A model supporting Business Continuity auditing & planning in Information Systems

Emmanuele Zambon[1], Damiano Bolzoni[1], Sandro Etalle[1], Marco Salvato[2*]

[1] *University of Twente, {emmanuele.zambon, damiano.bolzoni, sandro.etalle}@utwente.nl*
[2] *KPMG Italia S.p.a., msalvato@kpmg.it*

Abstract

One of the main tasks of IT business continuity planing (BCP) is guaranteeing that incidents affecting the IT infrastructure do not affect the availability of IT-dependent business processes beyond a given acceptable extent. Carrying out BCP of information systems is particularly challenging because it has to take into consideration the numerous interdependencies between IT assets typically present in an IT-based organization. In this paper we present a model and a tool supporting BCP auditing by allowing IT personnel to estimate and validate the Recovery Time Objectives (to be) set on the various processes of the organization. Our tool can be integrated in COBIT-based risk assessment applications. Finally, we argue that our tool can be particularly useful for the continuous auditing of the BCP.

1 Introduction

Business Continuity (BC) is the discipline supporting an organization in coping with the disruptive events that may affect its IT infrastructure. The goal of BC is guaranteeing that – after incidents – the infrastructure will recover operations within a predefined time. This is achieved by carrying out a *Business Continuity Plan* (BCP), which is part of the Risk Mitigation phase of the Information Risk Management process. In general, Risk Mitigation (RM) consists in developing and implementing a strategy to manage potential harmful threats to the information systems. Since risk may not be completely avoided because of financial and practical limitations, RM (and BCP as well) includes the

evaluation and the conscious acceptance of a residual risk.

BC is quickly becoming best practice among both enterprises and organizations also due to recent legislation such as the *Sarbanes-Oxley Act* (SOX) of 2002 or the *Basel II* [2] accord, which explicitly requires it.

Until recently, no widely agreed methodology was available to carry out a BCP. The new standard BS25999 [7], published in 2006 by the British Standard Institute, has changed this situation providing guidelines to understand, develop and implement a BCP, and aims to become a standard methodology. Notably, BS25999 requires an organization to (1) identify the activities/processes supporting the core services used by the organization, (2) identify the relationships/dependencies between activities/processes, (3) evaluate the impact of the disruption of the core services/processes previously identified (during the Business Impact Analysis, BIA).

One of the main goals of any BCP is achieving that crucial business processes should recover from disruption within a predefined *Maximum Tolerable Period of Disruption* (MTPD). The MTPD expresses the maximum acceptable downtime to guarantee the business continuity. As expected, the MTPD depends heavily on the organization business goals and therefore is defined *on the business processes*, and is determined *by the business unit*.

Since business processes typically depend on a variety of underlying IT assets, the MTPD has a direct and indirect impact on the maximum downtime that these assets may exhibit in practice. Indeed, the standard technical mean to realize a given MTPD is to define *Recovery Time Objectives* (RTOs) on all IT assets supporting business activities for which the BIA has determined that it is necessary ensure continuity; RTOs strongly depend on the technical and organizational measures the IT department implements to deal with incidents.

Nowadays, determining the RTOs that apply to the

IT assets is done manually, and is a subjective work which heavily depends on the experience of the IT personnel. This is not only error-prone, but it does not scale well (to the point that it is very often not done altogether, despite being required by the standard methodology). Moreover, it is inconvenient in case of changes in the IT infrastructure or in the business goals. In particular, new contracts and agreements can have an impact on the quality of service a business process should deliver and ultimately on the MTPD associated to it. Likewise, changes in the IT infrastructure may affect dependencies and therefore the impact of the IT assets on the business MTPDs. In both cases, adapting the BCP to these changes usually requires a costly new analysis involving both the IT and business units of the organization.

