

Model-Based Mitigation of Availability Risks

Emmanuele Zambon*, Damiano Bolzoni*, Sandro Etalle* and Marco Salvato†

*University of Twente

Email: {emmanuele.zambon, damiano.bolzoni, sandro.etalles} (at) utwente.nl

†KPMG Italia S.p.a.

Email: msalvato (at) kpmg.it

Abstract—The assessment and mitigation of risks related to the availability of the IT infrastructure is becoming increasingly important in modern organizations. Unfortunately, present standards for Risk Assessment and Mitigation show limitations when evaluating and mitigating availability risks. This is due to the fact that they do not fully consider the dependencies between the constituents of an IT infrastructure that are paramount in large enterprises. These dependencies make the technical problem of assessing availability issues very challenging. In this paper we define a method and a tool for carrying out a Risk Mitigation activity which allows us to assess the global impact of a set of risks and to choose the best set of countermeasures to cope with them. To this end, the presence of a tool is necessary, due to the high complexity of the assessment problem. Our approach can be integrated in present Risk Management methodologies (e.g. COBIT) to provide a more precise Risk Mitigation activity. We substantiate the viability of this approach by showing that most of the input required by the tool is available as part of a standard business continuity plan, and/or by performing a common tool-assisted Risk Management.

I. INTRODUCTION

Information Risk Management is the process of assessing the risks an organization's IT infrastructure is exposed to, and of developing strategies to manage them. The process of Risk Management is usually divided into two main steps: Risk Assessment (RA) and Risk Mitigation (RM). The former activity identifies potential harmful threats to the information systems, while the latter consists of developing and implementing a strategy to manage them. Nowadays, Risk Management is often a primary task in enterprise organizations and it is widely considered a key factor for improving an organization's IT performance. Moreover, recent legislations, such as the *Sarbanes-Oxley Act* (SOX) of 2002 or the international accord known as *Basel II* [5] (*International Convergence of Capital Measurement and Capital Standards*), explicitly requires this kind of activity to be conducted to ensure stakeholders that the organization is operating properly.

Among the three main security properties of information, Confidentiality, Integrity and Availability (CIA), the importance of *availability* has grown enormously: today, organizations strongly depend on the availability of their information systems; moreover, availability of IT-related services is continuously growing in importance for enterprise revenues (on-line

banking, reservations for e-tickets etc.). This fact is confirmed by the increasing importance that Service Level Agreements (SLAs) are gaining. Currently, SLAs are considered one of the fundamental ways to define and control the expected availability and quality of a given service and are widely used not only between different organizations but also between units of the same company.

That managing availability risks is particularly important is confirmed by the fact that most Risk Management methodologies specifically require, when the mitigation phase takes place, the implementation of a plan for Business Continuity (Business Continuity Plan, BCP). A BCP describes which are the countermeasures that have been chosen, the people involved and the response procedures they should take in case of a disruption event, to guarantee a timely recovery.

Contribution In this paper we focus on assessing and mitigating the risks related to the availability of the IT infrastructure. This is particularly challenging because of the (temporal) dependencies linking the various constituents of an IT infrastructure (machines, processes, assets, etc.) with each other. In complex information systems, a failure in a remote component may propagate across the infrastructure and eventually affect the availability of a good deal of the entire system. Failing to appropriately assess the consequences of such propagations will result in inaccurate RA and RMs.

We argue that current Risk Management methodologies (e.g. COBIT, ISO 17799 [18], ISO 13335 [17] or OCTAVE [26]) show limitations when evaluating and mitigating availability risks. This is due to the fact that they do not fully consider the consequences of the functional dependencies between the constituents of an IT infrastructure: the consideration of these dependencies is mostly left to the judgement of the assessor carrying out the RA phase (although this is not made explicit clearly). Thus, these methodologies can only be useful to identify and fix individual risks an organization is exposed to (see also Section II). On the other hand, these dependencies are considered in more specific assessment methods such as the Business Continuity Plans, like in the new standard BS25999 [16] (see Section II for a detailed overview). These methods, however, do not specify how to use this information for RM.

Summarizing, nowadays the process of assessing and mitigating availability related risks depends very much on the human expertise, making Risk Management more an art than

This work has been accomplished during the third author's stay at the University of Trento, Italy, supported by the Serenity project. The first author is supported by the research project PROSECCO. The second author is supported by the research program Sentinels (www.sentinels.nl).

a science.

Our thesis is that it is possible to carry out an accurate tool-based RM by using the data collected during RA and BCP activities. To substantiate this thesis, in this paper we present a framework and a tool for the assessment and mitigation of availability-related IT risks. The framework is based on the Time-Dependency (TD) model, an extension of the model of the IT infrastructure as it is done according to the BS25999 (which largely coincides with the data collected by the KARISMA tool developed at KPMG for RA, see Section V). This model allows us to determine how incidents will propagate across the organization, and therefore what is the actual *impact* of incidents. With this information, we can carry out an optimization study by comparing the true expected benefit determined by the different *countermeasures* that can be put in place to cope with the various risks.

As we will mention, the computational complexity of the problems posed by our method, make it impossible to carry out the underlying analysis by hand, and this is why the method we propose requires the presence of an appropriate tool. We have implemented the tool using UPPAAL CORA [21] and Prolog.

We consider our solution a concrete enhancement to RM methodologies, providing automatic support to better evaluate the IT relationships and dynamics.

