Resiliency Variance in Workflows with Choice

John C. Mace(✉), Charles Morisset, and Aad van Moorsel

School of Computing Science, Newcastle University,
Newcastle upon Tyne, NE1 7RU, UK
{john.mace,charles.morisset,aad.vanmoorsel}@ncl.ac.uk

Abstract. Computing a user-task assignment for a workflow coming with probabilistic user availability provides a measure of completion rate or *resiliency*. To a workflow designer this indicates a risk of failure, especially useful for workflows which cannot be changed due to rigid security constraints. Furthermore, resiliency can help outline a *mitigation strategy* which states actions that can be performed to avoid workflow failures. A workflow with choice may have many different resiliency values, one for each of its execution paths. This makes understanding failure risk and mitigation requirements much more complex. We introduce *resiliency variance*, a new analysis metric for workflows which indicates volatility from the resiliency average. We suggest this metric can help determine the risk taken on by implementing a given workflow with choice. For instance, high average resiliency and low variance would suggest a low risk of workflow failure.

Keywords: Workflow satisfiability problem · Quantitative analysis · Resiliency metrics

1 Introduction

Many business domains including finance, healthcare and eScience use the concept of workflow to efficiently orchestrate their everyday business processes [5, 14, 15]. Although definitions may vary, workflows typically consist of tasks (*the work*) and ordering conditions (*the flow*) [1]. Completing every execution, or instance of a workflow means assigning each task to a user in accordance with required security constraints. Often these are enforced by regulations ensuring only users with correct capabilities are matched with appropriate tasks whilst limiting data access and reducing the threat of collusion and fraud [7, 17].

Finding a user assignment for every task such that all security constraints are met is a well studied problem, known as the *workflow satisfiability problem* (WSP) [9, 29]. The WSP has been shown to be NP-hard, meaning every combination of users to tasks may have to be tried before finding an assignment that satisfies a workflow. The WSP assumes all users will be available during execution, however periodic user unavailability at runtime means a satisfiable workflow at design time may become unsatisfiable during its operation.

A. Fantechi and P. Patrizio (Eds.): SERENE 2015, LNCS 9274, pp. 128–143, 2015.
DOI: 10.1007/978-3-319-23129-7_10

In cases where no valid user is available for a specific task assignment, the security constraints inadvertently block a workflow from completing. Any available users are either not permitted to perform the task, or are permitted but cannot do so due to constraints with previously executed tasks. Without violating the security policy, the alternative is to terminate early thus causing a workflow *failure*. Forcing early termination of a workflow may bring heavy operational penalties in terms of monetary costs, lost productivity and reduced reputation. In practice, blocked workflows are typically managed by performing mitigating actions which facilitate a completable workflow, often essential in healthcare and other critical domains where failure tolerance is small. For example, it may be that authorising a security override (e.g. break glass [24]) has less long-term impact than allowing the workflow to fail. Elucidating permitted mitigation actions to be taken if a workflow becomes blocked forms a workflow *mitigation strategy*.

When designing workflows it is favourable to predict the risk of workflow failure and understand requirements, in terms of actions, impact and cost of a suitable mitigation strategy. This is especially important for workflows coming with rigid security constraints that cannot be changed at design time. One method is to consider the *workflow resiliency problem*, an extension of the WSP that looks to find an assignment to satisfy a workflow even when some users become unavailable [29]. The quantitative approach to this problem taken in [20] allows a workflow's resiliency to be expressed as a measure of expected completion rate. This value in turn indicates the risk of workflow failure, and therefore the likely need to perform mitigation actions.

In [20], the authors consider analysing the resiliency of workflows with only sequential and parallel control patterns such that each has a single execution path. Computing the resiliency for workflows of this form provides a singular comprehensible indicator of failure risk. Low failure risk (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 a security constraint emergency override. High failure risk (low resiliency) would suggest a broader strategy including more permanent yet costly mitigation actions such as staff training and repealing user unavailability.

