users to create an account, once implemented, must *"vet people and make sure not just anyone can gain access"* (CISO_2). The suggestions by both CISOs for personalising ontology content bring with them similar issues of guaranteeing that personal identifiers remain protected.

It appears from the evaluation responses that the need for users to submit information while not divulging specific organisation security practices can be achieved within the tool, although there are understandably many concerns surrounding the issue of user anonymity.

User Guidance

CISO_1 thought that the tool was helped by a *"very nice layout"* that was *"not confusing"*. According to CISO_2 the user interface was not thought to be confusing but seemed *"fairly simplistic"* suggesting an appropriate level of detail for *"non-ontology"* experts building an ontology of information security knowledge. CISO_1 suggested that users *"might ... want to follow something all the way through"* to *"produce another view"*, such that the interface changes dynamically depending on selected content (e.g., moving automatically to another screen). CISO_2 noted that *"two or three different learning styles are catered for"* by the help features, for example the *Welcome* screen provides an overview of the ontology structure in both graphical and textual form. *"Some people will read the prose while others will use the picture. For me the [diagram of] the knowledge base structure was very useful"*.

Further Comments

Some extensions to the tool's underlying ontology structure were suggested during the evaluation sessions. Such extensions could include - according to CISO_1 -, details of *"technical*

controls" and their threat management approach (e.g. *"detective"*, *"defending"*). CISO_2 suggested that policy implementation costs, methods of policy enforcement and measurements of policy success would be useful (*"Cost is a big deal. Quantifying cost, cost of implementation, cost of risk etc. [is important] but it will be hard to categorise the financial [costs]"*).

Summary

From the evaluation responses the tool largely satisfies the requirements outlined in Section 7.2.1. A summary of other key points that emerged from the evaluation sessions is as follows:

- The tool is approachable and easy to use.
- Care must be taken if dividing the underlying ontology according to distinct policies, as this may cause confusion.
- Concentrating the tool towards particular sector- or organisation-specific concerns may distort content that is essentially shared. Solutions include providing facilities to "tag" relevant content.
- Content duplication could cause problems that might only be resolved through concerted efforts by community members.
- Collaborative features can help users to reach consensus, but must provide proper incentives to maintain a collective community effort.
- Users would be able to preserve their anonymity with the tool, but would need reassurances that it does not disclose the organisation's security posture.
- The tool can support communication of policy behavioural impact and behavioural controls to stakeholders, however to fully exploit this capability ontology content would need to include or be enriched with relevant sector- or organisation-specific content.

7.3 Summary

This chapter has presented user requirements and high-level implementation details of two dedicated ontology development tools for Chief Information Security Officers (CISOs) and human factors researchers to collaborate and create a knowledge base whilst hiding the details

of ontology construction. The resulting knowledge base can be used by CISOs to analyse what impact changing a security policy may have on human behaviour and how in turn that behaviour impacts on the success rate of a workflow. Behavioural controls can also be extracted to help manage the predicted human behavioural impact. Both tools automatically deliver machine readable Web Ontology Language (OWL) ontology files based upon content recorded within the structure of an underlying information security and human factors ontology. The first tool is graphical in nature and implemented as a standalone downloadable application for use in face-to-face collaboration scenarios. Ontology development is carried out in the tool by users constructing a diagram of the ontology's concepts and relationships within a drawing canvas and translating this diagram automatically to an OWL ontology file for use in a suitable analysis tool. Diagram construction is intuitive with a simple drag and drop approach using a small set of pre-defined shapes representing high level ontology concepts and relationships. The second tool is Web-oriented for use in scenarios of distributed knowledge entry and ontology construction. Community users can therefore record and share their knowledge within the structure of a pre-defined information security ontology which has the potential to improve impact analysis of policy modifications by providing a single, shared, comprehensive reference of interrelated information security and human factors knowledge. The Web-oriented development tool's current implementation has been evaluated by two CISOs from varying backgrounds, meeting with general praise for both its appearance and functionality. Both tools simplify the development process, requires little or no instruction to use due to their interactive help system and reduce the potential for errors in ontology creation. Most importantly knowledge holders can develop ontology content without the need to know of ontology construction.

Chapter 8

Conclusion

This thesis has presented new tools and techniques to help a Chief Information Security Officer (CISO) analyse the potential impact modifying an information security policy has on the success rate of business processes expressed as workflows. Information security policies are necessary for organisations to achieve regulation compliance, establish trustworthiness as information guardians, and to gain business advantage. A CISO must manage this challenge and take responsibility as the main decision maker for setting information security policies which express rules, procedures and controls to guard against adversarial threats, and instigate secure behaviour of users processing information to complete workflow tasks. Dynamic business environments make it necessary for a CISO to periodically review and modify a security policy to keep it inline with the current security landscape. Policy modifications can affect the success rate of a workflow, that is, removing a security constraint can increase success rate but may be detrimental to security; adding a security constraint can reduce success rate or even eliminate it if users are blocked from completing tasks. Although restrictive modifications to security policies are deemed necessary, or even enforced by regulation updates, they often place new burdens on the completion of workflows, making it necessary to find an agreeable trade-off between business and security needs. Before implementing a policy modification, a CISO must gain the approval of business leaders concerned with the impact of the modification in terms of the benefit to security, and the cost to workflow completion. It is important then that the CISO can analyse the potential impact of a policy modifications on workflow completion and communicate this impact in an understandable way to business leaders.

The behaviour of human users who execute workflow tasks affects and is affected by security thus making the human factor an important consideration when analysing the true impact of policy modifications. Two specific types of security impact analysis involving

human behavioural factors have been considered. The first it quantitative in nature, centred around the notion of workflow resiliency which indicates how likely a security constrained workflow will complete under the assumption users may become unavailable at runtime. Security policy modifications can have an impact as they may prevent those users who are available from being assigned the execution of tasks in order to complete the workflow. This can either force the workflow to be terminated early of the security policy to be violated. Existing approaches to workflow resiliency offer binary solutions, in the sense that a workflow is either resilient or not when applying any policy modification. They do not indicates a degree of resiliency change and whether one policy modification has more or less impact than another. Current approaches may therefore not always be practical when analysing the potential impact of policy modifications on the success rate of workflows that do not have full resiliency, but still have some resiliency nonetheless.

The second type of analysis is qualitative in nature, facilitating analysis of how users are likely to behave in response to policy modifications and how this behaviour can affect the success rate of a workflow. A large amount of research exists on human behavioural factors in a security setting which can be incorporated with the knowledge held by CISOs within the structure of an ontology to form a security knowledge base. An ontology is a description of concepts, their properties, and their interrelationships that exist within a particular domain. A security ontology can be used by a CISO to analyse the potential impact of policy modifications on user behaviour and extract behavioural controls to manage that impact. Although ontology development tools exist they require expertise in ontology technologies and are not suited to CISOs and human factors researchers whose knowledge requires recording and interrelating to existing content, making the development process time consuming and error prone.

A lack of suitably aimed impact analysis tools and technologies for CISOs means impact analysis of security policy modifications is currently a somewhat manual and ambiguous procedure. Analysis can be overwhelming and yield unsubstantiated results, especially when workflows are complex, have a large workforce, and diverse security requirements. The tools and more formal techniques outlined in this thesis have been devised specifically for CISOs to help them analyse the potential impact of modifying security policies has on the success rate of a workflow. More precisely, these tools and techniques have been designed to efficiently compare the impact between two security policies applied to the same workflow, one before, the other after a policy modification. Analysing policy impact at design time can help avoid putting unworkable security policy into practice which may have a detrimental affect to workflow productivity.

8.1 Research Outcomes

Two open research problems were identified concerning security policy modification analysis in terms of how a modification potentially impacts the success rate of a workflow when considering human behavioural factors. This section revisits these two research problems outlined in Section 1.3 and summarises the contributions made in this thesis that build on existing work to lay the foundations for proper solutions to those problems.

8.1.1 Problem 1

CISOs needs fine grained metrics to analyse the potential impact security policy modifications have on the resiliency of a workflow.

- **Contribution 1.** We began by giving a formal definition of a security constrained workflow and considered the problem of workflow satisfiability, that is whether a plan exists which assigns the execution of all tasks to users while satisfying all security constraints. This problem is known as the workflow satisfiability problem. We then considered users may become unavailable to execute tasks at runtime and extended our workflow definition to expose this notion. We then considered the resiliency of a workflow, that is the likelihood it can be completed while still satisfying security constraints under the uncertainty of user availability. Existing work on the workflow satisfiability problem and workflow resiliency is more concerned with finding efficient algorithms to solving these problems, and tends to be binary in nature, stating whether a workflow is or is not satisfiable or resilient. In Chapter 2 we defined a novel approach to workflow satisfiability and resiliency in the form of workflow metrics which provides a quantitative measure of satisfiability and resiliency, rather than a boolean one. In Chapter 3 we showed that a plan can be found which satisfies a security constrained workflow by reducing the problem to that of solving a Markov decision process (MDP). We model the process of assigning the execution of tasks to users under security constraints as a workflow Markov decision process (MDP_W) and solve it to find the optimal policy. The optimal value function of the MDP_W returns the workflow's quantitative satisfiability, either 1 if the workflow is satisfiable, or 0 otherwise. By introducing probabilistic user availability into an MDP_W we showed how the optimal value function returns a measure of quantitative resiliency, in other words a value between 1 and 0 indicating the expected maximum probability a security constrained workflow will complete successfully. A workflow completes successfully if a feasible

complete and valid plan can be found, that is a plan that assigns the execution of all tasks to users who have a level of availability, while satisfying all security constraints. It is the measure returned by the optimal value function that expresses the impact of the current security policy on the success rate of a workflow. By changing the reward function of an MDP_W we showed different resiliency metrics can be generated; *quantitative resiliency* which indicates the likelihood of successfully completing a workflow, and *distance resiliency* which indicates how many tasks are likely to be completed before the workflow deadlocks. Distance resiliency is a useful analysis metric as it can indicate to a CISO at what step in a workflow the security policy causes a deadlock. The CISO can then effect necessary modifications to security constraints related to tasks they may be executed at that step.

- **Contribution 2.** Generating workflow metrics using an MDP_W can be a complex, time consuming and error prone process, therefore we turned to specialist automated tools. In Chapter 4 we presented a systematic approach to encode an MDP_W into the probabilistic model checker PRISM, a tool which enables the specification, construction and automatic verification of property existence in probabilistic models such as MDPs [104]. PRISM is managed and developed by active researchers at Oxford University and features in over 500 peer reviewed publications at the time of writing. Resiliency and satisfiability metrics are automatically generated by defining specific properties we wish PRISM to verify exist in one or more states of an MDP_W model. For instance, in the case of quantitative resiliency, PRISM verifies the maximum probability of a reachable state existing that represents a workflow has terminated and the execution of all tasks have be assigned to available users while satisfying the workflow's security policy. By example we applied arbitrary policy modifications to a workflow and showed how a CISO could use the metrics we generate to analyse the resiliency of a workflow, and how resiliency is impacted by calculating the resiliency before and after the policy modification and comparing the two values. This analysis example highlighted some interesting cases. For instance, removing a single security constraint may reduce the quantitative resiliency significantly but with little affect on the distance resiliency. In one case it was shown a policy change reduces the quantitative resiliency to 0 yet the workflow still has high distance resiliency. Furthermore, a workflow may have 0 quantitative resiliency yet still be satisfiable. One further case showed how a change to the security policy does not change the resiliency values but does have the effect of reducing the size of the MDP_W model PRISM must evaluate in terms of state space, thereby reducing resiliency verification time.

- **Contribution 3.** It is unlikely that any workflow executed by human users will have full resiliency, therefore it is desirable to manage the risk of a workflow deadlocking or terminating unsuccessfully due to the unavailability of those users. In Chapter 5 we suggested this risk can be reduced by re-computing at each step the expected quantitative resiliency, in order to adjust task assignments so resiliency is maximised in accordance with the current prediction of user availability. In a user driven execution model (UDEM), where users initiate requests to be assigned the execution of tasks at runtime, quantitative resiliency computation could ensure a request is granted only if the rest of the workflow has a resiliency value above a given threshold. Computing quantitative resiliency is a computationally demanding process hence evaluating quantitative resiliency for assignments at runtime has itself an impact on workflow execution time. We have used the observation from Chapter 4, that policy modifications can be made without affecting resiliency, by performing empirical analysis on a workflow by incrementally adding constraints to its security policy. These and further results indicated that adding or removing security components to the security policy of a workflow has a clear impact on the resiliency verification time which can either increase or decrease. The results also showed that the same resiliency value can in some cases be maintained even with the addition or removal of several restrictive security constraints. We have considered the addition of separation of duty constraints which define sets of tasks that cannot be executed by the same user in a workflow instance, and the removal of authorisation constraints which define which users can be assigned the execution of which tasks. We have provided a methodology for a CISO to analyse and change a security policy to improve the verification time for quantitative resiliency at runtime by adding or removing a set of dummy, or artificial, restrictive security constraints that do not impact the actual resiliency value. Our observations showed the gain in time can be significant, for instance in one example the verification time was reduced from 6.53 seconds to 0.63 seconds following the removal of three authorisation constraints while the resiliency value remained the same.