II. PRESENT METHODOLOGIES FOR RISK MANAGEMENT

There exists a number of standards and methodologies for Risk Management, among which COBIT (Control Objectives for Information and related Technology) [11] and BS25999 [16] are of particular relevance to our work. COBIT is the *de facto* standard for information control and IT Risk Management, addressing IT governance and control practices. It provides a reference framework for managers, users and security auditors. COBIT is mostly based on the concept of *control* (be it technical or organizational) which is used to assess, monitor and verify the current state of a certain process (that may refer to procedures, human resources, etc.) involved in the information system. To implement COBIT, the organization must benchmark its own processes against the control objectives suggested by the framework, using the so-called *maturity models* (derived from the Software Engineering Institute's Capability Maturity Model [27]). Maturity models basically provide: (1) a measure expressing the present state of an organization, (2) an efficient way to decide which is the goal to achieve and, finally, (3) a tool to evaluate progress toward the goal. Maturity modelling enables gaps in capabilities to be identified and demonstrated to management. Key Goal Indicators (KGI) and Key Performance Indicators (KPI) are then used to measure, respectively, when a process has achieved the goal set by management and when a goal is likely to be reached or not. Since COBIT does not suggest any technical solution but only organizational solutions, organizations combine COBIT and ISO 17799, applying the controls suggested in the part *Code of Practice for Information Security Management* of the standard.

Although COBIT does not provide any practical solution for mitigating the risks, it requires the organization to implement a Business Continuity Plan (BCP) to realize and improve the availability of an information system and its core processes. Until recently, no methodology was available to conduct in a precise way this activity although it is of primary importance when running a complex information system. The new standard for managing business continuity BS25999 [16] is mainly focused on providing guidelines to understand, develop and implement a BCP, and aims to provide a standard methodology. This standard requires the organization to complete different steps when preparing the BCP: (1) identify the activities/processes which carry the core service used by the organization, (2) identify the relationships/dependencies among themselves, (3) evaluate the impact of the disruption of the core services/processes previously identified (Business Impact Analysis, BIA). The most critical activities/processes are intended to be the ones whose direct/indirect monetary loss is significantly high.

When the risk has been assessed and evaluated, one has to identify the best countermeasures to reduce the risk. Typically, there exists a number of different solutions (technical or organisational) from which business and IT managers must choose the best one(s) meeting the required security level given the available budget (or finding the best compromise between the cost of the countermeasures and the benefit they provide). As we mentioned before, current methodologies are not sufficiently taking into account how business processes are linked together and the way a single incident could propagate and affect the whole organization's information system. The fact that COBIT and ISO 17799 do not consider dependencies between processes has even greater impact in the mitigation phase of availability risks: it is standard practice to protect the processes whose availability has a greater *direct* impact on the organization goals, while a more accurate analysis in many cases reveals that it is more cost effective to protect some of the processes that have an *indirect* impact as well.

III. MODELLING ORGANIZATIONS AND INCIDENTS

The framework we propose is based on the TD model, a model of the organization's IT-related architecture (including a part related to the organization's business goals), and a representation of the possible incidents. To simplify the exposition, we indicate by \mathbb{R}^+ the set of positive real numbers, and we use the following sets to indicate domains: \mathbb{T} is the set of all time intervals (expressed in hours), $\mathbb{E}ur$ is the domain of monetary values (expressed in Euro).

Assumptions We start by providing a brief summary of the data we need to build the model, later we describe this data in more detail. (1) A *Time-Dependency (TD) model*, consisting of: a set of *entities* (processes, applications, etc.) and a set of *relationships* between these entities. Relationships model which entities depend on other entities and must contain an estimate of *how long* an entity would be able to survive if another entity it depends on becomes unavailable. We express

this measure in hours. (2) The *cost* associated to the downtime of those processes *directly* affecting the business objective of the organization (indirect relationships are taken care of by the model). We express this measure in Euro per hour. (3) A list of possible incidents affecting the IT infrastructure, together with a conservative estimate of the average downtime each of them cause (per entity), given the controls already in place. We also need an estimate of their expected frequency. For the sake of uniformity, in the sequel we express the downtime caused by each incident in hours and their estimated frequency in times per year. (4) A list of *countermeasures*. For each countermeasure we need an estimate of (a) their deployment and maintenance costs (expressed in Euro per year), (b) the effect it has on the estimated frequency of the incidents and/or on the downtime they cause.

In Section V we address the problem of how and when this data can be collected during the RA and BCP processes.

TD model The basic elements of the model are the constituents of the IT infrastructure. We follow notable architecture frameworks such as TOGAF [29], Zachman [30] and ArchiMate [3] as well as IT Governance solutions (IBM [12] and ISACA [11]), to determine those elements which may directly or indirectly be involved in an incident: *Processes, Applications and Information, Technology and Infrastructure or Facilities*. *Processes* describe critical processes necessary to carry out the business, like manage orders or invoicing. *Applications and Information* are objects related to the software necessary to enable business operations e.g. production control applications, customer relationship management (CRM) applications or critical databases. *Technology* refers to systems, networks and industry-specific technology needed to enable applications and data, and *Infrastructure or Facilities* are physical locations necessary to house service technologies.

Running example - Part 1: We present here an example (intentionally oversimplified) of the business/IT infrastructure of a small bank segment with ten entities (see Table I):

TABLE I
LIST OF ENTITIES IN A SIMPLE ENTERPRISE ORGANIZATION'S SEGMENT.