This paper considers workflows with gateways, or choice co-ordinators such that multiple execution paths exist that can be taken at runtime to complete a workflow, and where each path may come with a different resiliency value. Understanding risk failure and mitigation strategy requirements of such workflows can be much more complex, especially when a workflow contains hundreds if not thousands of execution paths. Taking the resiliency average, or *expected resiliency* alone may be a misleading indicator of failure risk, especially when a workflow contains paths of both very high and very low resiliency.

We introduce *resiliency variance*, a new metric for workflow failure risk analysis 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 [11]. Similarly, resiliency variance could provide a workflow designer with an indicator of failure risk taken on by implementing a given workflow with choice. This could also be useful for predicting a suitable mitigation strategy. For example, a workflow with high expected resiliency and low variance indicates low failure risk and mitigation cost whilst high variance would suggest a much higher failure risk and mitigation cost.

We give an overview of workflow resiliency related work in Sect. 2 whilst Sect. 3 defines a workflow with choice. Section 4 discusses workflow resiliency and its calculation before introducing resiliency variance and show how it is calculated using a real-world university based purchase request workflow. Section 5 provides a discussion on workflow mitigation techniques and how resiliency variance could inform mitigation strategy choice. Concluding remarks are given in Sect. 6.

2 Related Work

A number of previous studies on workflow resiliency and its enhancement appear in the literature. Wang et al. took a first step in [29] to quantify resiliency and declare a workflow as k resilient if it can withstand up to k absent users in all instances. In [20] Mace et al. consider workflows that are not always k resilient and provide a measure of quantitative resiliency indicating how much a workflow is likely to terminate for a given security policy and user unavailability model. This approach illustrates a trade-off exists between aspects such as success rate, expected termination point and computation time.

Basin et al. in [4] overcome 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, administration and maintenance. This is feasible in certain business domains but may have limited application in workflows where roles are more specialised; for example is adding an untrained user to the role doctor to satisfy a security policy better than overriding it and enabling a constrained but qualified doctor?

Wainer et al. consider in [28] the explicit overriding of security constraints in workflows, by defining a notion of privilege. In [8] Brunel et al. suggest a security policy may still be satisfied even though some security constraints may be violated. This is considered acceptable by defining additional conditions that apply in the case of violation that must be satisfied to comply with the security policy. Bakkali [2] suggests enhancing resiliency through delegation and the placement of criticality values over workflows. Delegates are chosen on their suitability but may lack competence; this is considered the ‘price to pay’ for resiliency. As delegation takes place at a task level it is not currently clear whether a workflow can still complete while meeting security constraints. In [10] Crampton et al. suggest a mechanism that can automatically respond to the absence of users by delegating a task appropriately when no qualified user is available to perform it.

Current literature does not fully address the issue of workflows that must operate but may not be resilient in every instance. Although many approaches have been suggested in isolation, a range of different remediation options including policy overrides are necessary for a more optimal solution.

3 Workflow

In general, a workflow consists of a set of tasks which can be executed following some constraints: some tasks must be executed before others, some tasks can be executed in parallel, some tasks can be executed instead of others. There exist several definitions for workflows in the literature, for instance as a partial ordering of tasks [9] or as a directed graph [27]. The aim of the work presented here is to study the resiliency of workflows with choice, using the notion of resiliency introduced in [20], where a workflow is defined as a set of users, a partially ordered set of tasks and a security policy.

However, this definition does not allow for *choice*, i.e., for having different paths in a workflow according to the evaluation of some choices. For instance, a workflow managing the purchasing process might have different tasks based on the cost of the purchase. In this section, we first give an inductive definition for a workflow with choice inspired from [16], and we show how it can be reduced into a definition compatible with [20].

3.1 Task Structure with Choice

A *task structure* is built upon two sets: a set T of atomic tasks and a set C of atomic choices. Intuitively, the former set represents each action that can be performed, while the latter represents the different points where the 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;
- 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.