We present a new model-based tool for supporting the analysis of temporal dependencies among IT assets and between IT assets and business process. The primary goals of our model and tool are (1) to support the IT department in setting and validating the RTOs of the IT assets of the organization (2) to evaluate assigned RTOs w.r.t. the given MTPD to find critical points in the IT infrastructure. Ultimately, our model allows to put down the fine-grained set of premises and assumptions to infer that a given MTPD will be achieved, thereby obtaining a more objective assessment of the behaviour of the IT infrastructure.

While achieving these goals, we argue that our model is particularly useful for *continuously auditing* the BCP in various ways: first, the tool allows to visualize immediately how changes in business goals or in the IT infrastructure affect the compliance with given (or modified) MTPDs; in particular it is possible to compute whether the measures already in place continue giving enough guarantees also after the changes. Secondly, it allows to validate the actual response of the IT infrastructure w.r.t. the expected behaviour, allowing a continuous refinement of the model which can adapt to new external circumstances, allowing for early detection of new threats to the business continuity targets.

Technically, our model is an improvement of the one we presented in [18] for the optimization of countermeasures. The essential difference with the previous model lies in the modelling of the recovery time after disruption, which in the present situation has to be much more accurate. Notably, as we mention in Sec. 5, the data our model requires is collected anyhow during a BCP.

2 Time Dependency and Recovery model

We now present the *time dependency and recovery (TDR) model*, which allows us to model the response of an organization to incidents. This model is based on the TD model we introduced in [18] for the mitigation of availability risks. In principle, the main difference from the previous version is the way we model incidents and their response time; in practice, the tool supporting it has completely different functionalities. The basic elements of the model are the constituents of the IT infrastructure. The model is compatible with notable architecture frameworks such as TOGAF [16], Zachman [17] and ArchiMate [1] as well as IT Governance solutions (IBM [6] and ISACA [5]), to determine those elements, which may directly or indirectly be involved in an incident: *Processes*, *Applications and Information*, *Technology* and *Infrastructure or Facilities*.

We start by providing a brief summary of the data we need to build the model. (1) A representation of the organization, 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. (2) A list of possible incidents affecting the IT infrastructure, together with the time needed to repair them (per entity) given the controls already in place. We also need an estimate of their expected frequency, in times per year. (3) The *MTPD* value for each business process on the dependency graph. (4) Optionally, a first estimate of the *RTO* value for each entity (not business process) on the dependency graph. In Sec. 5 we address the problem of how and when this data can be collected during the RA and BC processes.

Let us formalize the main notions. For this, we indicate by $\mathbb{R}^+$ the set of positive real numbers, and by $\mathbb{T}$ is the set of all time intervals (expressed in hours).

We represent a TDR 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.

Definition 1 *A* TDR 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 TDR model expresses 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, the dependency of an application on another one feeding it at regular time intervals and the dependency of a business process on the applications supporting it. We will show in Sec. 4 that this graph can be built in a fully automatic way.

Running example - Part 1 *We present here an example (oversimplified, to fit the example in the format of the paper) of the business/IT infrastructure of a small bank segment with ten entities:*

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 founds management application
db_1	Checking account database
db_2	Trading database
m_1	Application server machine
m_2	DBMS machine
m_3	DBMS machine
n_1	Network segment

p_1 and p_2 represent two business processes; a_1, a_2 and a_3 are three applications and databases 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. Fig-
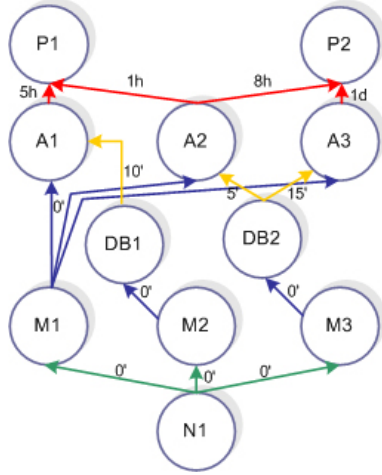

Figure 1. A TDR model example

ure 1 shows a TDR model *built with the entities from Table 1. 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 Internet banking and on-line trading, but with different time constraints (five hours for a_1 and only one hour for a_2). A 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.