Id	Description
p_1	Customer management process
p_2	Financial services process
a_1	Home banking application
a_2	On-line trading application
a_3	Financial funds management application
db_1	Checking account database
db_2	Trading database
m_1	Application server machine
m_2	Oracle machine
m_3	Oracle machine
n_1	Network segment

p_1 and p_2 represent two business processes; a_1 , a_2 and a_3 are three applications supporting business processes while db_1 and db_2 are two databases accessed by applications. Finally, m_1 , m_2 and m_3 are the three machines running applications and n_1 is the network segment connecting the three machines.

We represent a TD model using a graph, where nodes represent the basic entities and labelled edges between nodes

represent their relationships. The presence of an edge from node a to node b indicates that b depends on a , and that if a becomes unavailable for long enough, b will become unavailable as well. To model this correctly, we also need to indicate *how long* b will be able to survive without the presence of a . We do that by annotating each edge with the *survival time*: the time span the dependent entity can survive if the other one fails. While for some relationships, such as the dependency of an application onto the machine it runs on, this amount is obviously set to zero, in case of dependencies between applications this can vary between zero and several hours (e.g. in case that an application needs to be fed by another one with data at regular time intervals). Sometimes it is possible to extract this information from the functional requirements documentation or from the SLA specification. Although one can argue that these values could change over time, we have empirically verified (by inspecting documentation of several enterprise organizations) that this is not usually the case: organizations do not require such a level of detail yet.

Definition 3.1: A TD model is a pair $\langle N, \rightarrow \rangle$ where N is a set of nodes and $\rightarrow \subseteq N \times N \times \mathbb{T}$.

We write $n_1 \xrightarrow{t} n_2$ as shorthand for $(n_1, n_2, t) \in \rightarrow$.

A TD model allows one to express e.g. the dependencies of hardware components on the physical environment they are located in, the dependency of an application on the machines it runs on, and the dependency of a business process on the applications supporting it. We will show in Section V that this graph can be built in a fully automatic way.

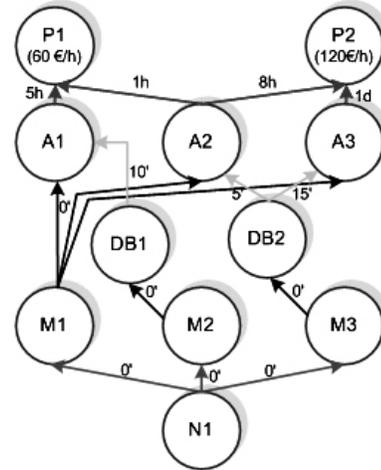


Fig. 1. A TD model example

Running example - Part 2: Figure 1 shows a TD model built with the entities from Table I. The edges connecting n_1 to m_1 , m_2 and m_3 express the dependency of the machines on the network connection with other machines. The connections from m_1 to a_1 , a_2 and a_3 , from m_2 to db_1 and from m_3 to db_2 express the dependency of software processes (applications or databases) on the machines they run on. For all of these

connections the survival time is set to zero, since no entity can survive the disruption of the ones it depends on, not even for a short time. In turn, p_1 depends on both a_1 and a_2 , since the customer management is achieved by providing on-line banking and trading, but with different time constraints (five hours for a_1 and only one hour for a_2). Similar reasoning apply to a_1 and p_2 .

Notice that dependency relationships are *and* relationships: a node depending on two or more other nodes is disrupted even if just one of these are affected by an incident. For the sake of simplicity, in this work we do not consider *or* relationships, even though it would be simple to include them in our model.

The number of entities can be very large in a real business environment. However, the information needed to build the model is already available after a RA (the first RA step, according to NIST methodology [28], is system characterization). For instance, the KARISMA tool developed at KPMG to support RA requires – among other things – the collection of enough data to build an accurate TD model. Any other tool based on the same standard methodology will basically do the same.

Incidents and their propagation Once the model of the architecture is defined, it is possible to simulate the availability of the system during and after the occurrence of an incident. We define incidents as events causing the unavailability of a given set of resources for a given time.

Definition 3.2 (Incident): Let $org = \langle N, \rightarrow \rangle$ be an organization. An incident i for org is a mapping $i : N \rightarrow \mathbb{T}$.

For instance, if we expect that the average occurrence of incident i would bring down machine m_1 for 3 hours, we model this by setting $i(m_1) = 3$.

Running example - Part 3: Let us now introduce three different incidents affecting the availability of m_3 : Table II presents them. In i_1 one of m_3 's hard disks is broken and the downtime is the average time required to replace the broken disk and restore data. i_2 consists of a power disruption in the building hosting m_3 , in this case the downtime is the average duration of a power disruption. i_3 consists in an OS failure, due to software bugs, causing the consequent freeze of applications running in m_3 and the downtime is the average time needed to detect the incident and reboot m_3 .

TABLE II
A LIST OF INCIDENTS POSSIBLY AFFECTING m_3 .

Id	Description	Target	Downtime
i_1	Disk failure	m_3	9h
i_2	Power disruption	m_3	3h
i_3	OS failure	m_3	2h

Every incident directly involves one or more entities, causing them to be unavailable for a certain amount of time. During this time, the incident may propagate to other entities, following the TD model.

Definition 3.3: We say that an incident *propagates* from a node n_1 to n_2 , if they have a functional relationship and

the unavailability time of n_1 , due to the incident, exceeds the survival time of n_2 w.r.t. n_1 , causing it to become unavailable until the incident is resolved.

Running example - Part 4: Figure 2 shows how i_1 propagates across our organization.