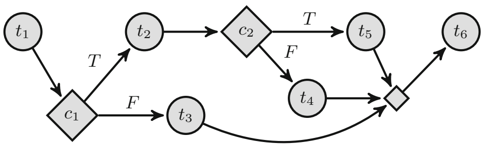

Fig. 1. Running example task structure

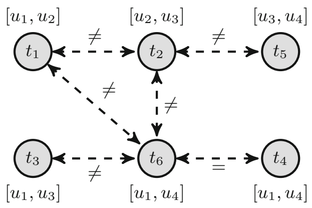

Fig. 2. Running example security policy

Running example. *As a running example to illustrate the different concepts presented here, we define $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$, $C = \{c_1, c_2\}$ and*

$$ts_1 = t_1 \rightarrow [c_1 : [t_2 \rightarrow [c_2 : t_5 \,?\, t_4]] \,?\, t_3] \rightarrow t_6$$

Note that for the sake of simplicity, we do not consider in the running example any parallel composition. We give a graphical representation of ts_1 in Fig. 1 where tasks are represented as circles and choices as diamonds. In order to represent the end of a choice, we use the empty diamond symbol, 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.

It is worth pointing out that in the graphical notation used in Fig. 1, the choice nodes correspond to or-nodes and the empty diamond to a merge coordinator in [27].

3.2 Task Structure Reduction

At runtime, the choices in a task structure are resolved, and only the corresponding paths are executed. We adopt here an approach where we do not know how each choice is going to be resolved at runtime, and we therefore consider beforehand all possible solutions. Intuitively, we want to *reduce* a task structure with choice to one without choice, for which all tasks should be executed.

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: $TSC \times \wp(C) \rightarrow TSC$, 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:

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

All possible instances without choice of a task structure with choice can be defined by:

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

A task structure without choice can be converted to a set of tasks with a partial ordering, thus allowing us to reuse existing corresponding techniques. 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 $ord : TS \rightarrow \wp(T \times T)$, which, given a task structure without choice

ts, returns the ordering relation over the tasks in ts.

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

Running example. *The possible instances of ts_1 are:*

- $t_1 \rightarrow t_2 \rightarrow t_5 \rightarrow t_6$ *(corresponding to $\gamma = \{c_1, c_2\}$);*
- $t_1 \rightarrow t_2 \rightarrow t_4 \rightarrow t_6$ *(corresponding to $\gamma = \{c_1\}$);*
- $t_1 \rightarrow t_3 \rightarrow t_6$ *(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.

3.3 Security Policy

Next we define a set of users U that comes with a security policy over the set of tasks T. In general, a security policy is a triple $p = (P, S, B)$ where:

- $P \subseteq U \times T$ are *user-task permissions*, such that $(u, t) \in P$ if, and only if u is allowed to perform t.
- $S \subseteq T \times T$ are *separations of duty*, such that $(t, t') \in S$ if, and only if the users assigned to t and t' are distinct.
- $B \subseteq T \times T$ are *bindings of duty*, such that $(t, t') \in B$ if, and only if the same user is assigned to t and t'.

A workflow therefore consists of a set of tasks, with an ordering relation over the tasks, a set of users, and a security policy.

Definition 1. *A workflow is a tuple $w = (ts, U, p)$, where ts is a task structure, U is a set of users, and p is a security policy.*

Note we assume ts to be equivalent to an inducement of the task manager τ given an initial task $t_0 \in T$ from the definition of workflow given in [20]. Given a workflow $w = (ts, U, p)$ and a set of choices $\gamma \subseteq C$, we abuse the notation and write $red(w, \gamma)$ for the workflow $w' = (red(ts, \gamma), U', p')$, where p' corresponds to p restricted to tasks appearing in $red(ts, \gamma)$ and U' corresponds to U restricted to users appearing in p'. Similarly, we write $ins(w)$ for the set of workflows w' such that there exists $\gamma \subseteq C$ satisfying $w' = red(w, \gamma)$.