- **Contribution 4.** Workflows that deadlock due to user unavailability, but must complete, are typically managed by performing mitigation actions such as emergency security overrides which allow those users who are available to complete the remaining tasks. Calculating the quantitative resiliency of a workflow can aid the formation of mitigation strategies, for instance a workflow with very high resiliency (low failure risk) may need only low cost, emergency type actions, whereas a workflow with low resiliency (high failure risk) may require more costly long-term mitigation actions.

Understanding the requirements of a mitigation strategy for a workflow with a single execution path can be straightforward as a single resiliency value is generated. Workflows with choice however can consist of many possible execution paths, each giving potentially different resiliency values making the formation of mitigation strategies complex. In Chapter 5 we have given a definition for a workflow with choice and shown how it can be reduced to a set of workflows without choice in order to calculate the quantitative resiliency of each execution path using the techniques described in Chapters 3 and 4. We have introduced a new metric for analysing policy modification impact in workflows with choice called *resiliency variance* which indicates an overall resiliency variability, or volatility, from the resiliency average. Several other resiliency measures have been considered for a workflow with choice; *resiliency extrema* which are the maximum and minimum resiliency values; *resiliency distribution* which the set of all resiliency values for all execution paths; and *expected resiliency* which is the average of the resiliency distribution. We then showed through example how a workflow with different predictions of user availability can have the same expected resiliency but significantly different measures of resiliency variance, meaning expected resiliency may be misleading when used alone. We have also discussed how resiliency variance could be used for predicting a suitable workflow mitigation strategy for a workflow with choice.

8.1.2 Problem 2

CISOs and human factors experts need to record and incorporate their security and human behavioural knowledge directly within the structure of an ontology.

- **Contribution 5.** Currently the construction and modification of an ontology aligning information security and human behavioural factors requires the use of a generic ontology development tool. This makes the creation of an ontology a complex process as existing tools assume a familiarity with ontology technologies and are aimed at ontology experts, and not those whose knowledge requires incorporating into an ontology. As such, an information security domain expert may be unable to develop ontology content themselves and so, requires dedicated ontology development tools that hide ontology complexity. In Chapter 7 we have outlined the requirements and high-level implementation details for a prototype graphical ontology development tool for information security domain experts. The tool requirements were extracted from discussions conducted with two CISOs from large organisations. The tool has

been designed as a standalone application imagining a scenario with group-based, face-to-face collaboration. It has been implemented in Visual Basic [129], and embeds the Microsoft Visio 2007's drawing control [128], thus providing a canvas on which users can construct visual representations of an information security ontology using a palette of pre-configured shapes and connectors. Knowledge fragments can be written by a user within the text box of appropriate shapes representing ontology concepts and connected together in a 'drag and drop' approach to form relationships according to an underlying security ontology structure. Ontology diagrams once constructed can be automatically translated via a Java implemented program to the machine readable Web Ontology Language (OWL) [170], for further analysis in a suitable tool. The tool's design removes the need to understand ontology construction techniques allowing collaborating information security domain experts to construct and modify, share, and analyse a security ontology at an abstract level.

- **Contribution 6.** An information security ontology constructed in a distributed fashion by multiple experts would provide a larger and more accurate repository of information security knowledge. Collaboration of this kind allows CISOs and human factors researchers to capture, integrate, publish and share their knowledge with peers and colleagues within the information security domain and reach a general consensus. The Web is a natural platform for collaboration and knowledge sharing by distributing the development process and the resulting knowledge base to the entire information security community. In Chapter 7 we outlined the requirements and high-level implementation details for a second prototype tool for information security domain experts. Additional tool requirements were extracted from the same semi-structured consultations with two CISOs mentioned above. The second tool is Web-oriented for use in scenarios of distributed knowledge entry and ontology construction. The tool has extended the Web based, open source ontology development tool Web-Protégé [172], to make ontology development approachable both to CISOs and human factors experts by hiding the complexities of the underlying ontology. Community users record and share their knowledge in a text-based fashion within the structure of a pre-defined information security ontology which is then automatically translated and stored as an OWL encoded ontology file. Ontology files created in the graphical ontology development tool can also be imported for analysis purposes or text based editing. Several pages are provided for the user to simplify knowledge entry, structuring and analysis, one for each major concept in the ontology (e.g. 'assets', 'threats', 'vulnerabilities', etc.). The tool supports several collaborative features allowing knowledge recording and incorporation

from across a number of participating organisations. This includes a notes portal that allows users to annotate, discuss and reach consensus on ontology content by posting messages. Tool evaluation was carried out by the two CISOs consulted previously and met with general praise for both its appearance and functionality. The evaluation has also provided useful feedback on tool design and functionality, and several avenues for future work.

8.2 Future Work

In this section we reflect on the research outcomes and outline possible future work.

8.2.1 Workflow Resiliency Analysis

- **Workflow Complexity.** The workflow examples used throughout this thesis are completely synthetic in terms of tasks structure, security policy and users. Example size has also been arbitrarily limited to workflows with a maximum of 10 tasks and 5 users for ease of understanding our approach. A natural next step is to integrate our approach into a real-life workflow execution setting to evaluate its feasibility and effectiveness. We have had some interest in this regard from the research division of SAP who is a multinational software corporation that develops and manages software to manage business operations such as workflows[1]. From a research perspective, we have used synthetic examples due to the difficulty in accessing real-life workflow examples with enough level of detail for our purposes. This is understandable as organisations intuitively do not want to release details of key business processes and workflows at the level of abstraction we require. Some data is available regarding 'typical' workflow structures, for instance in banking, insurance and purchasing which could provide a good starting point for more complex workflow models however security and user details may still require fabrication. It would be favourable to establish contact with a number of organisations to assess the possibility of a case study in order to validate our approach. As alluded to in the introduction, organisations can have different definitions of a workflow which need to be investigated in more detail to establish whether our workflow model is powerful enough to capture these definitions. Certainly it would be interesting to consider more complex workflows, for instance with a combination of choice and parallel control patterns, probabilistic parallel task ordering, and the

[1] https://www.sap.com/index.html

introduction of loops meaning the same task or group of tasks may be performed more than once. Loops would introduce another level of non-determinism in the execution of the workflow itself, and present as such some challenging aspects in their analysis. We made a simple assumption that tasks are viewed as 'black boxes' such that if a task can be assigned it will be completed. It would be interesting to open up these black boxes and perhaps consider other user attributes which could affect the success rate of a workflow such as capability.

- **Security Complexity.** As mentioned above, the security policies considered in our examples are synthetic in terms of their composition. Although they incorporate the main security constraints mentioned in the literature: authorisation constraints, and separations and bindings of duty, it is unclear whether the 'level' of security we apply is too little or too much in comparison to real-life security policies at this level of abstraction. We do aim to introduce more complex security constraints including cardinality, restricting the number of times a user can be assigned a specific task. Currently, authorisation constraints are stated on an individual level with an assumption that all users permitted to perform one task are assigned to the same role. It would be interesting and perhaps more realistic to consider a more role-based approach by introducing role related constraints, for instance a user may be permitted to take on a limited number of roles in a single workflow instance [32]. Incorporating roles would mean constraints such as separations of duty can be stated at both the user and the role level as defined in [38]. Adding role hierarchies would allow 'no read up' and 'no write down' policies to be introduced akin to the Bell-LaPadula security model [15, 181]. Adding such a complex security model will undoubtedly affect resiliency verification time which would need investigation to assess the viability of our approach as a workflow resiliency analysis tool.

- **Solution Complexity.** It is important to note that the initial focus of this work has been on proposing a novel approach to providing metrics for the satisfiability and resiliency of a workflow, rather than providing a necessarily efficient solution of these problems. We propose to carry out a full theoretical analysis on the complexity of our approach of using model checking of Markov decision process models. The main challenge in taking a model checking approach is dealing with the state space explosion problem [34]. This may currently inhibit our approach's applicability for resiliency calculated at runtime as suggested in Chapter 5 when a timely response is expected, or during offline resiliency analysis of large scale complex workflows. The state space

problem is an active area of research and considerable progress is being made, e.g. [33]. We believe however our approach paves the way to defining an efficient solution by using the extensive literature dedicated to the efficient solving of a Markov decision process, e.g. [105]. It would be interesting to carry out some comparison analysis in terms of computation time between our quantitative approach to solving the workflow satisfiability problem against other proposed algorithms, e.g., [41]. Another lead is the study of sub-optimal policies. Indeed, calculating a sub-optimal solution might be more tractable [30], at the cost of a loss of accuracy. In this case, it could be worth understanding the impact on workflow satisfiability and resiliency when using a sub-optimal solution. With regard to our approach in Chapter 5, to reduce resiliency verification time by adding dummy or redundant constraints to the security policy, some more investigation is required to understand the implications of this in terms of its practicality. As a side note, another application of this approach could be to identify redundant security constraints which can be removed without changing the resiliency of a workflow. For instance in a similar way to [99], the minimum number of required authorisation constraints could be found to satisfy a workflow and maintain its resiliency at a certain threshold. Another approach to reducing complexity would be to revisit our encoding of the workflow assignment process in PRISM and explore how this could be made more efficient, for instance reducing the number of variables and therefore the size of the model that must be built before verifying resiliency. We are in discussions with one of the main PRISM developers in this regard.

- **Model Checker.** Clearly, the verification of resiliency currently relies on our encoding but also on the model checking tool PRISM. Some more investigation is required to understand the capabilities of PRISM in terms of the workflows PRISM can handle, not only in terms of the number of tasks but also the complexity of the security policy. Our initial analysis carried out in Chapter 5 suggests a workflow with a large number of tasks and few security constraints may be less complex to analyse than a workflow with fewer tasks but more security constraints. It would therefore be favourable to define a notion of complexity for a workflow indicating workflow configurations which are tractable in terms of resiliency computation. Some comparison in terms of performance is also needed between PRISM and other 'off-the-shelf' probabilistic model checking tools capable of providing efficient and automatic solutions to MDPs. Some attractive alternatives are RAPTURE [88], which uses a Communicating Sequential Processes (CSP) type language [80], PASS [73] which uses a variant of PRISM's modelling language, and the Möbius modelling environment which offers a graphical interface for

model construction [47]. Some exploration is also required regarding the computing platform on which these tools are run, for instance comparing the computation time for a standard desktop deployment against one running on an external cloud platform.

- **User Availability.** To assess our approach 'in the wild' it is necessary to integrate it into a real life workflow management system such that task execution assignment decisions are based partly on resiliency calculations. Central to these resiliency calculations is the data predicting the future availability of users. In the case of a UDEM system this would require predicted user availability data to be constantly updated in order for optimal assignment decisions which maximise resiliency. It is favourable then to understand in more detail how such data is, or can be derived, what the practicalities are of collecting such data, and how reliable it is. This would involve a survey of current business techniques from workforce planning, statistical analysis and employee management used to predict staffing requirements and levels of absenteeism. One avenue for a fully automated UDEM system would be to look at workflow operational logs and other digital data sources to identify possible automated and more efficient and timely data collection strategies [24].

- **Risk Acceptance.** Our approach to manage workflow failure risk through mitigation strategies requires formalisation. By applying cost to mitigation actions and modelling them in the state of our MDP_W encoding it would be possible to find an optimal mitigation strategy. Policy overrides should be straightforward to implement by applying a cost (or reward in the case of the MDP_W) to transitions that assign a task but violate the security policy. Other actions such as skipping tasks and forward execution will not be so trivial to model. It should be possible to model complex mitigation action strategies such that each task has a different mix of allowable actions and where some may not have any. Furthermore restrictions could be placed on which users can perform which mitigation actions at each assignment step. Several avenues of mitigation strategy analysis can then be taken. For instance by finding the cheapest mitigation strategy that provides a required threshold of resiliency, or finding the maximum level of expected resiliency that can be achieved for a given mitigation strategy budget. In terms of workflows with choice we have not yet considered computing resiliency variance for workflows where choice is probabilistic. Furthermore we have yet to analyse how resiliency variance is impacted in a workflow with choice when modifying the workflow's security policy.