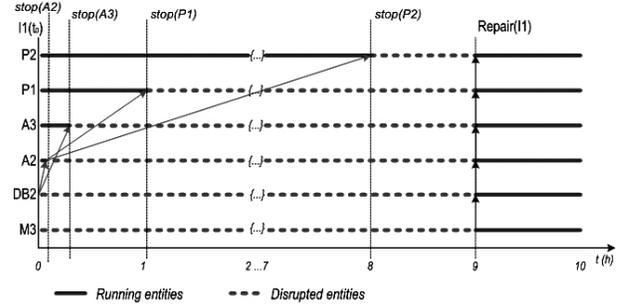


Fig. 2. Propagation chart of incident i_1 .

Assume that i_1 occurs at $t = 0$: i_1 brings down m_3 ; at the same time db_2 becomes unavailable, since its survival time w.r.t. m_3 is zero. After five minutes a_2 goes down and a_3 follows after fifteen minutes. Accordingly to the TD model, after one hour from the disruption of a_2 , the process p_1 goes down and after eight hours p_2 goes down as well. After i_1 has been repaired, nine hours after t_0 , all entities are repaired in turn.

Downtime With this information, we can finally define $Downtime(i, n)$: the downtime caused by incident i on node n (including propagation). This is the crucial information needed in the Risk Evaluation and Mitigation phases by evaluating the global consequences of an incident, as we will address in Section IV.

IV. A MODEL FOR RISK MITIGATION

The system we introduced in Section III allows us to model the propagation of incidents. We now show how we can use this information for selecting the best set of countermeasures; technically we aim at finding the set of countermeasures which minimizes the cost due to the forecasted downtime of relevant business processes.

A. Risk Evaluation

The first step toward Risk Mitigation is an accurate evaluation of the *costs* associated to the downtime of each process. In an organization, there are usually only few processes which – if unavailable – directly cause a real damage (in our running example, only p_1 and p_2). Clearly, this cost depends on the business goals of the company (a one hour downtime of the web server has a much higher monetary cost at Google than at an insurance company with comparable revenues). To model the cost of incidents we now define the *damage evaluation* function, relating the disruption time to the (monetary) loss affecting the organization.

Definition 4.1: Let $org = \langle N, \rightarrow \rangle$ be an organization. The *Business-driven damage evaluation function* (D) is a mapping from downtime to costs $D : N \times \mathbb{T} \rightarrow \mathbb{Eur}$.

In our simplified example, the downtime cost of p_2 is 120 Euro per hour (see Figure 1), so $D(p_2, x) = 120 \times x$. This means that the occurrence of an incident i_1 (which – after propagation – causes a downtime of 55 minutes on p_2) would create a damage of 110 Euro. In practice, D may not be linear (a downtime of 24 hours may well cause more losses than 24 downtimes of one hour). In general, D should be provided by the organization’s *business* department for the most important business processes and, in general, for all the business-relevant entities in the organization. One can argue that providing an accurate D function can be a time expensive task. In our experience this does not represent a particular problem, as the D function need to be defined only for the few business critical processes.

Frequencies and Global Cost Having determined the cost associated to an incident, we need now just one last factor for an accurate risk evaluation, and that is an assessment of frequency (likelihood) of an incident.

Definition 4.2: Given a set of incidents I , the incident *frequency*, $Freq(i)$, is a mapping $I \rightarrow \mathbb{R}^+$.

For instance, $Freq(x) = 0.1$ means that estimates indicate that incident x is likely to happen once in ten years. We should mention that NIST [28], [7] suggests a qualitative approach to assess likelihood (High, Medium, Low), while COBIT [11] promotes both qualitative and quantitative approaches. For our goals, we require a numerical value, which in practice can be derived from the past experiences of the assessment team or from public domain statistics.

Now, the downtime function computed using the TD model together with the damage and the frequency evaluation allows us to compute the expected cost (per year) due to service downtime for the whole organization.

Definition 4.3: Let $org = \langle N, \rightarrow \rangle$ be an organization, I be a set of incidents and $D(n, t)$ the damage evaluation for org . The *estimated downtime cost* for the system is defined as

$$Esdc(I) = \sum_{i, n \in I \times N} D(n, Downtime(i, n)) \times Freq(i) \quad (1)$$

Going back to our example, if we estimate i_1 , i_2 and i_3 can happen respectively 5, 12 and 50 times per year, then the estimated downtime cost of the system is approximately 10,000 Euro.

B. Risk Mitigation

The goal of Risk Mitigation is to bring down the estimated downtime cost by applying a set of *countermeasures*, which can be either technical or organizational. To achieve full generality we define a countermeasure as a function which modifies the organization, the set of incidents as well as their frequencies. Each countermeasure has also a cost per year (summing the amortization and the maintenance costs).

Definition 4.4 (Countermeasure): Let $org = \langle N, \rightarrow \rangle$ be an organization, I be a set of incidents and $Freq$ be the frequency estimate for I . A *countermeasure* c , is a pair $\langle m, cost \rangle$ where m maps $org, I, Freq$ into $org', I', Freq'$, and $cost \in \mathbb{Eur}$ is the cost per time unit (year).

We note that in practice most countermeasures fall into one of two classes: *frequency countermeasures* and *time countermeasures*, accordingly to the resulting effect. The former reduce the frequency of a given incident, while the latter reduce the downtime due to the incident. In *frequency countermeasures*, the projection of m on org', I' is the identity function. It is worth noting that a countermeasure completely preventing an incident can be modelled by setting to zero either the frequency or the downtime relative to the incident.