Running example. *We now consider a set of users $U_1 = \{u_1, u_2, u_3, u_4\}$ and a security policy $p_1 = (P_1, S_1, B_1)$ that states:*

- $P_1 = \{(u_1, t_1), (u_2, t_1), (u_2, t_2), (u_3, t_2), (u_1, t_3),$
  $(u_3, t_3), (u_1, t_4), (u_4, t_4), (u_3, t_5), (u_4, t_5), (u_1, t_6),$
  $(u_4, t_6)\}$

– $S_1 = \{(t_1, t_2), (t_1, t_6), (t_2, t_5), (t_2, t_6), (t_3, t_6\}$
– $B_1 = \{(t_4, t_6)\}$.

Figure 2 illustrates p_1, where the dotted arrows labelled '$\neq$' and $=$ signify the constraints given in S_1 and B_2 respectively. A label $[u_m, \ldots, u_n]$ states the users that are authorised by P_1 to execute t_i.

4 Workflow Resiliency

Given a workflow $w = (ts, U, p)$, we need to assign tasks in ts to users in U in order to execute them, while respecting the policy p. If ts contains some choice elements, it is not strictly necessary to assign 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 of task assignment for a workflow with choice to considering the task assignment of all possible instances without choice, thanks to the function *red*. In this section, we first describe the *resiliency problem* for workflows without choice, following the existing literature, and we then lift the problem to workflows with choice.

4.1 Resiliency Without Choice

Given a workflow without choice, finding a complete assignment that satisfies all the security constraints is known as the Workflow Satisfiability Problem (WSP), and we refer for instance to [9,29] for further reading on this problem.

Solving the WSP assumes any $u \in U$ will always be available for every instance of a workflow. However in practice, sickness, vacation, heavy workloads, etc., can cause users to periodically be unavailable for task assignments. It is then important to find a valid and complete assignment that maximises the chance of a workflow w to finish: finding an assignment such that w will likely finish 9 out of 10 cases is clearly better than choosing one where w will likely finished only 1 out of 10 cases. This is called the *resiliency problem*, whether a workflow w can be satisfied even when some users become absent.

User unavailability in workflows was introduced by Wang and Li [29], who considered a somewhat *binary* approach where users are either available or not. A workflow is classified as k resilient if the workflow can still be satisfied regardless of which k users become absent. In [20], Mace et al. introduced probabilistic user availability and showed that computing the optimal policy of a Markov Decision Process (MDP [6]) is equivalent to finding an assignment that maximises a value function returning the probability of the workflow w to finish. We refer to [20] for the detail of this approach, and given a workflow without choice w, we write $res(w) \in [0, 1]$ for its resiliency. Our main contribution in this paper consists in adapting this measure to workflows with choice.

It is worth pointing out that understanding when users will and will not be available is an obvious requirement when calculating resiliency, which may

Table 1. Running example probabilistic user availability models

	AM_1				AM_2			
	u_1	u_2	u_3	u_4	u_1	u_2	u_3	u_4
t_1	0.95	0.90	0.96	0.94	0.95	0.90	0.96	0.94
t_2	0.88	1.00	0.90	0.97	0.88	1.00	0.90	0.97
t_3	0.85	0.77	0.99	0.89	0.85	0.77	0.85	0.89
t_4	0.40	0.88	0.89	0.52	0.78	0.88	0.89	0.80
t_5	0.93	0.87	0.96	0.96	0.93	0.87	0.82	0.82
t_6	0.98	0.94	0.98	0.98	0.98	0.94	0.98	0.98

Table 2. Running example resiliency measures

	AM_1	AM_2
res(w_{11})	0.89	0.76
res(w_{22})	0.48	0.74
res(w_{33})	0.92	0.79
$expR(w_1)$	0.76	0.76
$varR(w_1)$	0.0403	0.0004

be deduced from a mixture of operational logs, behavioural analysis and user submissions (known and tentative absences). A key influential aspect is *how* the unavailability of users is modelled in the corresponding MDP [21]. In this paper we consider a dynamic user availability model meaning any user who becomes unavailable for a task may become available again at any step later in the workflow.

A key influential aspect is *how* the unavailability of users is modelled in the corresponding MDP [21]. We assume a dynamic availability model for the rest of this paper.

4.2 Resiliency with Choice

We now consider adapting the resiliency measure for a workflow without choice to a resiliency measure for a workflow with choice using the aid of our running example.