Incidents and their propagation From the modelling side, the main innovation of the TDR model w.r.t. the TD model lies in the representation of incidents, which in the present case needs to be much more accurate. Here, an incident is an event causing an entity (or a set of entities) to break down and become unavailable. Therefore, we identify an incident with the set of entities it brings down. Since incidents can happen several times a year, business continuity should deal with the frequency of incidents to determine the proper strategies to be put in place.

Definition 2 *Given a set of incidents I, the* frequency estimate freq, *is a mapping $I \rightarrow \mathbb{R}^+$.*

Finally, every disruptive event on an entity takes some time to be repaired. Our model encompasses an estimate of the repair time rt that is required by the affected entities to become operational again. Here it is important to notice that in many cases it is difficult to guarantee a uniform repair time: repairing a disk could take up to two hours in most cases (say, 90% of the cases), but up to four hours in the remaining (exceptional) 10% of the cases. For instance, a software bug affecting a new application can be repaired in eight hours if it is discovered during the week, or within 24 hours during the week-end, due to lack of personnel. To be accurate, our model requires an estimate of the revocery time for both the general and the exceptional cases. For this reason rt is expressed as a frequency distribution.

Definition 3 (Repair Time) *Let $Org = \langle N, \rightarrow \rangle$ be an organization and I be a set of incidents. The* repair time rt *is a mapping $rt : N \times I \times \mathbb{T} \rightarrow \mathbb{R}^+$.*

The explicit modelling of the repair time as a frequency distribution is the main theoretical difference between this TDR model and the model we exposed in [18], since dealing with BC requires a higher detail level w.r.t. the average repair time needed to perform just a risk assessment. Interestingly, the functionalities of the tool we use here and those described in [18] are completely different.

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 TDR model.

Definition 4 *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 n_2 to become unavailable until the incident is resolved.*

Running example - Part 2 *The following graph shows how an incident affecting m_3 propagates across our organization*
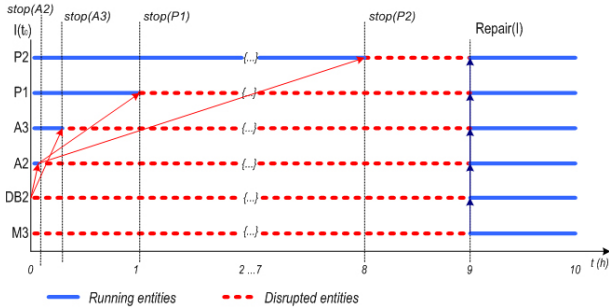
Assume that an incident i occurs at $t = 0$ and it is repaired within nine hours after t. It brings down m_3; at the same time db_2 becomes unavailable, since its survival time w.r.t. m_3 is zero. After five minutes a_2 goes down and and a_3 follows after fifteen minutes. Accordingly to the TDR model, after one hour from the disruption of a_2, 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, all entities are repaired in turn.

3 Assessing the RTO

Recall that our goal is assessing whether, during the normal operation, the system will comply with the Maximum Tolerable Period of Disruption that has been determined (by the business unit) for the business relevant processes. The formal definition is the following.

Definition 5 (MTPD) *Let $Org = \langle N, \rightarrow \rangle$ be an organization and $P \subset N$ be the set of business processes, the Maximum Tolerable Period of Disruption is a mapping $MTPD : P \rightarrow \mathbb{T}$.*

Running example - Part 3 *The two business processes in our example are noticeably time-dependent, because they both require customer interaction and, in the case of p_2, the operational disruption causes a direct financial loss to the bank. For these reasons it is reasonable to assume that the MTPD is very short, as reported on the following table.*

Id	Description	MTPD
p_1	Customer management process	3h
p_2	Financial services process	0.5h

3.1 Complying with the MTPD

One of our goals is to check under which circumstances we can expect to be able to comply with the MTPD (i.e., we can expect all business processes to recover from disruptions within the maximum time given by the MTPD). To this end, our model allows us to determine, given the MTPD for the business critical processes, what is the *maximum recovery time* that each entity in the TDR model has to respect. Assuming that the organizational graph is acyclic[1], this can be defined as follows.