- **Analysis Tools.** An interesting point is to develop useable analysis tools for a CISO to compute the predicted resiliency of a workflow when modifying its security policy. It is unlikely that a CISO would have the expertise, time or inclination to employ the formal techniques presented in Part 1 of this thesis as they stand. Indeed, the presentation of the work in this part is somewhat contradictory in some sense to the motivation for the ontology development tools presented in Part 2. The ontology development tools were implemented to hide the complexities of the underlying ontology methods and technologies. For usability purposes there is a necessity to encapsulate the resiliency analysis techniques into tools for a CISO whilst abstracting away unnecessary details such as MDPs and PRISM encodings. Such tools would instead enable a CISO to concern themselves with analysing the impact of security policy modifications and produce different resiliency metrics for inclusion in business cases. An attractive approach would be to implement a graphical tool that allows a CISO to construct workflow diagrams expressing tasks and security constraints with a palette of pre-configured shapes. Once diagrams are completed, perhaps in collaboration with a workflow designer or business analyst, resiliency analysis can be performed once the diagram is automatically encoded into PRISM. We have a prototype command line tool for this which takes some simple workflow and security policy parameters, automatically generates the PRISM encoding, and initiates PRISM to perform the resiliency analysis before writing the results to a text file. Indeed it may be possible to take a similar approach to the implementation of the ontology development tools outlined in Chapter 7, that is, either create a new standalone resiliency analysis tool, or embed the resiliency analysis techniques into a pre-existing workflow design tool modified specifically for a CISO. Another useful feature of a workflow resiliency analysis tool would be computing the optimal security policy modifications, that is all the modifications that can be made to the policy without reducing workflow resiliency below a given threshold.

8.2.2 Ontology Development

- **Tool Functionality.** We will consider several suggestions from the participating CISOs, that is extensions to the underlying ontology structure, and the issue raised of potential incentives for motivating members of the information security and human factors research communities to contribute to developing a security ontology. With further development work we will also look to address concerns raised about the control

and quality of content through enhancement of the Web-oriented tool's collaborative features. This may include provision of a voting system or content-rating scheme. Another avenue is to improve the analysis capabilities of the graphical tool, perhaps with a search tool that highlights a particular concept in the diagram. Ontology development is only 'one-way' at present meaning ontology diagrams are translated to OWL encoded ontology files and not vice-versa. Also the size of the tool's canvas may inhibit the size of the ontologies constructed, for instance an ontology with lots of concepts and relationships could soon become quite confused, necessitating a 'clean up' feature that automatically rearranges and tidies the diagram content. The Microsoft Visio drawing surface embedded in the tool comes with some automatic realignment functions that could be used for this purpose [128]. Another possibility could be to develop the security ontology in parts and automatically join them using the merge functions provided by the Java OWL API [81].

- **Tool Validation.** Currently the tools have not be validated in terms of analysing potential impact modifying security policies has on the behaviour of users, and therefore on the success rate of workflows. Clearly the effectiveness of the developed ontology for such analysis relies on the data within it which may be difficult to obtain in order to provide a comprehensive knowledge base. Studies have been conducted to identify normalised user behaviour when interacting with security policies related to USB sticks, passwords and access control which may be a useful starting point for populating a base security ontology (see Section 6.1.4). An assessment of the tools' effectiveness in supporting the recording and sharing of information security knowledge by CISOs and human factors researchers would require a prolonged study. However we aim to engage with more CISOs and human factor researchers with a scope to deploying our tools within their organisations to help us validate our assumptions relating to how security experts would participate in the collaborative development of an information security ontology, and use that ontology to analyse the impact security policy modifications have on the success rate of workflows.

References

[1] Agarwal, J. (2015). Improving resilience through vulnerability assessment and management. *Civil Engineering and Environmental Systems*, 32(1-2):5–17.

[2] Alfawaz, S., Nelson, K., and Mohannak, K. (2010). Information security culture: A behaviour compliance conceptual framework. In *Proceedings of the 8th Australasian Conference on Information Security*, AISC'10, pages 47–55.

[3] Armando, A., Giunchiglia, E., and Ponta, S. E. (2009). Formal specification and automatic analysis of business processes under authorization constraints: An action-based approach. In *Proceedings of the 6th International Conference on Trust, Privacy and Security in Digital Business*, TrustBus'09, pages 63–72.

[4] Armando, A. and Ponta, S. E. (2010). Model checking of security-sensitive business processes. In *Proceedings of the 7th International Workshop on Formal Aspects in Security and Trust*, FAST'09, pages 66–80.

[5] Armando, A. and Ponta, S. E. (2014). Model checking authorization requirements in business processes. *Computers & Security*, 40:1–22.

[6] Atluri, V. and Warner, J. (2008). Security for workflow systems. In *Handbook of Database Security: Applications and Trends*, pages 213–230. Springer.

[7] Auer, S., Dietzold, S., and Riechert, T. (2006). OntoWiki - a tool for social, semantic collaboration. In *Proceedings of the 5th International Conference on The Semantic Web*, ISWC'06, pages 736–749.

[8] AXELOS (2015). Information Technology Information Library. https://www.axelos.com/best-practice-solutions/itil. Accessed: 04-07-2015.

[9] Barletta, M., Ranise, S., and Vigano, L. (2009). Verifying the interplay of authorization policies and workflow in service-oriented architectures. In *Proceedings of the 12th International Conference on Computational Science and Engineering*, CSE'09, pages 289–296.

[10] Bartsch, S. and Sasse, M. A. (2013). How users bypass access control and why: Impact of authorization problems on individuals and the organization. In *Proceedings of the 21st European Conference on Information Systems*, ECIS'13, page 53.

[11] Basin, D., Burri, S., and Karjoth, G. (2011). Obstruction-free authorization enforcement: Aligning security with business objectives. In *Proceedings of the 24th IEEE Computer Security Foundations Symposium*, CSF'11, pages 99–113.

[12] Basin, D., Burri, S. J., and Karjoth, G. (2012). Optimal workflow-aware authorizations. In *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies*, SACMAT'12, pages 93–102.

[13] Basu, A. and Kumar, A. (2002). Research commentary: Workflow management issues in e-business. *Information Systems Research*, 13(1):1–14.

[14] Bauer, L., Cranor, L. F., Reeder, R. W., Reiter, M. K., and Vaniea, K. (2009). Real life challenges in access-control management. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI'09, pages 899–908.

[15] Bell, D. E. and LaPadula, L. J. (1973). MTR-2547-VOL-1: Secure computer systems: Mathematical foundations. Technical report, MITRE Corporation.

[16] Bellman, R. (1957). A Markovian decision process. *Indiana University Mathematics Journal*, 6(4):679–684.

[17] Bertino, E., Brodie, C., Calo, S., Cranor, L., Karat, C., Karat, J., Li, N., Lin, D., Lobo, J., Ni, Q., Rao, P., and Wang, X. (2009). Analysis of privacy and security policies. *IBM Journal of Research and Development*, 53(2):1–18.

[18] Bertino, E., Ferrari, E., and Atluri, V. (1999). The specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and Systems Security*, 2(1):65–104.

[19] Blanco, C., Lasheras, J., Valencia-Garcia, R., Fernandez-Medina, E., Toval, A., and Piattini, M. (2008). A systematic review and comparison of security ontologies. In *Proceedings of the 3rd International Conference on Availability, Reliability and Security*, ARES'08, pages 813–820.

[20] Botha, R. and Eloff, J. H. P. (2001). Separation of duties for access control enforcement in workflow environments. *IBM Systems Journal*, 40(3):666–682.

[21] British Standards Institution (2005a). BS ISO/IEC 27001:2005 - information technology - security techniques - information security management systems - requirements. http://www.bsigroup.com/en-GB/. Accessed: 04-07-2015.

[22] British Standards Institution (2005b). BS ISO/IEC 27002:2005 - information technology - security techniques - code of practice for information security management. http://www.bsigroup.com/en-GB/. Accessed: 04-07-2015.

[23] Brunel, J., Cuppens, F., Cuppens, N., Sans, T., and Bodeveix, J.-P. (2007). Security policy compliance with violation management. In *Proceedings of the 5th ACM Workshop on Formal Methods in Security Engineering*, FMSE'07, pages 31–40.

[24] Cain, R. and van Moorsel, A. (2012). Optimization of data collection strategies for model-based evaluation and decision-making. In *IEEE/IFIP International Conference on Dependable Systems and Networks*, DSN'12, pages 1–10.

[25] Calinescu, R., Ghezzi, C., Kwiatkowska, M., and Mirandola, R. (2012). Self-adaptive software needs quantitative verification at runtime. *Communications of the ACM*, 55(9):69–77.

[26] Calinescu, R., Grunske, L., Kwiatkowska, M., Mirandola, R., and Tamburrelli, G. (2011). Dynamic QoS management and optimisation in service-based systems. *IEEE Transactions on Software Engineering*, 37(3):387–409.

[27] Cañas, A. J., Carvajal, R., Carff, R., and Hill, G. (2004). IHMC CmapTools 2004-01: CmapTools, web pages & websites. Technical report, Florida Institute for Human and Machine Cognition.

[28] Cao, L. (2014). Behavior informatics: A new perspective. *IEEE Intelligent Systems*, 29(4):62–80.

[29] Casati, F., Ceri, S., Paraboschi, S., and Pozzi, G. (1999). Specification and implementation of exceptions in workflow management systems. *ACM Transactions on Database Systems*, 24(3):405–451.

[30] Cassandra, A. R. (1994). CS-94-14: Optimal policies for partially observable Markov decision processes. Technical report, Brown University.

[31] Caulfield, T. and Pym, D. (2015). Improving security policy decisions with models. *IEEE Security & Privacy*, 13(5):34–41.

[32] Chen, F. and Sandhu, R. S. (1996). Constraints for role-based access control. In *Proceedings of the 1st ACM Workshop on Role-based Access Control*, RBAC'95. Article 14.

[33] Clarke, E. M., Grumberg, O., Jha, S., Lu, Y., and Veith, H. (2001). *Progress on the State Explosion Problem in Model Checking*. Springer-Verlag.

[34] Clarke, E. M., Klieber, W., Nováček, M., and Zuliani, P. (2012). *Model Checking and the State Explosion Problem*. Springer Berlin Heidelberg.

[35] Cohen, D., Crampton, J., Gagarin, A., Gutin, G., and Jones, M. (2014a). Engineering algorithms for workflow satisfiability problem with user-independent constraints. In *Proceedings of the 8th International Workshop on Frontiers in Algorithmics*, FAW'14, pages 48–59.

[36] Cohen, D., Crampton, J., Gutin, G., and Jones, M. (2014b). Pattern-based plan construction for the workflow satisfiability problem. *Artificial Intelligence Research*, 51:555–577.

[37] Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on Theory of Computing*, STOC'71, pages 151–158.

[38] Crampton, J. (2003). Specifying and enforcing constraints in role-based access control. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies*, SACMAT'03, pages 43–50.

[39] Crampton, J. (2005). A reference monitor for workflow systems with constrained task execution. In *Proceedings of the 10th ACM Symposium on Access Control Models and Technologies*, SACMAT'05, pages 38–47.

[40] Crampton, J., Gutin, G., and Karapetyan, D. (2015). Valued workflow satisfiability problem. In *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies*, SACMAT'15, pages 3–13.

[41] Crampton, J., Gutin, G., and Yeo, A. (2013). On the parameterized complexity and kernelization of the workflow satisfiability problem. *ACM Transactions on Information and System Security*, 16(1):1–31.

[42] Crampton, J. and Khambhammettu, H. (2008). On delegation and workflow execution models. In *Proceedings of the 23rd ACM Symposium on Applied Computing*, SAC'08, pages 2137–2144.

[43] Crampton, J. and Morisset, C. (2011). An auto-delegation mechanism for access control systems. In *Proceedings of the 6th International Workshop on Security and Trust Management*, STM'11, pages 1–16.

[44] Dalkir, K. (2011). *Knowledge Management in Theory and Practice*. MIT Press.

[45] Damodaran, A. (2007). *Strategic Risk Taking: A Framework for Risk Management*. Wharton School Publishing.

[46] Davenport, T. H. and Prusak, L. (1997). *Working Knowledge: How Organizations Manage What They Know*. Harvard Business School Press.

[47] Deavours, D. D., Clark, G., Courtney, T., Daly, D., Derisavi, S., Doyle, J. M., Sanders, W. H., and Webster, P. G. (2002). The Möbius framework and its implementation. *IEEE Transactions on Software Engineering*, 28(10):956–969.

[48] Diver, S. (2004). Information security policy: A development guide for large and small companies. https://www.sans.org/reading-room/whitepapers/policyissues/information-security-policy-development-guide-large-small-companies-1331. Accessed: 13-07-2015.

[49] Domingos, D., Rito-Silva, A., and Veiga, P. (2003). Authorization and access control in adaptive workflows. In *Proceedings of the 8th European Symposium on Research in Computer Security*, ESORICS'03, pages 23–38.

[50] Donner, M. (2003). Toward a security ontology. *IEEE Security and Privacy*, 1(3):6–7.

[51] Doshi, P., Goodwin, R., Akkiraju, R., and Verma, K. (2004). Dynamic workflow composition using Markov decision processes. In *Proceedings of the IEEE International Conference on Web Services*, ICWS'04, pages 576–582.

[52] Downey, R. G. and Fellows, M. R. (1999). *Parameterized Complexity*. Springer.