Running example. *We consider a workflow* $w_1 = (ts_1, U_1, p_1)$ *such that* $ins(w_1) = \{w_{11}, w_{22}, w_{33}\}$ *where:*

- $w_{11} = (t_1 \rightarrow t_2 \rightarrow t_5 \rightarrow t_6, U_{11}, p_{11})$
- $w_{22} = (t_1 \rightarrow t_2 \rightarrow t_4 \rightarrow t_6, U_{22}, p_{22})$
- $w_{33} = (t_1 \rightarrow t_3 \rightarrow t_6, U_{33}, p_{33})$.

We consider two different probabilistic user availability models AM_1 *and* AM_2*, given in Table 1 and assume that* AM_2 *is the result of escalation type mitigation actions carried out on* AM_1*, for instance by cancelling user vacations (see Sect. 5.1). An entry* $t_i \times u_i$ *is the probability of user* u_i *being available for the assignment of task* t_i*. The resiliency of each* $w_{ii} \in ins(w_1)$ *is given in Table 2.*

Resiliency Extrema. Finding the minimal resiliency for a workflow with choice w indicates which $w' \in ins(w)$ will give the lowest success rate for w if executed. This can be interpreted as the worst case, or the instance in w with the highest failure risk. On first glance this indicates which parts of w need the most attention in terms of mitigation. For instance, in our running example, w_1 is most likely to fail when w_{22} is executed which gives the minimal resiliency, 0.48 and 0.74 under AM_1 and AM_2 respectively. Imagine now that under AM_1, w_{22} has a low probability of execution, e.g., 0.01, or 1 execution in 100 cases whereas w_{33} with 0.92 resiliency has a high execution probability, e.g., 0.80, or 80 in 100 cases. In general, the resiliency for w_1 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 w by calculating both the maximal and minimal resiliency for w. In our running example under AM_1, w_1 has a large bound with an expected finish rate of between 0.48 (w_{22}) and 0.92 (w_{33}). Under the mitigated AM_2, w_1 has a much smaller bound such that the expected finish rate is between 0.74 (w_{22}) and 0.79 (w_{33}). The resiliency bound can be a useful resiliency measure when all $w' \in ins(w)$ have an equiprobable chance of being executed. If however under AM_1, w_{33} has a low execution probability of 0.01 whilst w_{22} has a high execution probability of 0.8 then in general the resiliency achieved will tend towards the minimal value of 0.48. Placing a bounds on the resiliency in this case becomes a misleading measure of resiliency to the workflow designer.

Resiliency Distribution. Given a workflow with choice w, calculating the resiliency for every possible instance $w' \in ins(w)$ provides the full resiliency distribution for w. This can enable the workflow designer to identify instances of low resiliency, and therefore those needing more extensive mitigation. A tolerance threshold for resiliency may exist for w, deemed acceptable when every instance w' has a resiliency equal to or more than the threshold, in other words the probability that every w' meets the threshold is 1.

In our running example we assume a resiliency threshold of 0.50 and for simplicity, an equiprobable execution model for all $w' \in ins(w_1)$ where the execution probability of w' is 0.33. A more complex probabilistic model could easily be imagined, and we leave such cases for future works. Under AM_1 the probability of w_1 meeting this threshold is therefore 0.66 (*unacceptable*), whilst under the mitigated AM_2 the probability is now 1 (*acceptable*).

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 contain hundreds if not thousands of execution paths. It may be more useful for a workflow designer to provide a singular, easy to understand measure of resiliency for a workflow with choice.

Expected Resiliency. We now assume a probability function $prob : W \to [0, 1]$, which given a workflow without choice $w' \in ins(w)$, returns the probability of w' being executed. The expected resiliency indicates the likely success rate across every instance in a workflow with choice w, calculated as the average resiliency of all $w' \in ins(w)$. We define the function $expR : W \to [0, 1]$, which given a workflow with choice w returns the expected resiliency of w.

$$expR(w) = \sum_{w' \in ins(w)} prob(w').res(w')$$

In our running example, assuming $prob(w') = 0.33$ for all $w' \in ins(w_1)$, the expected resiliency is 0.76 for w_1 under both AM_1 and AM_2, shown in Table 2.