Definition 6 (mrt) *Let $Org\langle N, \rightarrow \rangle$ be an organization, and let μ be an instance of the MTPD function for Org, then for each $n \in N$ we define the* maximum recovery time *of n (w.r.t. μ) $mrt_\mu(n)$ to be*

- $mrt_\mu(n) = \mu(n)$ *if n is a business process,*

- $mrt_\mu(n) = min\{mrt_\mu(m) + d \mid n \xrightarrow{d} m\}$

Assuming that the TDR model is faithful, i.e. that it reflects well how incidents propagate across the organization, the relevance of the maximum recovery time is given by the following result

Proposition 7 *Let $Org\langle N, \rightarrow \rangle$ be an organization, $P \subset N$ be the set of business processes, and μ be a MTPD for it. If an incident on entity $n \in N \smallsetminus P$ is not repaired within $mrt_\mu(n)$, then at least one business process $p \in P$ will be disrupted for longer than its MTPD. On the other hand, if an incident on entity $n \in N \smallsetminus P$ is repaired within $mrt_\mu(n)$, then no business process $p \in P$ will be disrupted for longer than its MTPD.*

Therefore, the $mrt_\mu(n)$ we have calculated is actually the maximum downtime we can tolerate on n to ensure that the MTPD is respected for each business process depending (directly or indirectly) on it. Of course, the

---

[1]In most cases, the organizational graph *is* acyclic, or can be made acyclic by some preprocessing.

validity of this result depends on the accuracy of the TDR model, but it is worth mentioning here that (a) as we discuss later, the data needed to build the TDR model is in most cases readily available and (b) the model can be refined in time by using statistics on incidents and their recovery.

Recovery Time Objectives While the MTPD is a high-level measure imposed on the critical business processes, it is widely acknowledged that is should be good practice (also required by standard BCP auditing) to set a *recovery time objective* on all the IT assets of the organization that can be directly involved in incidents (machines, applications, infrastructure, etc.). Our tool can be used to do this in an automatic, fairly user-friendly way. This already represent an improvement on everyday practices, in which RTO are often not set at all.

Definition 8 (RTO) *Let $Org = \langle N, \rightarrow \rangle$ be an organization and $P \subset N$ be the set of business processes, the Recovery Time Objective is a mapping $RTO : N \smallsetminus P \rightarrow \mathbb{T}$.*

Clearly, Proposition 7 implies that if for each node n $RTO(n) \leq mrt_{mt}(n)$, then the compliance with respect to the RTO implies compliance w.r.t. MTPD. Our model allows us to *validate* the RTO imposed on the organization as follows.

Proposition 9 *Let $Org\langle N, \rightarrow \rangle$ be an organization, and rto be an RTO for it. Assume that for each $n, m \in N$ such that $n \xrightarrow{d} m$ the following holds*

$$(1) \qquad RTO(n) \leq RTO(m) - d$$

Then for any two entities n and m holds that an incident on the entity $n \in N$ which causes on n a disruption shorter than $RTO(n)$ will never cause by propagation on m a disruption longer than $RTO(m)$.

In other words, with our model we can *validate* the RTO imposed on the entities by checking that they are truly pairwise compatible. If (1) is not satisfied for some n, m, then an incident on n which causes on n a downtime shorter than $RTO(n)$ would cause on m by propagation a downtime *longer* than $RTO(m)$. In other words, if (1) is not satisfied then one could witness the paradoxical situation that the RTO on m is not satisfied because of an incident on another entity n, while this incident remained within the RTO of n in the first place. RTOs are meant to define a local standard that guarantees a global continuity level; because of this we believe that an RTO not respecting (1) would be of no practical use.