Running example - Part 5: Table III reports a list of countermeasures to be applied on m_3 to mitigate the negative effects of incidents i_1 - i_3 . Notice that c_1 - c_7 are technical

TABLE III
A LIST OF COUNTERMEASURES TO BE APPLIED ON m_3 TO MITIGATE THE NEGATIVE EFFECTS OF INCIDENTS i_1 - i_3 .

Id	Description	Cost €/y	I	Freq		Downt.	
				bef	aft	bef	aft
c_1	New disks	1000	i_1	3	5	9	9
c_2	UPS	3000	i_2	12	12	1	3
c_3	Backup machine	4000	I	-	-	2	-
c_4	Service pack	900	i_3	20	50	2	2
c_5	New OS version	6200	i_3	5	50	2	2
c_6	Patch #143	300	i_3	40	50	2	2
c_7	Patch #146	300	i_3	42	50	2	2
c_8	Disk backup strategy	2000	i_1	5	5	5	9

countermeasures while c_8 is organizational; moreover c_1, c_4 - c_7 are frequency countermeasures since their effect is to reduce the frequency of certain incidents, while c_2, c_3 and c_8 are time countermeasures since they reduce the downtime of m_3 . Figure 3 shows the propagation of incident i_1 after the application of c_8 , which reduces the downtime of m_3 to five hours. Since the survival time of p_2 (eight hours) is longer than the downtime of a_2, p_2 is never disrupted by this incident, and the component relative to p_2 of the cost of i_1 is zeroed, reducing the overall estimated downtime cost.

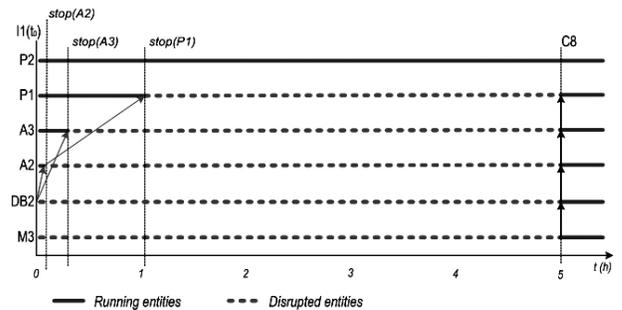


Fig. 3. Propagation chart of incident i_1 with countermeasure c_8 in place.

It is usually possible to apply more than one countermeasure on the same entity, but for this we have to consider that one

countermeasure may be *incompatible* with another one. An OS patch, for example, can be incompatible with other patches; moreover, deploying a backup machine can be useless if other backup techniques are already in place.

For instance, in our Running Example, countermeasures c_4 - c_7 are mutually incompatible because the service pack can not be installed if single patches are already installed, and because installing patches for the old OS version with the new version already installed would be impossible.

By combining the TD model, countermeasures and incidents with their cost and frequency, we now give a formal definition of *best set of countermeasures* as the set of countermeasures that reduces the most the estimated downtime cost (taking into account the cost of the countermeasures). In the following definition we extend (1) to take into account the selected countermeasures. We also denote by $Downtime[C](i, n)$ the downtime the incident i causes on node n in presence of countermeasures $C = \{c_1, \dots, c_n\}$. Likewise, $freq[C](i)$ is the expected frequency of incident i presence of countermeasures $C = \{c_1, \dots, c_n\}$.

Definition 4.5: Let org be an organization, I be a set of incidents, $Freq$ be the frequency estimate for I , and C be a set of countermeasures.

- We call the *estimated global cost* of incidents I in presence of C , $Esdc(I, C)$ the value: $\sum_{i, n \in I \times N} D(n, Downtime[C](i, n)) \times Freq[C](i)$
- We say that $BC \subseteq C$ is a *best set of countermeasures* (w.r.t. C) if the countermeasures in BC are pairwise compatible, and for every $D \subseteq C$ of pairwise compatible countermeasures, $Esdc(I, BC) \leq Esdc(I, D)$.

Thus, the best set of countermeasures is the one minimizing the expected global cost. Similarly, the *expected benefit* of a given set of countermeasures is the difference between the expected downtime cost $Esdc(I)$ and the expected downtime cost after applying the countermeasures: $Esdc(I, BC)$.

Running example - Part 6: We can now compute $Esdc$ considering the three incidents (i_1 - i_3) and each possible combination of countermeasures (c_1 - c_8). Recall that only the disruption of p_1 and p_2 involve a loss to the organization (see Figure 1). The result is $BC = \{c_1, c_4\}$, i.e. the most cost-effective strategy to mitigate the risk is to install the OS service pack and to update m_3 's disks.

Summarizing, our model provides IT managers with an effective way of choosing the best set of countermeasures for a given system. For space reasons, we have not addressed other optimization possibilities which are made possible by this model, but it is easy to see that one can use it to find for instance “the least expensive set of countermeasures which bring the expected downtime of service A down to 10 hours per year” or “the best set of countermeasures within a given budget”.

V. FEASIBILITY

In this section we argue that the approach we propose is *feasible*. In particular, we show that the vast majority of the

input data required by our tool is readily available after a RA and a BCP, and that it is possible to cope with the complexity of the algorithms the tool requires.