This in turn indicates an expected failure rate for w_1 of 0.24. Under AM_1 with an equiprobable execution model means the expected resiliency of 0.76 is not assured with every execution of w_1. Each time the instance w_{22} is executed, the probability of w_1 terminating successfully is only 0.48. This means roughly half of these instance executions will cause w_1 to fail. When executing w_{11} and w_{33} the actual 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 the mitigated model AM_2, the expected resiliency is now roughly attained whichever $w' \in ins(w_1)$ is executed. In this case the expected resiliency measure alone is arguably enough to indicate the true failure risk of w_1. In other words, a resiliency of ≈ 0.76 can be expected with every execution of w_1. This remains so even when the probabilistic execution model for all $w' \in ins(w_1)$ is not equally weighted. Note that to achieve this the resiliency of w_{11} and w_{33} under AM_1 has been reduced to 0.76 and 0.79 respectively.

Resiliency Variance. The resiliency variance is a measure of how spread out a distribution is, or the variability from the expected resiliency of all instances in a workflow with choice w. A resiliency variance value of zero indicates that the resiliency of all $w' \in ins(w)$ 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. As discussed in the Introduction, 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 $varW : W \rightarrow \mathbb{R}$, which given a workflow with choice w returns the resiliency variance of w.

$$varR(w) = \sum_{w' \in ins(w)} prob(w').(res(w') - expR(w))^2$$

The resiliency variance for our running example w_1, calculated under availability models AM_1 and AM_2 is given in Table 2.

An equiprobable execution model is again used for simplicity. Under AM_1 a resiliency variance of 0.0403 is calculated, equivalent to a large standard deviation of 0.20 ($\sqrt{varR(w_1)}$). Under the mitigated AM_2 the resiliency variance has been reduced to 0.0004, equivalent to a much smaller standard deviation of 0.02

Table 3. Resiliency measures for purchase request workflow

	AM_3	AM_4
minR(w_2)	0.48	0.55
maxR(w_2)	0.96	0.83
expR(w_2)	0.67	0.67
varR(w_2)	0.0183	0.0052

from the expected resiliency. Here we have a decrease by a factor of 10. Clearly this indicates in this case that all instances of w_1 under AM_2 have a probability of terminating successfully close to the expected resiliency of 0.76.

The former case (AM_1) indicates that instances in w_1 can have a large spread in terms of resiliency despite having the same expected resiliency as the latter case (AM_2) coming with a small spread, or variance. Under AM_1, the results show that instances exist in w_1 with much lower and higher probabilities of terminating successfully than the expected resiliency for w_1. The workflow w_1 can be considered *volatile* or *high risk* as it has a high risk of failing if one such instance with low resiliency is executed. Coupled with expected resiliency, resiliency variance can provide an easy to understand measure of workflow risk failure and allow workflow designers to quickly compare similar complex workflows (e.g., pre and post mitigation) to help them predict a suitable mitigation strategy.

4.3 Purchase Request Workflow

In this section we calculate resiliency measures including resiliency variance for a purchase request workflow w_2 that forms part of a real-life procurement procedure used by a large Australian-based university[1]. The workflow consists of 18 atomic tasks and 5 atomic choices, 4 users, and a security policy with 9 separation of duty constraints. The workflow task structure consists of 18 instances, i.e., 18 possible execution paths.

To calculate the resiliency measures we encode w_2 within the probabilistic model checking tool PRISM, which enables the specification, construction and analysis of probabilistic models such as MDPs [19]. PRISM is an intuitive choice as it can model both probabilistic and non-deterministic choice, and gives an efficient way to solve an MDP. For a systematic encoding of a workflow in PRISM and the mechanisms to compute the resiliency measure we refer the reader to the following technical report [22].