Running example - Part 4 *By applying an algorithm based on Prop. 7, to the TDR model in Fig. 1, we evaluate the* mrt *for each entity w.r.t. the MTPD expressed in the previous example. The graph below reports the original RTO value assigned in the traditional way (i.e. manually) by the IT-BCP group on the IT assets of the TDR model as well as the automatically evaluated* mrt.

Id	RTO	mrt
a_1	6h	8h
a_2	6h	4h
a_3	6h	24h 30'
db_1	5h	8h 10'
db_2	5h	4h 5'
m_1	5h	4h
m_2	7h	8h 10'
m_3	3h	4h 5'
n_1	8h	4h

The traditionally assessed RTO in some cases is too short and in other cases it is too long, i.e. insufficient to ensure the business continuity. By applying Prop. 9, we compare the original RTO w.r.t. the mrt *and find four critical points (outlined by a red circle in the figure below), where the original RTO value exceeds the* mrt.
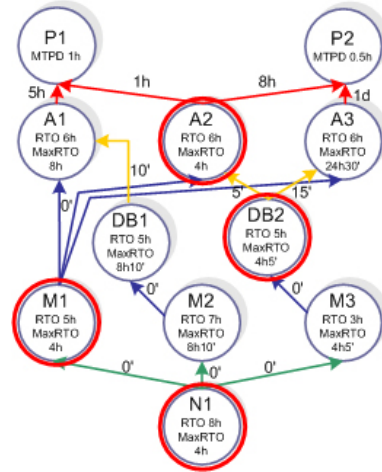
3.2   Exceeding the MTPD

As it is impossible to achieve total security, it is often difficult to comply all the times with the given MTPD. Disasters happen and an it is normal to accept a residual risk that implies that the given MTPD may be exceeded in truly exceptional situations. For instance, there could be some particularly serious incidents that cannot be recovered in time. On the other hand the IT department may be unprepared to handle some disruptive events due to lack of personnel or resources. To deal with that, two ways are possible: (1) the organization's management decides to employ more

resources and deploy new controls allowing to shorten the disruption time, or (2) the risk of exceeding the MTPD is accepted within a given probability.

However, to be able to accept the risk of exceeding the MTPD, an organization needs to have a reasonable estimate of *how often* this is going to happen in a given time frame (which could be one year or ten years, for instance).

To this end, we can use our TDR model. We know from Prop. 7 that every time an incident occurring on an entity is not repaired within its *mrt* one or more business processes depending on the entity will become unavailable for longer than their MTPD.

Therefore, to evaluate the frequency a business process p exceeds its MTPD we need to know how many occurrences of the incidents, affecting the entities on which p (directly or indirectly) depends on, exceed their *mrt*. To this end we use the recovery time distribution that is evaluated during the risk assessment phase.

Definition 10 *Let $n \in N$ be an entity, $p \in P$ be a business process and $mt = MTPD(p)$ be its MTPD. If I is the set of incidents affecting n, the frequency n exceeds its mrt because of incidents I is:*

$$\Phi(n, I) = \sum_{i \in I} freq(i) \times \sum_{t > mrt_{mt}(n)} rt(n, i, t)$$

Intuitively, $\Phi(n, I)$ expresses the number of times an entity n exceeds its *mrt* because of a set of incidents I. The following proposition expresses how we use Φ to evaluate the how many times is exceeded the MTPD for a given business process p, $Fex(p)$.