Input Data The main concern regarding the feasibility of our approach is whether the set of data it requires is easy to collect. If this was not the case, organizations would not be willing to accept it. Fortunately, the data it requires is typically available after RA and BCP: first of all, an accurate map of the IT infrastructure is readily available after a BCP carried out following the BS25999 [16] standard (and is also available after standard RAs). Secondly, an inventory of possible incidents, together with their frequency *has* to be compiled during the RA. Finally, a BCP should provide (according to BS25999 standard) a complete evaluation of the effectiveness of chosen incident response strategies (i.e. countermeasures): thus, the organization is also required to quantify downtime costs of the different entities before and after the countermeasures have been applied. It is common practice, and a well-known principle in RM [28], to derive this information from previous observations (history of attacks): the more accurate these observations are, the more precise the assessment outcome is.

To further substantiate our argument, we note that this data is also collected by tools devised to assist the RA and RM processes. For instance, the Italian branch of KPMG [20] (a worldwide company delivering also Information Risk Advisory services) has developed a customizable tool, KARISMA (Kpmg Advanced RISK Management), to support their RA activities. Among the information KARISMA collects via a question-driven procedure, there is a map of the business process entities (together with their relationships) and the Business Impact Analysis values. KARISMA is based on COBIT, and it is very likely that other tools for RA based on COBIT would collect the same information. Our system can thus be regarded as an additional component for KARISMA or for any other COBIT-based tool for RA, supporting in particular the Business Continuity Planning activity.

We also note that most of the information required to build the TD model is also available when applying to an organization an architectural framework, such as TOGAF [29], Zachman [30] and ArchiMate [3]. Indeed, the layers defined in those frameworks are similar to the ones we adopt for our model, though used for different purposes (e.g. architectural support, new component impact evaluation, etc.). Since those project are widely employed (ArchiMate for instance is used by ABN Amro and the Dutch Tax Office), and are supported by several tools, they provide us an indirect confirmation of the feasibility of actually obtaining the data needed by our model.

Summarizing, our tool does not require organizations to acquire new information (i.e. to employ new resources), rather it uses in a different way the information already available after RA and BCP.

Computational complexity The second concern regarding the feasibility of our approach is whether the algorithms

underlying our framework are not too complex to be carried out in reasonable time. It is easy to see that – even if we assume that the organizational graph is acyclic – evaluating the optimal of countermeasures has complexity in the order of $(e \times r \times i \times c!)$ where e is the total number of entities, r is the total number of relationships between entities, i is the total number of possible incidents and, finally, c is the total number of possible countermeasures. The presence of cycles could increase the complexity, but we believe that practical situations present graphs that can be rendered acyclic after some preprocessing. The only problematic factor in the equation is of $c!$, which indicates that the presence of a relatively large set of countermeasures would make it infeasible to carry out a brute-force analysis to find the best set of countermeasures. Presently, we are working at a brute force implementation which is already giving satisfactory results on real datasets, and we have developed heuristics based algorithms finding a *local* optimum whose complexity is $(e \times r \times i \times c^3)$, which give very satisfactory results (in our experiments, the local optimum always coincides with the global optimum). Other ways to bring down the $c!$ include automatically splitting the set of countermeasures into various set of independent countermeasures, which will make it possible to apply compositional methods.

VI. IMPLEMENTATION

A preliminary problem we had to solve when tackling the implementation issue is that of automatically building the TD model. The information about the IT and business infrastructure is typically spread across a number of free text documents. To build the model it is necessary to report information in a structured form, such as database tables. Fortunately there exist tools for supporting the RA (such as KARISMA) which can deliver this information in a structured format. We automatically build the TD model by representing each entry in the entity table of the dataset with a node and each entry in the link table with an edge between nodes, annotated with the survival time.

The actual implementation requires us to realize an algorithm which (a) explores the TD model to simulate the consequences of the incidents, (b) evaluates the global cost of a set of incidents, (c) simulates the new behaviour of the TD model in presence of a set of countermeasures and (d) evaluates the new global cost of the set of incidents with different subsets of countermeasures.

To realize this we use in first instance model checking [9], which is a technique to algorithmically analyse concurrent systems, typically used for verifying if (a model of) the system satisfies some given properties, often specified as a temporal logic formulas. The reason of this choice is that model checkers are already devised to quickly explore a graph of several (thousands of) possible system behavioural traces, to find the one realizing a given property. Therefore, model checkers provide us with a way of doing fast prototyping without sacrificing performance too much. Among the several model checkers available (e.g., SPIN [15], SMV [24], etc.) we

adopt UPPAAL [22], because (1) it allows to specify a time dependent system (such as the one we need to model) and (2) its extension UPPAAL CORA allows to solve optimization problems such as those previously required in points (b) and (d).

UPPAAL requires the system to be specified as a timed automaton [9], [6], which is a finite automaton extended with a finite set of real-valued *clocks*. Clock constraints, i.e. guards on edges, are used to restrict the behaviour of the automaton. UPPAAL CORA, is an extension of UPPAAL for cost optimal reachability analysis which applies the theory of Linearly Priced Timed Automata (LPTA) [21]. LPTA extend the model of timed automata with prices on all edges and locations. In these models, the cost of taking an edge is the price associated with it, and the price of a location gives the cost-rate applied when delaying in that location. In UPPAAL CORA prices are defined by means of an implicit monotonically growing variable called *cost*.

UPPAAL has the additional advantage of allowing us to map in a very natural and straightforward way every element of our model into a timed automaton with the same behaviour. This one-to-one translation ensures the absence of side effects due to the implementation. For the sake of presentation we do not report here further implementation details.