Resiliency measures for w_2 are given in Table 3 under two probabilistic availability models AM_3 and AM_4. We assume AM_4 results from mitigation carried out on AM_3. Measures calculated are minimal and maximal resiliency represented as $minR(w_2)$ and $maxR(w_2)$ respectively, expected resiliency and

[1] http://www.fin.unsw.edu.au/files/PP/Purchase_Order_Procedure.pdf.

resiliency variance. The expected resiliency is the same for w_2 under both AM_3 and AM_4 yet the resiliency variance is reduced by a factor of 3.5 under the latter, indicating w_2 now has a lower risk of failure.

5 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 no valid user-task assignment exists. These mitigation actions are categorised into two classes, long-term actions and emergency actions.

5.1 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 [25,26]. 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 and/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. We assume the use of this action in our running example to mitigate user availability model AM_1, thereby creating AM_2.
- *training:* a user's capabilities are raised to an acceptable level before granting permission to perform a task.
- *change policy* [3,4] - security constraints are removed or changed (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.

5.2 Emergency Actions

Emergency actions can help raise the 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 blocked. 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* [2,13,18]*:* 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 user-task permissions but can result in lower standards and higher risk.
– *break glass* [23]*:* 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 [12] until another assignment path can be taken which bypasses the invalid user-task assignment.

5.3 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 dependent. We do however offer some discussion on this matter and show how the resiliency measures for a workflow with choice discussed in Sect. 4.2 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.

6 Conclusion

It is important that a workflow designer can predict the risk of failure before implementing a workflow, especially if its design must include rigid security constraints. In [20] the probability of finding a user assignment for all tasks in a workflow without choice provides a measure of completion rate or *resiliency*. We extend this approach by considering workflows with choice which may come with multiple resiliency values, one for each execution path. We consider computing different resiliency measures including *resiliency variance* which indicates volatility from the resiliency average. We suggest this metric can help predict the risk taken on when implementing a given workflow and help determine suitable mitigation actions which should be executed when no valid user assignment exists for a workflow task.

References

1. Workflow Management Coalition: The workflow reference model. In: Lawrence, P. (ed.) Workflow Handbook 1997, pp. 243–293. Wiley, New York (1997)
2. Bakkali, H.E.: Enhancing workflow systems resiliency by using delegation and priority concepts. J. Digit. Inf. Manag. **11**(4), 267–276 (2013)
3. Basin, D., Burri, S.J., Karjoth, G.: Obstruction-free authorization enforcement: aligning security with business objectives. In: Proceedings of the 2011 IEEE 24th Computer Security Foundations Symposium, CSF 2011, pp. 99–113. IEEE Computer Society, Washington, DC (2011)
4. Basin, D., Burri, S.J., Karjoth, G.: Optimal workflow-aware authorizations. In: Proceedings of the 17th ACM Symposium on Access Control Models and Technologies, SACMAT 2012, pp. 93–102. ACM, New York (2012)
5. Basu, A., Kumar, A.: Research commentary: workflow management issues in e-business. Inf. Syst. Res. **13**(1), 1–14 (2002)