Proposition 11 *Let p be a business process and E_p be the set of entities on which p depends on, directly or by propagation. If $\forall m \in E_p$, I_m is the set of all the incident that can occur on m then:*

$$(2) \qquad Fex(p) = \sum_{m \in E_p} \Phi(m, I_m)$$

In other words, the frequency a process exceeds its MTPD is determined by the sum of the frequencies the entities it depends on exceed their *mrt*. With such an information the business unit is able to verify if the residual risk it is willing to accept is not further exceeded by the IT department. Such a condition would require the development of more effective strategies to reduce the recovery time to incidents.

Running example - Part 5 *Let us introduce two incidents i_1 and i_2, the first affecting A2, the second affecting DB2. i_1 is estimated to happen five times a year and it is repaired within three hours in the 80% of the cases and within eight hours in the remaining 20%. i_2 is estimated to happen seven times a year and it is repaired within four hours in the 90% of the cases and within six hours in the remaining 10%. If we consider the MTPD of P1 (three hours), then the mrt is 4h for A2 and 4h5' for DB2. The frequency DB2 exceeds its mrt is 0.7 times a year while the frequency A2 exceeds its mrt is 1 time a year. Consequently, assuming the only incidents affecting the entities in the TDR model are i_1 and i_2, our tool allows us to comute that P1 is expected to exceed its MTPD 1.7 times a year (once a year by 3h equivalent to the 200% of the MTPD and 0.7 times a year by 1h55' equivalent to the 164% of the MTPD).*

4 The Practice

Our experience on BCP auditing is based on the general approach used by KPMG Italy. Regarding the BIA, the procedure that is commonly used within organizations to establish the MTPD for the business processes is based on the qualitative analysis of the impact as perceived by the process owner (business unit), of the consequences of a disruption on the process itself. On the other hand, regarding RTOs, only certain entities (most of the applications) are taken into consideration and are labeled with a RTO, since it is difficult to properly evaluate the relationships between the different entities manually.

The first important contribution of our model is that all the relationships are properly evaluated thus enabling the IT department to extract the RTO values for each involved entity (even machines, networks and infrastructures).

We implemented our model with a tool designed to be an additional component of KARISMA (which is the tool developed at KPMG to support Risk Assessment, see Sec. 5): this enables us to repeat and validate the assessment previously done by the KPMG auditing team. The tool is based on the TDR model as an annotated graph by representing each entry with a node and each link with an edge between nodes, annotated with the survival time. It is provided with algorithms directly derived from the framework proposed in Sec. 3.

We tested our model on a KARISMA database of an Italian primary insurance company. This data was collected during an auditing activity carried out by KPMG, and contains information regarding the TDR model (19 macro business processes and 122 sub-processes) and the results of the BIA analysis providing the MTPD value for each sub-process. The remaining information required by our model (about incidents,
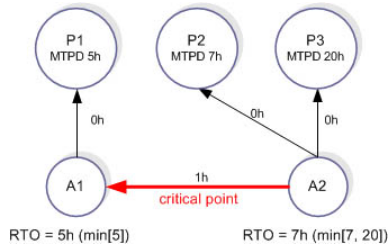
Figure 2. An example of critical point found with our tool. The misleading evaluation of A2's RTO to 7h is caused by the lack in considering the relationship of A2 with A1. If A2 is repaired in more than 6h, process P1 will be disrupted for more than its MTPD.

repair time and frequencies) was also provided by the KPMG auditing team who conducted the assessment.

We achieved two results: first we were able to assign an RTO to all the entities of the TDR model by using the maximum recovery time automatically calculated by the tool. This allows the IT department to assess its ability to comply with its RTOs and, if necessary, arrange with the business unit an exceeding rate. Second, we found some critical points in the previous setting of RTO for the applications. In case of applications supporting more than one business process and other applications, we found that the RTO was in some cases either very close to the mrt (in which case it becomes critical) or it was underestimated (longer than required) because, as we mentioned before, the relationships between applications were not taken into consideration properly (see Fig. 2).