[53] Eder, J. and Liebhart, W. (1996). Workflow recovery. In *Proceedings of the 1st IFCIS International Conference on Cooperative Information Systems*, COOPIS'96, pages 124–134.

[54] Ekelhart, A., Fenz, S., Klemen, M., and Weippl, E. (2007). Security ontologies: Improving quantitative risk analysis. In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences, 2007*, HICSS'07, pages 156–162.

[55] El Bakkali, H. (2012). Bypassing workflow satisfiability problem due to access control constraints. In *Proceedings of the 4th International Conference on Networked Digital Technologies*, NDT'12, pages 178–191.

[56] El Bakkali, H. (2013). Enhancing workflow systems resiliency by using delegation and priority concepts. *Digital Information Management*, 11(4):267–276.

[57] Elenius, D., Denker, G., Martin, D., Gilham, F., Khouri, J., Sadaati, S., and Senanayake, R. (2005). The OWL-S editor: A development tool for semantic web services. In *The Semantic Web: Research and Applications*, pages 78–92. Springer.

[58] Espinosa, E. D., Frausto, J., and Rivera, E. J. (2010). Markov decision processes for optimizing human workflows. *Service Science*, 2(4):245–269.

[59] Fatema, K. and Chadwick, D. (2014). Resolving policy conflicts: Integrating policies from multiple authors. In *Proceedings of the 4th International Workshop on Information Systems Security Engineering*, WISSE'14, pages 310–321.

[60] Federal Office of Information Security (2005). IT-Grundschutz. https://www.bsi.bund.de/EN/Topics/ITGrundschutz/itgrundschutz_node.html. Accessed: 07-07-2015.

[61] Fenz, S. and Ekelhart, A. (2009). Formalizing information security knowledge. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ASIACCS'09, pages 183–194.

[62] Fenz, S., Goluch, G., Ekelhart, A., Riedl, B., and Weippl, E. (2007). Information security fortification by ontological mapping of the ISO/IEC 27001 standard. In *Proceedings of the 13th Pacific Rim International Symposium on Dependable Computing, 2007*, PRDC'07, pages 381–388.

[63] Ferreira, A., Cruz-Correia, R., Antunes, L., Farinha, P., Oliveira-Palhares, E., Chadwick, D. W., and Costa-Pereira, A. (2006). How to break access control in a controlled manner. In *Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems*, CBMS'06, pages 847–854.

[64] Fillies, C. and Weichhardt, F. (2005). On ontology-based event-driven process chains. http://www.semtalk.com/pub/semtalkepk.pdf. Accessed: 07-07-2015.

[65] Financial Conduct Authority (2015). http://www.fca.org.uk/. Accessed: 04-07-2015.

[66] Forejt, V., Kwiatkowska, M., Norman, G., and Parker, D. (2011). Automated verification techniques for probabilistic systems. In *Proceedings of the 11th International School on Formal Methods for the Design of Computer, Communication and Software Systems*, SFM'11, pages 53–113.

[67] Gaaloul, K., Schaad, A., Flegel, U., and Charoy, F. (2008). A secure task delegation model for workflows. In *Proceedings of the 2nd International Conference on Emerging Security Information, Systems and Technologies*, SECURWARE'08, pages 10–15.

[68] Gao, A., Yang, D., Tang, S., and Zhang, M. (2005). Web service composition using Markov decision processes. In *Proceedings of the 6th International Conference on Advances in Web-Age Information Management*, WAIM'05, pages 308–319.

[69] Georgakopoulos, D., Hornick, M., and Sheth, A. (1995). An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–153.

[70] Giunchiglia, E. and Lifschitz, V. (1998). An action language based on causal explanation: Preliminary report. In *Proceedings of the 15th National Conference on Artificial Intelligence*, AAAI'98, pages 623–630.

[71] Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing. *Human-Computer Studies*, 43(5-6):907–928.

[72] Guarino, N. (1995). Formal ontology, conceptual analysis and knowledge representation. *Human-Computer Studies*, 43(5-6):625–640.

[73] Hahn, E. M., Hermanns, H., Wachter, B., and Zhang, L. (2010). PASS: Abstraction refinement for infinite probabilistic models. In *Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'10, pages 353–357.

[74] Hamming, R. W. (1991). *The Art of Probability: For Scientists and Engineers*. Addison-Wesley Longman Publishing Co., Inc.

[75] Hayes, P., Eskridge, T. C., Saavedra, R., Reichherzer, T., Mehrotra, M., and Bobrovnikoff, D. (2005). Collaborative knowledge capture in ontologies. In *Proceedings of the 3rd International Conference on Knowledge Capture*, K-CAP'05, pages 99–106.

[76] He, L., Huang, C., Duan, K., Li, K., Chen, H., Sun, J., and Jarvis, S. A. (2012). Modeling and analyzing the impact of authorization on workflow executions. *Future Generation Computer Systems*, 28(8):1177–1193.

[77] Herbert, L. and Sharp, R. (2013). Precise quantitative analysis of probabilistic business process model and notation workflows. *Computing and Information Science in Engineering*, 13(1):011007.

[78] Herzog, A., Shahmehri, N., and Duma, C. (2007). An ontology of information security. *Information Security and Privacy*, 1(4):1–23.

[79] Hiden, H., Woodman, S., Watson, P., and Cala, J. (2013). Developing cloud applications using the e-Science Central platform. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1983):20120085.

[80] Hoare, C. A. R. (1978). Communicating sequential processes. *Communications of the ACM*, 21(8):666–677.

[81] Horridge, M. and Bechhofer, S. (2011). The OWL API: A Java API for OWL ontologies. *Semantic Web*, 2(1):11–21.

[82] Horrocks, I. (2002). DAML+OIL: A reason-able web ontology language. In *Proceedings of the 8th International Conference on Extending Database Technology*, EDBT'02, pages 2–13.

[83] Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. MIT Press.

[84] Hussain, F., Lucas, C., and Ali, M. (2004). Managing knowledge effectively. *Knowledge Management Practice*, 5(1):1–12.

[85] I4 - International Information Integrity Institute (2015). https://i4online.com/. Accessed: 04-07-2015.

[86] Information Security Forum (2015). https://www.securityforum.org/. Accessed: 04-07-2015.

[87] Jakoubi, S., Tjoa, S., Goluch, S., and Kitzler, G. (2010). Risk-aware business process management: Establishing the link between business and security. In *Complex Intelligent Systems and Their Applications*, pages 109–135. Springer.

[88] Jeannet, B., D'Argenio, P., and Larsen, K. (2002). Rapture: A tool for verifying Markov decision processes. In *Proceedings of Tools Day, affiliated to 13th International Conference on Concurrency Theory*, CONCUR'02, pages 84–98.

[89] Jelenic, D. (2011). The importance of knowledge management in organizations - with emphasis on the balanced scorecard learning and growth perspective. In *Proceedings of the International Conference on Management, Knowledge and Learning*, MakeLearn'11, pages 33–43.

[90] Jensen, K., Kristensen, L. M., and Wells, L. (2007). Coloured Petri Nets and CPN tools for modelling and validation of concurrent systems. *Software Tools for Technology Transfer*, 9(3):213–254.

[91] Jisc (2015). https://www.jisc.ac.uk/. Accessed: 04-07-2015.

[92] Jordan, D., Evdemon, J., Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., et al. (2007). Web services business process execution language: Version 2.0. *OASIS standard*, 11(120):5.

[93] Joy, B., Steele Jr, G. L., Gosling, J., and Bracha, G. (1998). *The Java Language Specification*. Addison-Wesley.

[94] Khan, A. A. and Fong, P. W. L. (2012). Satisfiability and feasibility in a relationship-based workflow authorization model. In *Proceedings of the 17th European Symposium on Research in Computer Security*, ESORICS'12, pages 109–126.

[95] Kiepuszewski, B., ter Hofstede, A. H. M., and Bussler, C. (2000). On structured workflow modelling. In *Proceedings of the 12th International Conference on Advanced Information Systems Engineering*, CAiSE'00, pages 431–445.

[96] Kirlappos, I., Parkin, S., and Sasse, M. A. (2015). Shadow security as a tool for the learning organization. *ACM SIGCAS Computers and Society*, 45(1):29–37.

[97] Knorr, K. and Stormer, H. (2001). Modeling and analyzing separation of duties in workflow environments. In *Proceedings of the 16th International Conference on ICT Systems Security and Privacy Protection*, IFIP SEC'01, pages 199–212.

[98] Kohler, M., Liesegang, C., and Schaad, A. (2007). Classification model for access control constraints. In *Proceedings of the 26th IEEE International Performance, Computing, and Communications Conference*, IPCCC'07, pages 410–417.

[99] Kohler, M. and Schaad, A. (2008). Avoiding policy-based deadlocks in business processes. In *Proceedings of the 3rd International Conference on Availability, Reliability and Security*, ARES'08, pages 709–716.

[100] Kothari, V., Blythe, J., Smith, S. W., and Koppel, R. (2015). Measuring the security impacts of password policies using cognitive behavioral agent-based modeling. In *Proceedings of the 2015 Symposium and Bootcamp on the Science of Security*, HotSoS'15, pages 13:1–13:9.

[101] Krivov, S., Williams, R., and Villa, F. (2007). GrOWL: A tool for visualization and editing of OWL ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):54–57.

[102] Kumar, A., Van der Aalst, W. M. P., and Verbeek, E. M. W. (2002). Dynamic work distribution in workflow management systems: How to balance quality and performance. *Management and Information Systems*, 18(3):157–193.

[103] Kwiatkowska, M., Norman, G., and Parker, D. (2002). Probabilistic symbolic model checking with prism: A hybrid approach. In *Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS?02, pages 52–66.

[104] Kwiatkowska, M., Norman, G., and Parker, D. (2011). PRISM 4.0: Verification of probabilistic real-time systems. In *Proceedings of the 23rd International Conference on Computer Aided Verification*, CAV'11, pages 585–591.

[105] Lassaigne, R. and Peyronnet, S. (2015). Approximate planning and verification for large markov decision processes. *International Journal on Software Tools for Technology Transfer*, 17(4):457–467.

[106] Lawrence, P. (1997). The workflow reference model. In *Workflow Handbook*, pages 243–293. John Wiley & Sons, Inc.

[107] Le Berre, D. and Parrain, A. (2010). The Sat4j library, release 2.2. *Satisfiability, Boolean Modeling and Computation*, 7:59–64.

[108] LeMay, E., Ford, M., Keefe, K., Sanders, W., and Muehrcke, C. (2011). Model-based security metrics using ADversary VIew Security Evaluation (ADVISE). In *Proceedings of the 8th International Conference on the Quantitative Evaluation of Systems*, QEST'11, pages 191–200.

[109] Li, N., Wang, Q., and Tripunitara, M. (2009). Resiliency policies in access control. *ACM Transactions on Information and System Security*, 12(4):20:1–20:34.

[110] Lowalekar, M., Tiwari, R. K., and Karlapalem, K. (2009). Security policy satisfiability and failure resilience in workflows. In *The Future of Identity in the Information Society*, pages 197–210. Springer.

[111] Lupu, E. and Sloman, M. (1997). Conflict analysis for management policies. In *Proceedings of the 5th IFIP/IEEE International Symposium on Integrated Network Management*, IM'97, pages 430–443.

[112] Ma, C., Lu, G., and Qiu, J. (2009). Conflict detection and resolution for authorization policies in workflow systems. *Journal of Zhejiang University SCIENCE A*, 10(8):1082–1092.

[113] Mace, J. C., Morisset, C., and Van Moorsel, A. P. A. (2014). Quantitative workflow resiliency. In *Proceedings of the 19th European Symposium on Research in Computer Security*, ESORICS'14, pages 344–361.

[114] Mace, J. C., Morisset, C., and Van Moorsel, A. P. A. (2015a). Impact of policy design on workflow resiliency computation time. In *Proceedings of the 12th International Conference on the Quantitative Evaluation of Systems*, QEST'15, pages 244–259.

[115] Mace, J. C., Morisset, C., and Van Moorsel, A. P. A. (2015b). Modelling user availability in workflow resiliency analysis. In *Proceedings of the 3rd Symposium and Bootcamp on the Science of Security*, HotSoS'15. Article 7.

[116] Mace, J. C., Morisset, C., and Van Moorsel, A. P. A. (2015c). Resiliency variance in workflows with choice. In *Proceedings of the 7th International Workshop on Software Engineering for Resilient Systems*, SERENE'15, pages 128–143.

[117] Mace, J. C., Parkin, S. E., and Van Moorsel, A. P. A. (2010a). A collaborative ontology development tool for information security managers. In *Proceedings of the 4th ACM Symposium on Computer Human Interaction for Management of Information Technology*, CHIMIT'10. Article 5.

[118] Mace, J. C., Parkin, S. E., and Van Moorsel, A. P. A. (2010b). Ontology editing tool for information security and human factors experts. In *Proceedings of the 2nd International Conference on Knowledge Management and Information Sharing*, KMIS'10, pages 207–212.