To test our implementation we use a dataset related to a real insurance company collected by KPMG auditors using KARISMA during a RA. The dataset contains all the information needed to build the TD model (19 macro business processes and 122 sub-processes); the remaining information (about incidents, costs and countermeasures) is also provided by the KPMG auditing team who conducted the assessment. In first instance, to avoid the state explosion problem and maintain a reasonable computational time, we perform the analysis on portions of the infrastructure, and then merge results. In second instance we realize a translation of the UPPAAL model into Prolog. This second implementation allows us to deal with the entire dataset at once, without splitting the IT infrastructure, and tens of incidents while maintaining the computational time in the order of minutes. We carry out optimal analysis for partitions of up to 18 countermeasures and a suboptimal analysis that can deal with thousands of them, on a 3GHz Pentium IV machine with 1Gb RAM.

VII. RELATED WORK

There exist various academic frameworks for carrying out RA, but they all differ from our proposal in that they do not model the propagation of incidents across an organization as precisely as we do. For instance, Lenstra and Voss [23] present a quantitative approach to IT risk management to determine the optimal RM strategy given a limited budget. Their approach requires performing a risk assessment on all the applications supporting business processes and identifying the (monetary) loss due to each threat on the business process they support, thus the risk is evaluated in terms of the likelihood and the loss. Authors define an action plan (set of countermeasures) as

something influencing the likelihood of a threat thus reducing the risk; furthermore they associate a cost to it. The selection of the best set of action plans consists in finding the set that mostly reduces the likelihood of all threats within a given budget. Since this approach is designed to deal with threats to all the three aspects of information security (CIA), to keep it feasible it lacks in a complete representation of the constituents of an IT infrastructure (machines, facilities, etc.) and in modelling the time dependencies between them, which - as we have discussed in the introduction - is essential for properly modelling the availability risks. Our model, on the other hand, being specifically tailored for availability risks, takes into consideration the time dependencies and therefore allows us to simulate how an incident propagates across the organization.

Furthermore, the authors' choice of allowing a single, atomic, action plan per threat implies that the risk management team should already have found manually the best set of countermeasures to be applied in response to an incident. The proposed framework then, simply decides to apply or not this set of countermeasures. On the other hand, our model is able to compute the best set of countermeasures without requiring any pre-processing phase and allowing one to find a more fine-grained solution.

Asnar and Giorgini [4] introduce an extended Tropos [8] goal model to analyse risk at organization level and to identify and enumerate relevant countermeasures for RM. Their approach is mainly devoted to the enumeration of incidents and countermeasures, while our approach focuses on selecting and prioritizing incidents to be mitigated and possible countermeasures to perform the mitigation. Another proposal is that of Aagedal et al. [1], who developed the CORAS framework to produce an improved methodology for precise, unambiguous, and efficient risk analysis of security critical systems. CORAS focuses on the tight integration of viewpoint-oriented visual modelling in the RA process, using an UML-based approach in the context of security and RA. Our approach is orthogonal to CORAS, in the sense that we could use the output of CORAS to feed out tool.

In addition to academic work there exist a number of commercial tools supporting the Risk Management and RM process. The most closely related to our work are CounterMeasures and GSTool. Alion's CounterMeasures [2] performs Risk Management based on the NIST 800 series and OMB Circular A-130 USA standards. It provides the ability to perform cost/benefit analysis and ROI on countermeasures. GSTool [14] is developed by Federal Office for Information Security (BSI) to assist users of the IT Baseline Protection Manual. GSTool supports a qualitative assessment of protection requirements. The main difference between these approaches and ours is that they face the countermeasures selection by an economic prospective (ROI) or a technical prospective only, rather we merge the two aspects in an holistic behavioural model of the whole organization. For a wider list of Risk Management supporting tools refer to [13].

Finally, our work has some analogy with some proposal for

using model checking to assess the survivability of distributed systems [19], [10]. Jha and Wing [19] use the NuSMV model checker to model the distributed environment and generate a failure scenario graph (sum of counterexamples of survivability properties) by injecting faults into the model. Secondly, they add some additional information about the probability of harmful events to perform reliability analysis and cost/benefit analysis of possible countermeasures. Our approach differs in that we model also time dependencies between entities: thus we are able to perform a more accurate evaluation of the *global impact*. Furthermore our approach is strictly focused on information Risk Management. Cloth and Haverkort [10] develop a model checking-based approach to evaluate the survivability of a system. Survivability is defined as the ability of a system to recover in a timely manner predefined service levels after the occurrence of a disaster. They describe the system as a Stochastic Petri Net and then automatically convert it into a Continuous Time Markov Chain (CTMC). Finally they use a model checking engine to obtain a time-probability chart that expresses the recovery probability in relation to the recovery time.

VIII. DISCUSSION, FUTURE WORK AND LIMITATIONS

In this paper we focus on the mitigation of risks related to the *availability* of an organization's IT infrastructure. We argue that the way present methodologies address the time and functional relationships between the constituents of the IT infrastructure is inadequate to properly evaluate the global consequences of an incident. Our contribution consists of a methodology and a tool for carrying out a Risk Mitigation activity which allows to assess the *global impact* of a set of risks and to choose the best set of countermeasures to cope with them. This is achieved by employing the TD model that allows us to represent the actual propagation of an incident across the organization and to deal with the countermeasures selection process. To this end, the presence of a tool is necessary due to the complexity of the selection process.

We argue that the input required by our approach is typically already available after a serious RA and BCP assessments; this makes our proposal attractive as it does not require the collection of new information. Indeed we believe that our approach can be integrated in Risk Management methodologies to provide a more precise Risk Mitigation activity.