Summarizing, our tool allows one to automatically perform two different assessments: firstly to set properly RTO values for a given business process. Secondly, to support auditors during the BCP validation. Once a BCP has been established and put in place, the validation phase occurs to ensure that the plan is adequate, complete and appropriate w.r.t. the organization's information system [14]. A crucial point is based on the auditing of recovery controls: the auditor must verify that RTO values meet the business requirements. Our tool supports this kind of verification, since every check is made in an automatic way, possibly discovering weaknesses in the BCP.

5 Discussion

In this section we argue the feasibility of our approach and its usefulness to support *continuous auditing of the BCP*.

Feasibility and Validation The main concern regarding the feasibility of our approach is whether the required set of data 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 [7] standard (and is also after RAs). Secondly, an inventory of possible incidents, together with their frequency *has* to be compiled during the RA. Finally, a BCP should provide (accordingly to BS25999 standard) a complete evaluation of the effectiveness of chosen incident response strategies.

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 [11] (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 TDR model is also available when applying to an organization an architectural framework, such as TOGAF [16], Zachman [17] and ArchiMate [1]. 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 projects 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.

Continuous Auditing Finally, we argue that our framework is particularly useful to support a *continuous* auditing process. The concept of continuous auditing is well-known among the risk management

strategies, particularly in the field of software engineering [13]. The goal of this process is continuously assessing what could go wrong in projects (i.e., what the risks are), determining which of these risks are most relevant, and implementing strategies to deal with them. Even though many of the methodologies for risk management [5, 3, 8], as well as those for BC [7], include a monitoring and reviewing step, this process can be performed with different degrees of granularity, according on how flexible the methodology is. For example, a change on the IT infrastructure, involving the dismantling of a set of applications and machines and the introduction of new software and hardware components, may involve either the assessment of the new components only or of the whole organization, depending on how much it is possible to reuse the previous assessment results.

Thanks to the fine granularity and the high degree of independence of the used information (time dependencies, assessment of incidents, importance of processes to the business), our model and tool are particularly suitable to support a continuous assessment process.

For instance, when dealing with a change in the organization, be it the rearrangement of the IT infrastructure or a new business strategy, after a simple update of the model the framework can be used to evaluate the new time constraints within which incidents must be repaired to preserve the business continuity. In the case a new component is added to the information system, it is only necessary to add the new component in the TDR model and specify its functional and temporal relationships with the other components to evaluate its new RTO. On the other hand, if (due to changes in business strategy) a process becomes more important for the organization's business, it is possible to change its MTPD and automatically assess the IT infrastructure to verify if it is still able to ensure the new time constraints.

In addition, after the occurrence of an incident, our model allows to verify if the incident response propagation is compliant to the expected behaviour. It might happen that a time dependency between two applications, that was estimated to be of one hour, is in fact of one hour and a half. Furthermore, it might be observed that the response time to an incident exceeds the forecasted RTO. In those cases, the model can be easily updated with the new collected information, thereby allowing to rapidly assess the new situation and develop new and more efficient BC strategies, if needed. This feature adds quality to our solution since it enables the BC team to organically capitalize on practical experience to improve accuracy of the model and of the outcome in time.

In this perspective, the ability to easily refine the model helps at improving the way organizations traditionally deal with incidents. Instead of simply solving the problem when it happens and then forgetting about it, our solution promotes the continuous monitoring of the performances of the repair operations by collecting new information as incidents occur and then use them to improve the efficiency of the response on new occurrences.

Summarizing, our system allows one to (a) easily adjust the model to changes in the organization and/or its business target, without the need of a complete new assessment, and (b) refine the model (i.e., make it more precise) in the moment that new more accurate information is available about the actual behaviour of the organization.