[119] Mace, J. C., Van Moorsel, A. P. A., and Watson, P. (2011). The case for dynamic security solutions in public cloud workflow deployments. In *Proceedings of the 41st IEEE/IFIP International Conference on Dependable Systems and Networks Workshops*, DSN-W'11, pages 111–116.

[120] Marinovic, S., Craven, R., Ma, J., and Dulay, N. (2011). Rumpole: A flexible break-glass access control model. In *Proceedings of the 16th ACM Symposium on Access Control Models and Technologies*, SACMAT'11, pages 73–82.

[121] Markov, A. (1954). The theory of algorithms. *Trudy Matematicheskogo Instituta Steklova*, 42:3–375.

[122] Martinelli, F. and Morisset, C. (2012). Quantitative access control with partially-observable Markov decision processes. In *Proceedings of the 2nd ACM Conference on Data and Application Security and Privacy*, CODASPY'12, pages 169–180.

[123] Massacci, F., Paci, F., and Gadyatskaya, O. (2011). Dynamic resiliency to changes. In *Trustworthy Internet*, pages 213–220. Springer.

[124] Mathias, A. R. D. (1974). The order extension principle. In *Axiomatic set theory*, pages 179–183. American Mathematical Society.

[125] McLeod, S. (2008). Social roles. http://www.simplypsychology.org/social-roles.html. Accessed: 07-07-2015.

[126] Metalidou, E., Marinagi, C., Trivellas, P., Eberhagen, N., Skourlas, C., and Giannakopoulos, G. (2013). The human factor of information security: Unintentional damage perspective. In *Proceedings of the 3rd International Conference on Integrated Information*, IC-ININFO'13, pages 424–428.

[127] Meyer, J. F. and Sanders, W. H. (1993). Specification and construction of performability models. In *Proceedings of the 2nd International Workshop on Performability Modeling of Computer and Communication Systems*, PMCCS'93, pages 28–30.

[128] Microsoft Developer Network (2012). About using the Visio drawing control in your application. https://msdn.microsoft.com/en-us/library/office/ff765109.aspx. Accessed: 07-07-2015.

[129] Microsoft Developer Network (2015). Visual Basic 6.0 Resource Center. https://msdn.microsoft.com/en-us/vstudio/ms788229.aspx. Accessed: 07-07-2015.

[130] Microsoft Office Online (2015). Microsoft Visio. https://products.office.com/en-gb/visio/flowchart-software. Accessed: 07-07-2015.

[131] Miner, A. and Parker, D. (2004). Symbolic representations and analysis of large probabilistic systems. In *Validation of Stochastic Systems: A Guide to Current Research*, pages 296–338. Springer.

[132] Monakova, G., Brucker, A. D., and Schaad, A. (2012). Security and safety of assets in business processes. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC'12, pages 1667–1673.

[133] Musen, M. A. (2015). The Protégé project: A look back and a look forward. *AI Matters*, 1(4):4–12.

[134] Norman, D. A. (2009). The way I see it: When security gets in the way. *ACM Interactions*, 16(6):60–63.

[135] Noy, N. F. and McGuinness, D. L. (2001). SMI-2001-0880: Ontology development 101: A guide to creating your first ontology. Technical report, Stanford Medical Informatics.

[136] Object Management Group (2011). Business Process Model and Notation (BPMN): Version 2.0. http://http://www.omg.org/spec/BPMN/2.0/PDF/. Accessed: 13-07-2015.

[137] Obrst, L., Chase, P., and Markeloff, R. (2012). Developing an ontology of the cyber security domain. In *Proceedings of the 7th International Conference on Semantic Technologies for Intelligence, Defense, and Security*, STIDS'12, pages 49–56.

[138] O'Leary, D. E. (2000). *Enterprise Resource Planning Systems: Systems, Life Cycle, Electronic Commerce, and Risk*. Cambridge University Press.

[139] Paci, F., Ferrini, R., Sun, Y., and Bertino, E. (2008). Authorization and user failure resiliency for WS-BPEL business processes. In *Proceedings of the 6th International Conference of Service-Oriented Computing*, ICSOC'08, pages 116–131.

[140] Parkin, S., Van Moorsel, A. P. A., Inglesant, P., and Sasse, M. A. (2010). A stealth approach to usable security: Helping IT security managers to identify workable security solutions. In *Proceedings of the 2010 Workshop on New Security Paradigms*, NSPW '10, pages 33–50.

[141] Parkin, S. E., Kassab, R. Y., and Van Moorsel, A. P. A. (2008). The impact of unavailability on the effectiveness of enterprise information security technologies. In *Proceedings of the 5th International Service Availability Symposium*, ISAS'08, pages 43–58.

[142] Parkin, S. E., Van Moorsel, A. P. A., and Coles, R. (2009). An information security ontology incorporating human-behavioural implications. In *Proceedings of the 2nd International Conference on Security of Information and Networks*, SIN '09, pages 46–55.

[143] PCI Security Standards Council (2015). https://www.pcisecuritystandards.org/. Accessed: 04-07-2015.

[144] Post, G. V. and Kagan, A. (2007). Evaluating information security tradeoffs: Restricting access can interfere with user tasks. *Computers and Security*, 26(3):229–237.

[145] Povey, D. (1999). Optimistic security: A new access control paradigm. In *Proceedings of the 2nd Workshop on New Security Paradigms*, NSPW'99, pages 40–45.

[146] Pruesse, G. and Ruskey, F. (1994). Generating linear extensions fast. *SIAM Journal on Computing*, 23(2):373–386.

[147] Raskin, V., Hempelmann, C. F., Triezenberg, K. E., and Nirenburg, S. (2001). Ontology in information security: a useful theoretical foundation and methodological tool. In *Proceedings of the 4th Workshop on New Security Paradigms*, NSPW'01, pages 53–59.

[148] Reichert, M. and Dadam, P. (1998). $Adept_{Flex}$: Supporting dynamic changes of workflows without losing control. *Intelligent Information Systems*, 10(2):93–129.

[149] Reichert, M. and Weber, B. (2012). *Enabling flexibility in process-aware information systems: challenges, methods, technologies*. Springer Science & Business Media.

[150] Ruskey, F. (1992). Generating linear extensions of posets by transpositions. *Journal of Combinatorial Theory, Series B*, 54(1):77–101.

[151] Russell, N., van der Aalst, W., and ter Hofstede, A. (2006). Workflow exception patterns. In *Proceedings of the 18th International Conference on Advanced Information Systems Engineering*, CAiSE'06, pages 288–302.

[152] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-based access control models. *Computer*, 29(2):38–47.

[153] Sasse, A. M., Ashenden, D., Lawrence, D., Coles-Kemp, L., Flechais, I., and Kearney, P. (2007). Human vulnerabilities in security systems. http://www.ktn.qinetiq-tim.net/content/files/groups/humanvuln/HFWGWhitePaperfinal.pdf. Accessed: 13-07-2015.

[154] Schefer, S., Strembeck, M., Mendling, J., and Baumgrass, A. (2011). Detecting and resolving conflicts of mutual-exclusion and binding constraints in a business process context. In *Proceedings of On the Move to Meaningful Internet Systems*, OTM'11, pages 329–346.

[155] Schiavone, S., Garg, L., and Summers, K. (2014). Ontology of information security in enterprises. *Electronic Journal of Information Systems Evaluation*, 17(1):71–87.

[156] Singhal, A. and Wijesekera, D. (2010). Ontologies for modeling enterprise level security metrics. In *Proceedings of the 6th Annual Workshop on Cyber Security and Information Intelligence Research*, CSIIRW'10. Article 58.

[157] Skyrme, D. (2003). Knowledge management: Making sense of an oxymoron. http://www.skyrme.com/insights/22km.htm. Accessed: 07-07-2015.

[158] Souag, A., Salinesi, C., and Comyn-Wattiau, I. (2012). Ontologies for security requirements: A literature survey and classification. In *Proceedings of the The 2nd International Workshop on Information Systems Security Engineering*, WISSE'12, pages 61–69.

[159] Stanton, J. M., Stam, K. R., Mastrangelo, P., and Jolton, J. (2005). Analysis of end user security behaviors. *Computers and Security*, 24(2):124–133.

[160] Stoneburner, G., Goguen, A. Y., and Feringa, A. (2002). Sp 800-30: Risk management guide for information technology systems. Technical report, National Institute of Standards & Technology.

[161] Strembeck, M. and Mendling, J. (2010). Generic algorithms for consistency checking of mutual-exclusion and binding constraints in a business process context. In *Proceedings of On the Move to Meaningful Internet Systems*, OTM'10, pages 204–221.

[162] Strohmaier, M., Walk, S., Pöschko, J., Lamprecht, D., Tudorache, T., Nyulas, C., Musen, M. A., and Noy, N. F. (2013). How ontologies are made: Studying the hidden social dynamics behind collaborative ontology engineering projects. *Web Semantics: Science, Services and Agents on the World Wide Web*, 20:18–34.

[163] Sun, Y., Wang, Q., Li, N., Bertino, E., and Atallah, M. (2011). On the complexity of authorization in RBAC under qualification and security constraints. *IEEE Transactions on Dependable and Secure Computing*, 8(6):883–897.

[164] Sunstein, C. R. (1996). Social norms and social roles. *Columbia Law Review*, 96(4):903–968.

[165] Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press.

[166] TechRepublic (2015). http://www.techrepublic.com/. Accessed: 04-07-2015.

[167] Texas Action Group (2015). The Causal Calculator. http://www.cs.utexas.edu/users/tag/ccalc/. Accessed: 13-07-2015.

[168] The Apache Software Foundation (2015). The Apache Xerces project. http://xerces.apache.org/xerces2-j/. Accessed: 07-07-2015.

[169] The Law Society (2015). http://www.lawsociety.org.uk/. Accessed: 04-07-2015.

[170] The World Wide Web Consortium (W3C) (2004). OWL web ontology language overview. http://www.academia.edu/download/30759881/5.3-B1.pdf. Accessed: 04-07-2015.

[171] Tipton, H. F. and Krause, M. (2008). *Information Security Management Handbook*. Auerbach Publications.

[172] Tudorache, T., Vendetti, J., and Noy, N. F. (2008). Web-Protégé: A lightweight OWL ontology editor for the web. In *Proceedings of the 5th International Workshop on OWL: Experiences and Directions*, OWLED'08.

[173] Tuyikeze, T. and Pottas, D. (2010). An information security policy development life cycle. In *Proceedings of the South African Information Security Multi-Conference*, SAISMC'10, pages 165–176.

[174] Van der Aalst, W. M. (1998). The application of Petri Nets to workflow management. *Circuits, Systems, and Computers*, 8(1):21–66.

[175] Van der Aalst, W. M. and Jablonski, S. (2000). Dealing with workflow change: Identification of issues and solutions. *Computer Systems Science and Engineering*, 15(5):267–276.

[176] Van der Aalst, W. M., Ter Hofstede, A. H., Kiepuszewski, B., and Barros, A. P. (2003). Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51.

[177] Van der Aalst, W. M. P., Hirnschall, A., and Verbeek, H. M. W. E. (2002). An alternative way to analyze workflow graphs. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering*, CAiSE'02, pages 535–552.

[178] Wainer, J., Barthelmess, P., and Kumar, A. (2003). W-RBAC a workflow security model incorporating controlled overriding of constraints. *Cooperative Information Systems*, 12(4):455–485.

[179] Wang, Q. and Li, N. (2007). Satisfiability and resiliency in workflow systems. In *Proceedings of the 12th European Symposium on Research in Computer Security*, ESORICS'07, pages 90–105.

[180] Wang, Q. and Li, N. (2010). Satisfiability and resiliency in workflow authorization systems. *ACM Transactions on Information and System Security*, 13(4). Article 40.

[181] Watson, P. (2012). A multi-level security model for partitioning workflows over federated clouds. *Cloud Computing*, 1(1):1–15.

[182] Weber, B., Reichert, M., Wild, W., and Rinderle, S. (2005). Balancing flexibility and security in adaptive process management systems. In *Proceedings of On the Move to Meaningful Internet Systems: CoopIS, DOA, and ODBASE*, OTM'05, pages 59–76.

[183] Wen, Z. and Watson, P. (2013). Dynamic exception handling for partitioned workflow on federated clouds. In *Proceedings of the 5th IEEE International Conference on Cloud Computing Technology and Science*, CloudCom'13, pages 198–205.

[184] Whalen, T., Smetters, D., and Churchill, E. F. (2006). User experiences with sharing and access control. In *Proceedings of the Conference on Human Factors in Computing Systems (Extended Abstracts)*, CHI EA'06, pages 1517–1522.

[185] Wijs, A. and Bošnački, D. (2012). Improving GPU sparse matrix-vector multiplication for probabilistic model checking. In *Proceedings of the 19th International Workshop on Model Checking Software*, SPIN'12, pages 98–116.