6. Bellman, R.: A Markovian decision process. Indiana Univ. Math. J. **6**, 679–684 (1957)
7. Botha, R., Eloff, J.H.P.: Separation of duties for access control enforcement in workflow environments. IBM Syst. J. **40**(3), 666–682 (2001)
8. Brunel, J., Cuppens, F., Cuppens, N., Sans, T., Bodeveix, J.-P.: Security policy compliance with violation management. In: Proceedings of the 2007 ACM Workshop on Formal Methods in Security Engineering, FMSE 2007, pp. 31–40. ACM, New York (2007)
9. Crampton, J., Gutin, G., Yeo, A.: On the parameterized complexity and kernelization of the workflow satisfiability problem. ACM Trans. Inf. Syst. Secur. **16**(1), 4 (2013)
10. Crampton, J., Morisset, C.: An auto-delegation mechanism for access control systems. In: Cuellar, J., Lopez, J., Barthe, G., Pretschner, A. (eds.) STM 2010. LNCS, vol. 6710, pp. 1–16. Springer, Heidelberg (2011)
11. Damodaran, A.: Strategic Risk Taking: A Framework for Risk Management, 1st edn. Wharton School Publishing, Upper Saddle River (2007)
12. Eder, J., Liebhart, W.: Workflow recovery. In: Proceedings of the First IFCIS International Conference on Cooperative Information Systems, 1996, pp. 124–134, June 1996
13. Gaaloul, K., Schaad, A., Flegel, U., Charoy, F.: A secure task delegation model for workflows. In: Second International Conference on Emerging Security Information, Systems and Technologies, 2008, SECURWARE 2008, pp. 10–15, August 2008
14. Georgakopoulos, D., Hornick, M., Sheth, A.: An overview of workflow management: from process modeling to workflow automation infrastructure. Distrib. Parallel Databases **3**(2), 119–153 (1995)
15. Hiden, H., Woodman, S., Watson, P., Cala, J.: Developing cloud applications using the e-science central platform. Philos. Trans. R. Soc. A Math. Phys. Eng. Sci. **371**(1983), 20120085 (2013)
16. Kiepuszewski, B., ter Hofstede, A.H.M., Bussler, C.J.: On structured workflow modelling. In: Wangler, B., Bergman, L.D. (eds.) CAiSE 2000. LNCS, vol. 1789, pp. 431–445. Springer, Heidelberg (2000)
17. Kohler, M., Liesegang, C., Schaad, A.: Classification model for access control constraints. In: IEEE International Performance, Computing, and Communications Conference, 2007, IPCCC 2007, pp. 410–417, April 2007
18. Kumar, A., van der Aalst, W.M.P., Verbeek, E.M.W.: Dynamic work distribution in workflow management systems: how to balance quality and performance. J. Manage. Inf. Syst. **18**(3), 157–193 (2002)
19. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011)
20. Mace, J.C., Morisset, C., van Moorsel, A.: Quantitative workflow resiliency. In: Kutyłowski, M., Vaidya, J. (eds.) ICAIS 2014, Part I. LNCS, vol. 8712, pp. 344–361. Springer, Heidelberg (2014)
21. Mace, J., Morisset, C., van Moorsel, A.: Modelling user availability in workflow resiliency analysis. In: Proceedings of the Symposium and Bootcamp on the Science of Security, HotSoS. ACM (2015)
22. Mace, J.C., Morisset, C., van Moorsel, A.: Impact of policy design on workflow resiliency computation time. Technical Report CS-TR-1469, School of Computing Science, Newcastle University, UK, May 2015

23. Marinovic, S., Craven, R., Ma, J., Dulay, N.: Rumpole: a flexible break-glass access control model. In: Proceedings of the 16th ACM Symposium on Access Control Models and Technologies, SACMAT 2011, pp. 73–82. ACM, New York (2011)
24. Povey, D.: Optimistic security: a new access control paradigm. In: Proceedings of the 1999 Workshop on New Security Paradigms, NSPW 1999, pp. 40–45. ACM, New York (2000)
25. Reichert, M., Weber, B.: Enabling Flexibility in Process-aware Information Systems: Challenges, Methods, Technologies. Springer Science & Business Media, Heidelberg (2012)
26. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Workflow exception patterns. In: Martinez, F.H., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 288–302. Springer, Heidelberg (2006)
27. van der Aalst, W.M.P., Hirnschall, A., Verbeek, H.M.W.E.: An alternative way to analyze workflow graphs. In: Pidduck, A.B., Mylopoulos, J., Woo, C.C., Ozsu, M.T. (eds.) CAiSE 2002. LNCS, vol. 2348, pp. 535–552. Springer, Heidelberg (2002)
28. Wainer, J., Barthelmess, P., Kumar, A.: W-RBAC - a workflow security model incorporating controlled overriding of constraints. Int. J. Coop. Inf. Syst. **12**, 2003 (2003)
29. Wang, Q., Li, N.: Satisfiability and resiliency in workflow authorization systems. ACM Trans. Inf. Syst. Secur. **13**(4), 40:1–40:35 (2010)