6 Related Work

Although the Business Continuity Planning process is well described in a number of works [12, 9] and recently has been standardized by the British Standard Institute [7], formal models to support it are still understudied. Despite this situation there are some specific fields for which specific tools have been developed to accurately evaluate the survivability of IT systems. This is the case of telecommunication networks, where the high availability of the network must be ensured through a proper BCP. Jrad et al. [10] propose a BCP model devised to determine the expected downtime due to disaster events as well as normal and software failures in a networked environment and especially tailored for the IT infrastructure of telecommunication networks. The model can be used to predict the probability that a disaster will cause a service disruption. Even though their approach may be extended to all IT networked infrastructures, we believe that it does not properly evaluate the dependencies between the constituents of the IT infrastructure, and in particular the survival time between them. Furthermore our model is explicitly designed to assess the current RTOs and determine critical points, whatever the technique is used to determine incidents likelihood and response time.

Another approach to evaluate the survivability of a system is the one proposed by Cloth and Haverkort in [4]. They describe the system under assessment 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. The scope of their approach (1) is limited to a particular distributed environment, where a much more fine grained

description of the system is considered and (2) only the recovery time is the desired output. This is a very appreciated requirement when dealing with dependability issues in system design, but is not suitable for large infrastructures considered in BCP. On the other hand, our approach is devised to assess the compatibility with the business unit requirements with those ensured by the IT department.

In addition to academic work there exist a number of commercial tools supporting BCP. The most closely related to our work is Shadow Planner [15]. It is an (industrial) application developed to support organizations in assessing risks and establish a BCP. The software has several modules to map the organizations's IT infrastructure, collect BIA information, asset values and etc. Thus, it is able to evaluate the monetary impact of a certain incident. Differently from our approach, it is not based on a model and the relationships between different entities are not properly evaluated. This could hardly affect the way a disruption event is evaluated, resulting in an erroneous planning of countermeasures to ensure business process MTPDs (and the related RTOs).

References

[1] The ArchiMate project. http://archimate.telin.nl.

[2] Basel II: Revised international capital framework, 2005. http://www.bis.org/publ/bcbsca.htm.

[3] P. Bowen, J. Hash, and M. Wilson. Information Security Handbook: A Guide for Managers. Technical report, NIST, 2006. SP 800-100.

[4] 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.

[5] CobiT: Control Objectives for Information and related Technology. http://www.isaca.org.

[6] R. Cocchiara. Beyond disaster recovery: becoming a resilient business. Technical report, IBM, 2005. http://ibm.com/services/its/resilience.

[7] British Standards Institute. Business continuity management - Part1: Code of practice. Technical Report 25999-1, BSI, 2006.

[8] ISO/IEC 27001:2006 Information Security - Specifications, 2005. http://www.iso.org.

[9] ITIL: IT Infrastructure Library. http://www.itil.co.uk/.

[10] A. M. Jrad, C. K. Chan, and T. B. Morawski. Incorporating the downtime due to disaster events in the network reliability model. In *NETWORKS 2004: 11th International Telecommunications Network Strategy and Planning Symposium*, pages 365–371. IEEE Computer Society, 2004.

[11] http://www.kpmg.com.

[12] W. Lam. Ensuring business continuity. *IT Professional*, 4(3):19–25, 2002.

[13] 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.

[14] S. A. Sayana. Auditing Business Continuity. *Information Systems Control Journal*, 1, 2005. http://www.isaca.org/TemplateRedirect.cfm?/template=/ContentManagement/ContentDisplay.cfm&ContentID=23553.

[15] Shadow-Planner, Business Continuity Management software. http://www.office-shadow.com/.

[16] The Open Group. TOGAF (The Open Group Architecture Framework), 2003. http://www.opengroup.org/architecture/togaf8-doc/arch/.

[17] The Zachman Institute for Framework Advancement. Zachman Framework, 2007. http://www.zifa.com/.

[18] E. Zambon, D. Bolzoni, S. Etalle, and M. Salvato. Model-based Mitigation of Availability Risks. Technical Report TR-CTIT-07-04, University of Twente, 2007. http://eprints.eemcs.utwente.nl/9449/.