[186] Wolter, C. and Meinel, C. (2010). An approach to capture authorisation requirements in business processes. *Requirements Engineering*, 15(4):359–373.

[187] Wolter, C., Miseldine, P., and Meinel, C. (2009). Verification of business process entailment constraints using SPIN. In *Proceedings of the 1st International Symposium on Engineering Secure Software and Systems*, ESSoS'09, pages 1–15.

[188] Wolter, C., Schaad, A., and Meinel, C. (2008). Task-based entailment constraints for basic workflow patterns. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, SACMAT'08, pages 51–60.

[189] Wu, C., Zhang, X., and Urban, C. (2013). A formal model and correctness proof for an access control policy framework. In *Proceedings of the 3rd International Conference on Certified Programs and Proofs*, CPP'13, pages 292–307.

[190] Yang, P., Xie, X., Ray, I., and Lu, S. (2014). Satisfiability analysis of workflows with control-flow patterns and authorization constraints. *IEEE Transactions on Services Computing*, 7(2):237–251.

[191] Yevseyeva, I., Morisset, C., Groß, T., and Van Moorsel, A. P. A. (2014). A decision making model of influencing behavior in information security. In *Proceedings of the 11th European Workshop on Performance Engineering*, EPEW'14, pages 194–208.

Appendix A

PRISM Encodings

A.1 Workflow Execution Specification WES_1

A full PRISM encoding of workflow execution specification WES_1, defined in Section 2.3.3, is given below:

```
//WORKFLOW SPECIFICATION
//task schema
module task_schema
t : [0..5] init 0;
[d] decision & !t1 → (t'=1);
[d] decision & t1 & !t2 → (t'=2);
[d] decision & t2 & !t3 → (t'=3);
[d] decision & t1 & !t4 → (t'=4);
[d] decision & t3 & t4 & !t5 → (t'=5);
[t] true → (t'=0);
endmodule

//users
module users
u : [0..4] init 0;
[d] true → (u'=1);
[d] true → (u'=2);
[d] true → (u'=3);
[d] true → (u'=4);
[t] true → (u'=0);
endmodule
```

```
//security policy
// authorisation constraints
formula a1 = u=1 & (t=1 | t=3 | t=5);
formula a2 = u=2 & (t=1 | t=2 | t=3 | t=4 );
formula a3 = u=3 & (t=2 );
formula a4 = u=4 & (t=4 | t=5);
formula a = (u=0 & x=1) | a1 | a2 | a3 | a4;

// separation of duty constraints
module sod1
us1 : [0..4] init 0;
fs1 : bool init false;
[e] (t=2 | t=3) & us1=0 →(us1'=u);
[e] (t=2 | t=3) & us1!=0 & u=us1 →(fs1'=true);
[e] (t!=2 & t!=3) | (us1!=0 & u!=us1) →true;
[t] true → (us1'=0)&(fs1'=true);
endmodule

module sod2
us2 : [0..4] init 0;
fs2 : bool init false;
[e] (t=2 | t=4) & us2=0 →(us2'=u);
[e] (t=2 | t=4) & us2!=0 & u=us2 →(fs2'=true);
[e] (t!=2 & t!=4) | (us2!=0 & u!=us2) →true;
[t] true → (us2'=0)&(fs2'=true);
endmodule

module sod3
us3 : [0..4] init 0;
fs3 : bool init false;
[e] (t=3 | t=4) & us3=0 →(us3'=u);
[e] (t=3 | t=4) & us3!=0 & u=us3 →(fs3'=true);
[e] (t!=3 & t!=4) | (us3!=0 & u!=us3) →true;
[t] true → (us3'=0)&(fs3'=true);
endmodule
```

```
module sod4
us4 : [0..4] init 0;
fs4 : bool init false;
[e] (t=4 | t=5) & us4=0 →(us4'=u);
[e] (t=4 | t=5) & us4!=0 & u=us4 →(fs4'=true);
[e] (t!=4 & t!=5) | (us4!=0 & u!=us4) →true;
[t] true → (us4'=0)&(fs4'=true);
endmodule

formula s = !fs1 & !fs2 & !fs3 & !fs4;

//binding of duty constraints
module bod1
ub1 : [0..4] init 0;
fb1 : bool init false;
[e] (t=1 | t=3) & ub1=0 →(ub1'=u);
[e] (t=1 | t=3) & ub1!=0 & u!=ub1 →(fb1'=true);
[e] (t!=1 & t!=3) | (ub1!=0 & u=ub1) →true;
[t] true → (ub1'=0)&(fb1'=true);
endmodule

formula b = !fb1;

// security policy
formula valid = a & s & b;

//EXECUTION SPECIFICATION
// execution schema
module execution_schema
x : [−1..5] init 1;
[e] x < 5 → (x'=x+1);
[e] x = 5 → (x'=−1);
endmodule
```

```
// availability  forecast  & plan
module plan
tx1 : [0..5] init 0;
ux1 : [0..4] init 0;
tx2 : [0..5] init 0;
ux2 : [0..4] init 0;
tx3 : [0..5] init 0;
ux3 : [0..4] init 0;
tx4 : [0..5] init 0;
ux4 : [0..4] init 0;
tx5 : [0..5] init 0;
ux5 : [0..4] init 0;

// availability  forecast
//user 1
[e] event & x=1 & u=1 →0.8:(tx1'=t)&(ux1'=u)+0.2:(tx1'=0);
[e] event & x=2 & u=1 →0.8:(tx2'=t)&(ux2'=u)+0.2:(tx2'=0);
[e] event & x=3 & u=1 →0.8:(tx3'=t)&(ux3'=u)+0.2:(tx3'=0);
[e] event & x=4 & u=1 →0.6:(tx4'=t)&(ux4'=u)+0.4:(tx4'=0);
[e] event & x=5 & u=1 →0.6:(tx5'=t)&(ux5'=u)+0.4:(tx5'=0);

//user 2
[e] event & x=1 & u=2→ 0.7:(tx1'=t)&(ux1'=u)+0.3:(tx1'=0);
[e] event & x=2 & u=2→ 0.8:(tx2'=t)&(ux2'=u)+0.2:(tx2'=0);
[e] event & x=3 & u=2→ 0.3:(tx3'=t)&(ux3'=u)+0.7:(tx3'=0);
[e] event & x=4 & u=2→ 0.9:(tx4'=t)&(ux4'=u)+0.1:(tx4'=0);
[e] event & x=5 & u=2→ 0.9:(tx5'=t)&(ux5'=u)+0.1:(tx5'=0);

//user 3
[e] event & x=1 & u=3→ 0.1:(tx1'=t)&(ux1'=u)+0.9:(tx1'=0);
[e] event & x=2 & u=3→ 0.9:(tx2'=t)&(ux2'=u)+0.1:(tx2'=0);
[e] event & x=3 & u=3→ 0.7:(tx3'=t)&(ux3'=u)+0.3:(tx3'=0);
[e] event & x=4 & u=3→ 0.7:(tx4'=t)&(ux4'=u)+0.3:(tx4'=0);
[e] event & x=5 & u=3→ 0.7:(tx5'=t)&(ux5'=u)+0.3:(tx5'=0);

//user 4
[e] event & x=1 & u=4→ 0.1:(tx1'=t)&(ux1'=u)+0.9:(tx1'=0);
[e] event & x=2 & u=4→ 0.1:(tx2'=t)&(ux2'=u)+0.9:(tx2'=0);
[e] event & x=3 & u=4→ 0.1:(tx3'=t)&(ux3'=u)+0.9:(tx3'=0);
[e] event & x=4 & u=4→ 0.4:(tx4'=t)&(ux4'=u)+0.6:(tx4'=0);
[e] event & x=5 & u=4→ 0.0:(tx5'=t)&(ux5'=u)+1:(tx5'=0);
endmodule
```

```
//PROCESS ACTIONS
//assignment decisions
formula null = t=0 & u=0;
formula tinR = (x=2&tx1=t) | (x=3&tx2=t) | (x=4&tx3=t) | (x=5&tx4=t) |(x=−1&tx5=t);
formula decision = x!=−1 & valid & sp=x−1 & (x=1 & null | tinR);

//assignment events
formula event = x!=−1 & sp=x−1 & !tinR;

//termination actions
module termination
[t] valid & complete →true;
[t] (x!=−1 & sp!=x−1 & valid) | (x=−1 & !complete) | (x!=1 & !valid) →true;
endmodule

formula complete = tx1!=0 & tx2!=0 & tx3!=0 & tx4!=0 & tx5!=0;
formula terminated = x!=1 & null;

//plan
formula t1 = tx1=1 | tx2=1 | tx3=1 | tx4=1 | tx5=1;
formula t2 = tx1=2 | tx2=2 | tx3=2 | tx4=2 | tx5=2;
formula t3 = tx1=3 | tx2=3 | tx3=3 | tx4=3 | tx5=3;
formula t4 = tx1=4 | tx2=4 | tx3=4 | tx4=4 | tx5=4;
formula t5 = tx1=5 | tx2=5 | tx3=5 | tx4=5 | tx5=5;

formula sp = (tx1/tx1) + (tx2/tx2) + (tx3/tx3) + (tx4/tx4) + (tx5/tx5);

//REWARD FUNCTIONS
rewards " resiliency "
[t] complete & valid : 1;
endrewards

rewards "distance"
[d] x > 1 : 1;
[t] complete & valid : 1;
endrewards
```

A.2 Workflow Execution Specification WES_{21}

A full PRISM encoding of workflow execution specification WES_{21}, defined in Section 3.2.3, is given below:

```
//WORKFLOW SPECIFICATION
//task schema
module task_schema
t : [0..2] init 0;
[d] decision & !t1 → (t'=1);
[d] decision & t1 & !t2 → (t'=2);
[t] true → (t'=0);
endmodule

//users
module users
u : [0..2] init 0;
[d] true → (u'=1);
[d] true → (u'=2);
[t] true → (u'=0);
endmodule

//security policy
// authorisation constraints
formula a1 = u=1 & t=2;
formula a2 = u=2 & (t=1 | t=2);
formula a = (u=0 & x=1) | a1 | a2;

// separation of duty constraints
module sod1
us1 : [0..2] init 0;
fs1 : bool init false;
[e] (t=1 | t=2) & us1=0 →(us1'=u);
[e] (t=1 | t=2) & us1!=0 & u=us1 →(fs1'=true);
[e] (t!=1 & t!=2) | (us1!=0 & u!=us1) →true;
[t] true → (us1'=0)&(fs1'=true);
endmodule

formula s = !fs1;

//binding of duty constraints
formula b = true;
```

```
// security policy
formula valid = a & s & b;

//EXECUTION SPECIFICATION
// execution schema
module execution_schema
x : [−1..2] init 1;
[e] x < 2 → (x'=x+1);
[e] x = 2 → (x'=−1);
endmodule

// availability forecast
module availability_forecast
//plan
tx1 : [0..2] init 0;
ux1 : [0..2] init 0;
tx2 : [0..2] init 0;
ux2 : [0..2] init 0;

// availability forecast
//user 1
[e] event & x=1 & u=1 →1:(tx1'=t)&(ux1'=u)+0:(tx1'=0);
[e] event & x=2 & u=1 →1:(tx2'=t)&(ux2'=u)+0:(tx2'=0);
//user 2
[e] event & x=1 & u=2 →1:(tx1'=t)&(ux1'=u)+0:(tx1'=0);
[e] event & x=2 & u=2 →1:(tx2'=t)&(ux2'=u)+0:(tx2'=0);
endmodule

//PROCESS ACTIONS
//assignment decisions
formula null = t=0 & u=0;
formula tinR = (x=2&tx1=t) | (x=−1&tx2=t);
formula decision = x!=−1 & valid & sp=x−1 & (x=1 & null | tinR);

//assignment events
formula event = x!=−1 & sp=x−1 & !tinR;
//assigned tasks
formula t1 = tx1=1 | tx2=1;
formula t2 = tx1=2 | tx2=2;
```

```
//termination actions
module termination
[t] valid & complete →true;
[t] (x!=−1 & sp!=x−1 & valid) | (x=−1 & !complete) | (x!=1 & !valid) →true;
endmodule

formula complete = tx1!=0 & tx2!=0;
formula terminated = x!=1 & null;

//plan
formula sp = (tx1/tx1) + (tx2/tx2);

//REWARD FUNCTIONS
rewards " resiliency "
[t] complete & valid : 1;
endrewards

rewards "distance"
[d] x > 1 : 1;
[t] complete & valid : 1;
endrewards
```