Our approach is aimed at finding the set of countermeasures minimizing the expected yearly cost due to the unavailability of IT services. Here we note that a related organization goal is that of achieving a given Recovery Time Objective (RTO), i.e. the latest point in time at which operation must resume after a failure. While this does not reduce the value of our proposal, we believe our model for incident propagation can be extended to analyse the required steps to achieve the given RTO. This is one of the targets for our future work.

Actually, our present system could already be used for this purpose by employing a cost function which is zero before the RTO and very high after the RTO. However, we must warn

that using such a highly discontinuous cost function may result in an inaccurate analysis.

Finally, our system is particularly suited to support continuous risk management [25]: thanks to its fine granularity, it can be easily reviewed to match situational changes, allowing for early detection of service deterioration, and prompt reaction to changing environments.

ACKNOWLEDGEMENTS

We thank Pascal Van Eck and Pieter Hartel for their valuable comments.

REFERENCES

- [1] J. Ø. Aagedal, F. den Braber, T. Dimitrakos, B. A. Gran, D. Raptis, and K. Stölen. Model-Based Risk Assessment to Improve Enterprise Security. In *EDOC '02: Proc. 6th International Enterprise Distributed Object Computing Conference*, pages 51–63. IEEE Computer Society, 2002.
- [2] Alion Science and Technology. CounterMeasures. <http://www.countermeasures.com>.
- [3] The ArchiMate project. <http://archimate.telin.nl>.
- [4] Y. Asnar and P. Giorgini. Modelling Risk and Identifying Countermeasure in Organizations. Technical report, University of Trento, 2006. oai:UNITN.Eprints:1035.
- [5] Basel II: Revised international capital framework, 2005. <http://www.bis.org/publ/bcbsca.htm>.
- [6] J. Bengtsson and W. Yi. Timed Automata: Semantics, Algorithms and Tools. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *LNCS*, pages 87–124. Springer-Verlag, 2003.
- [7] P. Bowen, J. Hash, and M. Wilson. Information Security Handbook: A Guide for Managers. Technical report, NIST, 2006. SP 800-100.
- [8] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. TROPOS: An Agent-Oriented Software Development Methodology. Technical report, University of Trento, 2002. oai:UNITN.Eprints:84.
- [9] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.
- [10] L. Cloth and B. R. Haverkort. Model Checking for Survivability. In *Proc. 2nd Int. Conference on the Quantitative Evaluation of Systems QEST'05*, pages 145–154. IEEE Computer Society, 2005.
- [11] CobiT: Control Objectives for Information and related Technology. <http://www.isaca.org>.
- [12] R. Cocchiara. Beyond disaster recovery: becoming a resilient business. Technical report, IBM, 2005. <http://ibm.com/services/its/resilience>.
- [13] ENISA Technical Department. Risk Management: Implementation principles and Inventories for Risk Management/Risk Assessment methods and tools. Technical report, European Network and Information Security Agency (ENISA), 2006. http://www.enisa.europa.eu/rmra/rm_home.html.
- [14] Federal Office for Information Security (BSI). GSTool. <http://www.bsi.bund.de/english/gstool/>.
- [15] G. J. Holzmann. *The SPIN model checker*. Addison-Wesley, 2003.
- [16] British Standards Institute. Business continuity management - Part1: Code of practice. Technical Report 25999-1, BSI, 2006.
- [17] ISO. Information Technology - Security techniques - Guidelines for the management of IT security. Technical Report 13335, ISO/IEC, 2001.
- [18] ISO/IEC 17799:2005 Information Security - Code of Practice for Information Security Management, 2000. <http://www.iso.org>.
- [19] S. Jha and J. M. Wing. Survivability analysis of networked systems. In *Proc. 23rd Int. Conference on Software Engineering (ICSE '01)*, pages 307–317. IEEE Computer Society, 2001.
- [20] <http://www.kpmg.com>.
- [21] K. Larsen, G. Behrmann, E. Brinksma, A. Fehnker, T. Hune, P. Pettersson, and J. Romijn. As Cheap as Possible: Efficient Cost-Optimal Reachability for Priced Timed Automata. *LNCS*, 2102:493–506, 2001.
- [22] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, 1997.
- [23] A. Lenstra and T. Voss. Information Security Risk Assessment, Aggregation, and Mitigation. In *ACISP: Information Security and Privacy: Australasian Conference*, 2004.
- [24] Z. Liu and M. Joseph. Verification of Fault Tolerance and Real Time. In *26th IEEE Symposium on Fault Tolerant Computing Systems (FTCS-26)*, pages 220–229. IEEE Computer Society, 1996.
- [25] R. L. Murphy, C. J. Alberts, R. C. Williams, R. P. Higuera, A. J. Dorofee, and J. A. Walker. *Continuous Risk Management Guidebook*. Carnegie Mellon Software Engineering Institute, 1996.
- [26] OCTAVE risk methodology. <http://www.cert.org/octave/>.
- [27] M. C. Paulk, C. V. Weber, B. Curtis, and M. B. Chrissis. *The capability maturity model: guidelines for improving the software process*. Addison-Wesley Longman Publishing Co., Inc., 1995.
- [28] G. Stoneburner, A. Goguen, and A. Feringa. Risk Management Guide for Information Technology Systems. Technical report, NIST, 2002. SP 800-30.
- [29] The Open Group. TOGAF (The Open Group Architecture Framework), 2003. <http://www.opengroup.org/architecture/togaf8-doc/arch/>.
- [30] The Zachman Institute for Framework Advancement. Zachman Framework, 2007. <http://www.zifa.com/>.