A.3 Workflow Execution Specification WES_{22}

A full PRISM encoding of workflow execution specification WES_{22}, defined in Section 3.2.3, is given below:

```
//WORKFLOW SPECIFICATION
//task schema
module task_schema
t : [0..2] init 0;
[d] decision & !t1 → (t'=1);
[d] decision & t1 & !t2 → (t'=2);
[t] true → (t'=0);
endmodule

//users
module users
u : [0..2] init 0;
[d] true → (u'=1);
[d] true → (u'=2);
[t] true → (u'=0);
endmodule

//security policy
// authorisation constraints
formula a1 = u=1 & t=2;
formula a2 = u=2 & (t=1 | t=2);
formula a = (u=0 & x=1) | a1 | a2;

// separation of duty constraints
module sod1
us1 : [0..2] init 0;
fs1 : bool init false;
[e] (t=1 | t=2) & us1=0 →(us1'=u);
[e] (t=1 | t=2) & us1!=0 & u=us1 →(fs1'=true);
[e] (t!=1 & t!=2) | (us1!=0 & u!=us1) →true;
[t] true → (us1'=0)&(fs1'=true);
endmodule

formula s = !fs1;

//binding of duty constraints
formula b = true;
```

```
//security policy
formula valid = a & s & b;

//EXECUTION SPECIFICATION
// execution schema
module execution_schema
x : [−1..2] init 1;
[e] x < 2 → (x'=x+1);
[e] x = 2 → (x'=−1);
endmodule

// availability forecast
module availability_forecast
//plan
tx1 : [0..2] init 0;
ux1 : [0..2] init 0;
tx2 : [0..2] init 0;
ux2 : [0..2] init 0;

// availability forecast
//user 1
[e] event & x=1 & u=1 →0.8:(tx1'=t)&(ux1'=u)+0.2:(tx1'=0);
[e] event & x=2 & u=1 →0.8:(tx2'=t)&(ux2'=u)+0.2:(tx2'=0);
//user 2
[e] event & x=1 & u=2 →0.6:(tx1'=t)&(ux1'=u)+0.4:(tx1'=0);
[e] event & x=2 & u=2 →0.7:(tx2'=t)&(ux2'=u)+0.3:(tx2'=0);
endmodule

//PROCESS ACTIONS
//assignment decisions
formula null = t=0 & u=0;
formula tinR = (x=2&tx1=t) | (x=−1&tx2=t);
formula decision = x!=−1 & valid & sp=x−1 & (x=1 & null | tinR);

//assignment events
formula event = x!=−1 & sp=x−1 & !tinR;

//termination actions
module termination
[t] valid & complete →true;
[t] (x!=−1 & sp!=x−1 & valid) | (x=−1 & !complete) | (x!=1 & !valid) →true;
endmodule
```

```
formula complete = tx1!=0 & tx2!=0;
formula terminated = x!=1 & null;

//plan
formula t1 = tx1=1 | tx2=1;
formula t2 = tx1=2 | tx2=2;

formula sp = (tx1/tx1) + (tx2/tx2);

//REWARD FUNCTIONS
rewards " resiliency "
[t] complete & valid : 1;
endrewards

rewards "distance"
[d] x > 1 : 1;
[t] complete & valid : 1;
endrewards
```

A.4 PRISM Model State Diagrams

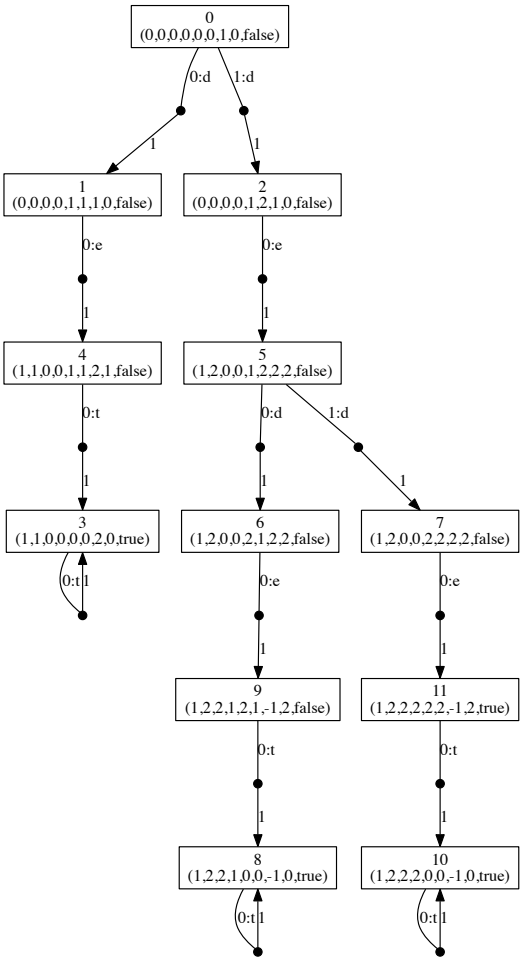


Fig. A.1 PRISM state diagram of MDP_W for workflow execution specification WES_{21}.

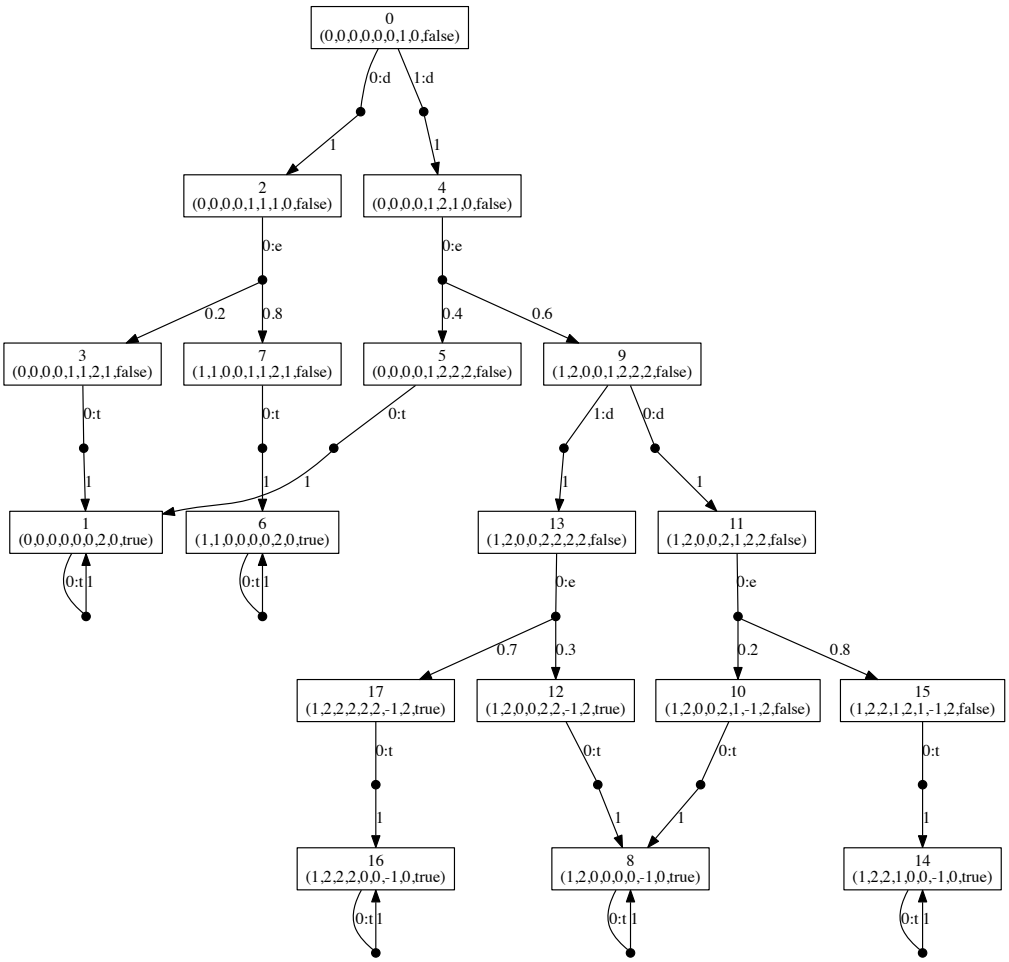


Fig. A.2 PRISM state diagram of MDP_W for workflow execution specification WES_{22}.

Appendix B

Experimental Data

B.1 Workflow Execution Specification WES_3

A full definition of the workflow execution specification $WES_3 = (WS_3, ES_3)$, introduced in Chapter 5 is given below:

Workflow specification $WS_3 = ((T_3, <_3), U_3, (A_3, S_3, B_3))$

- $T_3 = \{t_1, \ldots, t_{10}\}$
- $<_3 = \{(t_1, t_2), \ldots, (t_9, t_{10})$
- $U_3 = \{u_1, u_2, u_3, u_4, u_5\}$
- $A_3 = \emptyset$
- $S_3 = \emptyset$,
- $B_3 = \emptyset$

Execution specification $ES_3 = ((Z_3, \prec_3), \theta_3))$

- $Z_3 = \{x_1, \ldots, x_{10}, x_\perp\}$
- $\prec_3 = (x_1, \ldots, x_{10}, x_\perp)$
- θ_3 is shown in the table below were an entry $u_i \times x_j = \theta_3(x_j, x_i)$

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
u_1	0.93	0.99	0.85	0.87	0.91	0.84	0.95	0.86	0.97	0.99
u_2	0.84	0.89	0.82	0.87	0.81	0.97	0.99	0.85	0.98	0.81
u_3	0.97	0.92	0.84	0.95	0.96	0.84	0.86	0.96	0.83	0.83
u_4	0.93	0.97	0.82	0.99	0.99	0.98	0.92	0.93	0.99	0.80
u_5	0.97	0.88	0.94	0.83	0.93	0.84	0.93	0.87	0.82	0.88

B.2 Workflow Execution Specification WES_4

A full definition of the workflow execution specification $WES_4 = (WS_4, ES_4)$, introduced in Chapter 5 is given below:

Workflow specification $WS_4 = (((T_4, <_4), U_4, (A_4, S_4, B_4))$

- $T_4 = \{t_1, \ldots, t_5\}$
- $<_4 = \{(t_1,t_2), (t_1,t_4), (t_2,t_3), (t_3,t_5), (t_4,t_5)\}$
- $U_4 = \{u_1, u_2, u_3, u_4\}$
- $A_4 = \{(t_1,u_1), (t_1,u_2), (t_2,u_2), (t_2,u_3), (t_3,u_1), (t_3,u_2), (t_4,u_2), (t_4,u_4), (t_5,u_1), (t_5,u_4)\}$
- $S_4 = \{(t_2,t_4), (t_3,t_4), (t_4,t_5)\}$
- $B_3 = \{(t_1,t_2)\}$

Execution specification $ES_4 = ((Z_4, \prec_4), \theta_4))$

- $Z_4 = \{x_1, \ldots, x_5, x_\perp\}$
- $\prec_4 = (x_1, \ldots, x_5, x_\perp)$
- θ_4 is shown in the table below were an entry $u_i \times x_j = \theta_4(x_j, x_i)$

	x_1	x_2	x_3	x_4	x_5
u_1	0.96	0.83	0.72	0.72	0.71
u_2	0.86	0.92	0.80	0.81	0.95
u_3	0.83	0.86	0.77	0.72	0.71
u_4	0.72	0.90	0.95	0.84	0.81

B.3 Workflow Execution Specification WES_5

A full definition of the workflow execution specification $WES_5 = (WS_5, ES_5)$, introduced in Chapter 5 is given below:

Workflow specification $WS_5 = ((T_5, <_5), U_5, (A_5, S_5, B_5))$

- $T_5 = \{t_1, \ldots, t_{10}\}$
- $<_3 = \{(t_1, t_2), \ldots, (t_9, t_{10})$
- $U_5 = \{u_1, u_2, u_3, u_4, u_5\}$
- $A_5 = \{(u_4, t_1), (u_5, t_1), (u_1, t_2), (u_3, t_2), (u_2, t_3), (u_3, t_3), (u_5, t_3), (u_2, t_4),$ $(u_3, t_4), (u_4, t_4), (u_5, t_4), (u_1, t_5), (u_2, t_5), (u_3, t_5), (u_4, t_5), (u_1, t_6), (u_2, t_6),$ $(u_4, t_6), (u_5, t_6), (u_2, t_7), (u_4, t_7), (u_5, t_7), (u_2, t_8), (u_3, t_8), (u_1, t_9), (u_2, t_9),$ $(u_4, t_9), (u_4, t_{10}), (u_5, t_{10})\}$
- $S_5 = \{(t_1, t_4), (t_1, t_5), (t_1, t_8), (t_1, t_{10}), (t_2, t_9), (t_3, t_5), (t_3, t_6), (t_3, t_7), (t_3, t_9),$ $(t_4, t_{10}), (t_5, t_{10}), (t_6, t_8), (t_8, t_9), (t_8, t_{10}), (t_9, t_{10})\}$
- $B_5 = \emptyset$

Execution specification $ES_5 = ((Z_5, \prec_5), \theta_5))$

- $Z_5 = \{x_1, \ldots, x_{10}, x_\perp\}$
- $\prec_5 = (x_1, \ldots, x_{10}, x_\perp)$
- θ_5 is shown in the table below were an entry $u_i \times x_j = \theta_5(x_j, x_i)$

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
u_1	0.93	0.99	0.85	0.87	0.91	0.84	0.95	0.86	0.97	0.99
u_2	0.84	0.89	0.82	0.87	0.81	0.97	0.99	0.85	0.98	0.81
u_3	0.97	0.92	0.84	0.95	0.96	0.84	0.86	0.96	0.83	0.83
u_4	0.93	0.97	0.82	0.99	0.99	0.98	0.92	0.93	0.99	0.80
u_5	0.97	0.88	0.94	0.83	0.93	0.84	0.93	0.87	0.82	0.88

Appendix C

CISO Consultations

C.1 Consultation Questions

$CISO_1$

1. Who, if anyone, both within and outside of the organisation do you consult when reviewing policies? (e.g., internal: your own security officers, legal representatives, internal auditors, human resources personnel, etc.) (e.g., external: auditors, government representatives, police, etc.)

2. What sources of information do you interact with when reviewing policies? (e.g., standards such as ISO or COBIT, management forums, journals, etc.?)

3. What do you do with information obtained from other sources? (e.g., is it 'recorded' in policy? Stored as-is within the organisation? Integrated into a knowledge base or standardised knowledge repository for future reference?)

4. Are your policies divided into any particular categories (e.g. password policy, building access policy)?

5. Would you consider using information security knowledge supplied by other individuals (i.e. security managers / security officers)? Would you trust it? What if content had been supplied anonymously?

6. Would you consider using content that had been agreed on by the majority of community users but which you didn't agree with?

7. Specific to security controls that interact with people in your organisation, how do you go about defining methods for evaluating the correctness and effectiveness of your security controls (where those controls are enacted as part of your security policies)? Do you use well-known procedures and performance metrics that can also be seen in other organisations, or do you have scope to define your own methods and metrics for evaluating security mechanisms? (E.g. spot checks to measure number of infractions of a clear-desk policy, audit logs, instrumentation data, etc.)

8. How do you justify proposed security policy changes to senior management? In what terms must controls be described in order to adequately convey to senior management that any given security change is necessary and appropriate?

9. How is it determined that guidelines satisfy particular legal requirements, and that in turn specific controls address the legal requirements associated with specific guidelines?

10. Do you implement any verified 'best practices' in information security management? If so, how do these differ from 'accepted standards'? I.e. do they offer a more evidence-based approach over a 'common sense' approach?

11. When translating external standards/guidelines into internal policies, what is the typical process that is followed? (E.g. simply extracting and directly copying relevant advice, down to low-level implementation configuration details?)

12. You have stated that organisations typically converge upon similar security solutions. How are qualities of security solutions conveyed between organisations, i.e. without openly divulging security practices to one another or other parties?

13. Where security controls affect individuals within your organisation, do you find that there any often many different methods for enforcing those controls? If so, what is the process for evaluating the best approach to use? (e.g. enforcing controls in hardware to guarantee compliance but aggravate staff vs. e-mailing staff and asking them kindly to do as they are asked with the potentially unsatisfied hope that they will comply)

14. How regularly are security policies reviewed and changed in your organisation?

CISO_2

1. Which other professions, both within and outside of the organisation, do you consult with when reviewing policies, and for what purposes do you consult with them? (e.g.,

internal: your own security officers, legal representatives, internal auditors, human resources personnel, etc.) (e.g., external: auditors, government representatives, etc.)

2. Which sources of information do you interact with when reviewing policies? (e.g., standards such as ISO or COBIT, management forums, journals, etc.?)

3. What do you do with information obtained from other sources? (e.g., is it 'recorded' in policy? Stored as-is within the organisation? Integrated into a knowledge base or standardised knowledge repository for future reference?) .

4. Are your policies divided into any particular categories? (e.g. password policy, building access policy)

5. How regularly are security policies reviewed and changed in your organisation?

6. Do you ever share information security management knowledge with individuals in similar positions in other organisations? If so, how are qualities of security solutions conveyed between organisations, i.e. without openly divulging security practices to one another or other parties? (e.g. sharing information on how to configure password authentication policies)

7. Would you consider using information security knowledge supplied by other individuals in similar positions to you in other organisations (i.e. security managers / security officers) if it was supplied anonymously to a trusted/credible knowledge repository of some description? If so, what guarantees would need to be made about that knowledge for you to trust it?

8. Would you consider using management advice that had been agreed on by the majority of the information security management community but which you didn't agree with? If not, why not?

9. Where you have security controls that directly affect people in your organisation, how do you go about evaluating the correctness and effectiveness of those controls?

10. Do you use any well-known procedures and performance metrics that can also be seen in other organisations, or do you have your own methods and metrics for evaluating human-facing security mechanisms? (e.g. spot checks to measure number of infractions of a clear-desk policy, audit logs, instrumentation data, etc.)

11. How do you justify proposed security policy changes to senior management?

12. In what terms must controls be described in order to adequately convey to senior management that any given security policy change is necessary and appropriate?

13. Does the approach change when discussing human-facing security directives?

14. How is it determined that specific controls address the requirements of specific guidelines or standards of information security management?

15. Do you find that there are factors external to your decisions concerning security which must be considered when assessing the range of solutions you can implement within your organisation? If so, what are these factors? (e.g. financial limitations, expectations of industry/government i.e. 'common practices')

16. When translating external standards/guidelines into internal policies, what is the typical process that is followed? (e.g. extracting and directly copying relevant advice, down to implementation details?)

17. Where security controls affect individuals within your organisation, do you find that there any often many different methods for enforcing those controls? If so, what is the process for evaluating the best approach to use?

C.2 Tool Evaluation Session Structure

Demonstration of tool (5∼10 minutes)

- Brief high level overview of tool layout (what is where)
- How and what content is displayed
- User-specific features - login to edit/add notes
- Editing content
- Add individual
- Add new/existing properties
- Edit individual
- Note keeping

Exploratory evaluation by participant (5∼10 minutes)

- Allow participant to add knowledge relating to human factors in information security
- Prompt the participant where appropriate to utilise some of the interface features

Q&A session (15∼20 minutes)

1. What kinds of knowledge could be recorded with this tool?
2. Are there any features of the interface that are confusing?
3. Does the knowledge structure seem clear to you?
4. Are there elements within or missing from the knowledge structure that are especially useful for capturing your knowledge and which should be given focus within the interface?
5. Are there any features of the interface or knowledge structure that you think would benefit from more or less detail?
6. Were you restricted in any way from recording knowledge?
7. Would the collaboration features (notes, annotations, etc.) be useful in discussing knowledge with peers or other stakeholders in the organisation?
8. Would you benefit from additional mechanisms for reaching consensus (e.g. voting/rating system)?
9. How could the tool give more encouragement for you to supply content?
10. Are there any elements of the interface which raise security/privacy concerns for your organisation?
11. Are there aspects of the interface that you think might confuse some users?
12. Does available help system give enough guidance in constructing knowledge base?
13. How do you think such a tool might be used in your organisation or other organisations?
14. How would your organisation benefit from using this tool?
15. Are there any drawbacks you can see to introducing the tool into your organisation?
16. Would you benefit from extensions to the content such as costs, methods to measuring policy success, methods of enforcing controls (i.e., behavioural controls), previous success history and how to implement?

Appendix D

OWL Ontology Encoding

D.1 Security Ontology

Below is the content of an example information security ontology automatically encoded in the Web Ontology Language (OWL) by the graphical ontology development tool described in Chapter 7.

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY example "http://www.semanticweb.org/ontologies/2010/3/example.owl#" >
]>

<rdf:RDF xmlns="http://www.semanticweb.org/ontologies/2010/3/example.owl#"
     xml:base="http://www.semanticweb.org/ontologies/2010/3/example.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:example="http://www.semanticweb.org/ontologies/2010/3/example.owl#">
    <owl:Ontology rdf:about=""/>
```

```
<!--    ////////////////////
//
// Object Properties
//
//////////////////////////////
 -->

<!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#exploitedBy -->

<owl:ObjectProperty rdf:about="#exploitedBy">
    <rdfs:range rdf:resource="#ProceduralThreat"/>
    <rdfs:domain rdf:resource="#Vulnerability"/>
</owl:ObjectProperty>

<!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#
followsRiskApproach -->

<owl:ObjectProperty rdf:about="#followsRiskApproach">
    <rdfs:domain>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#followsRiskApproach"/>
            <owl:onClass rdf:resource="#BehaviourControl"/>
            <owl: qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1
            </owl: qualifiedCardinality >
        </owl:Restriction>
    </rdfs:domain>
    <rdfs:range>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#followsRiskApproach"/>
            <owl:onClass rdf:resource="#RiskControlType"/>
            <owl: qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1
            </owl: qualifiedCardinality >
        </owl:Restriction>
    </rdfs:range>
</owl:ObjectProperty>
```

```
<!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#hasFoundation -->

<owl:ObjectProperty rdf:about="#hasFoundation">
    <rdfs:domain>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasFoundation"/>
            <owl:onClass rdf:resource="#ProceduralThreat"/>
            <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1
            </owl:qualifiedCardinality>
        </owl:Restriction>
    </rdfs:domain>
    <rdfs:range>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasFoundation"/>
            <owl:onClass rdf:resource="#BehaviouralFoundation"/>
            <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1
            </owl:qualifiedCardinality>
        </owl:Restriction>
    </rdfs:range>
</owl:ObjectProperty>

<!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#hasVulnerability -->

<owl:ObjectProperty rdf:about="#hasVulnerability">
    <rdfs:domain rdf:resource="#Asset"/>
    <rdfs:domain rdf:resource="#BehaviourControl"/>
    <rdfs:range rdf:resource="#Vulnerability"/>
</owl:ObjectProperty>

<!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#managesRiskOf -->

<owl:ObjectProperty rdf:about="#managesRiskOf">
    <rdfs:domain rdf:resource="#BehaviourControl"/>
    <rdfs:range rdf:resource="#ProceduralThreat"/>
</owl:ObjectProperty>

<!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#mitigatedBy -->

<owl:ObjectProperty rdf:about="#mitigatedBy">
    <rdfs:range rdf:resource="#BehaviourControl"/>
    <rdfs:domain rdf:resource="#Vulnerability"/>
</owl:ObjectProperty>
```

```
<!-- ////////////////////
//
// Data properties
//
//////////////////////////
 -->

<!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#
threatConsequence -->

<owl:DatatypeProperty rdf:about="#threatConsequence">
    <rdfs:domain rdf:resource="#ProceduralThreat"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<!-- ////////////////
//
// Classes
//
//////////////////////////
 -->


<!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#Asset -->

<owl:Class rdf:about="#Asset">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>

<!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#BehaviourControl -->

<owl:Class rdf:about="#BehaviourControl">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>

<!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#
BehaviouralFoundation -->

<owl:Class rdf:about="#BehaviouralFoundation">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>
```

```
<!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#ProceduralThreat -->

<owl:Class rdf:about="#ProceduralThreat">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>

<!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#RiskControlType -->

<owl:Class rdf:about="#RiskControlType">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>

<!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#Vulnerability -->

<owl:Class rdf:about="#Vulnerability">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>

<!-- http://www.w3.org/2002/07/owl#Thing -->

<owl:Class rdf:about="&owl;Thing"/>

<!--    ///////////
//
//  Individuals
//
//////////////////////
 -->

<!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#capability -->

<BehaviouralFoundation rdf:about="#capability">
    <rdf:type rdf:resource="&owl;Thing"/>
</BehaviouralFoundation>
```

```
    <!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#
    make_password_easier_to_remember -->

    <BehaviourControl rdf:about="#make_password_easier_to_remember">
        <rdf:type rdf:resource="&owl;Thing"/>
        <followsRiskApproach rdf:resource="#reduction"/>
        <managesRiskOf rdf:resource="#single_password_forgotten"/>
    </BehaviourControl>

    <!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#password -->

    <Asset rdf:about="#password">
        <rdf:type rdf:resource="&owl;Thing"/>
        <hasVulnerability rdf:resource="#single_password_memorisation_difficult"/>
    </Asset>

    <!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#reduction -->

    <owl:Thing rdf:about="#reduction">
        <rdf:type rdf:resource="#RiskControlType"/>
    </owl:Thing>

    <!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#
    single_password_forgotten -->

    <ProceduralThreat rdf:about="#single_password_forgotten">
        <rdf:type rdf:resource="&owl;Thing"/>
        <threatConsequence
            >user temporarily without access</threatConsequence>
        <hasFoundation rdf:resource="#capability"/>
    </ProceduralThreat>

    <!-- http://www.semanticweb.org/ontologies/2010/3/example.owl#
    single_password_memorisation_difficult -->

    <Vulnerability rdf:about="#single_password_memorisation_difficult">
        <rdf:type rdf:resource="&owl;Thing"/>
        <mitigatedBy rdf:resource="#make_password_easier_to_remember"/>
        <exploitedBy rdf:resource="#single_password_forgotten"/>
    </Vulnerability>
</rdf:RDF>
```