Softw Syst Model
DOI 10.1007/s10270-010-0166-8

REGULAR PAPER

Model-based qualitative risk assessment for availability of IT infrastructures

Emmanuele Zambon · Sandro Etalle · Roel J. Wieringa · Pieter Hartel

Received: 29 June 2009 / Revised: 6 May 2010 / Accepted: 6 June 2010

Abstract For today's organisations, having a reliable information system is crucial to safeguard enterprise revenues (think of on-line banking, reservations for e-tickets etc.). Such a system must often offer high guarantees in terms of its availability; in other words, to guarantee business continuity, IT systems can afford very little downtime. Unfortunately, making an assessment of IT availability risks is difficult: incidents affecting the availability of a marginal component of the system may propagate in unexpected ways to other more essential components that functionally depend on them. General-purpose risk assessment (RA) methods do not provide technical solutions to deal with this problem. In this paper we present the qualitative time dependency (QualTD) model and technique, which is meant to be employed together with standard RA methods for the qualitative assessment of availability risks based on the propagation of availability incidents in an IT architecture. The QualTD model is based on our previous quantitative time dependency (TD) model (Zambon et al. in BDIM '07: Second IEEE/IFIP international workshop on business-driven IT management. IEEE Computer Society Press, pp 75–83, 2007), but provides more flexible modelling capabilities for the target of assessment. Furthermore, the previous model required quantitative data which is often too costly to acquire, whereas QualTD applies only qualitative scales, making it more applicable to industrial practice. We validate our model and technique in a real-world case by performing a risk assessment on the authentication and authorisation system of a large multinational company and by evaluating the results with respect to the goals of the stakeholders of the system. We also perform a review of the most popular standard RA methods and discuss which type of method can be combined with our technique.

Keywords Information risk management · Risk assessment · Availability · Information security · System modelling

1 Introduction

Among the three main security properties of information, confidentiality, integrity and availability (CIA), the importance of *availability* (defined as: ensuring that authorised users have access to information and associated assets when required [18]) grows as organisations are increasingly dependent on their information systems (on-line banking, reservations for e-tickets etc.) [23]. As a consequence of this, disruptions in the IT infrastructure often leads to monetary loss. This fact is confirmed by the increasing importance that service level agreements (SLAs) are gaining: 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 organisations but also among units of the same company. It is therefore not

Communicated by Prof. Ketil Stølen.

E. Zambon (✉) · S. Etalle · R. J. Wieringa · P. Hartel
University of Twente, Enschede, The Netherlands
e-mail: emmanuele.zambon@utwente.nl

S. Etalle
e-mail: sandro.etalle@utwente.nl

R. J. Wieringa
e-mail: r.j.wieringa@utwente.nl

P. Hartel
e-mail: pieter.hartel@utwente.nl

S. Etalle
Technical University of Eindhoven, Eindhoven, The Netherlands
e-mail: s.etalle@tue.nl

Published online: 22 June 2010

surprising that the mitigation of availability risks (i.e., the risks that affect the availability of the target of assessment) receives attention from the business and the research communities [34,41,44].

In general, IT risk assessment (RA) is the first and most critical phase of IT risk management (RM). During the RA one determines risks related to (recognised) threats and vulnerabilities. Proper RA also ensures that IT risks in the organisation are audited and dealt with in a structured and transparent way.

Among the security risks an organisation faces, availability risks are often particularly important and difficult to assess with ordinary techniques. For instance, incidents affecting the availability of a marginal component of the system may propagate in unexpected ways to other more essential components that functionally depend on them. These dependencies among components are essential to assess availability risks and are difficult to consider without the use of dedicated techniques. Unfortunately, general techniques that could be used to assess availability risks (e.g., FTA or Attack Graphs) are too expensive in terms of time and required resources [22] to be adopted in most RAs. In fact, IT RAs are often carried out based on the intuition and expertise of the auditor and give little guarantee in terms of objectivity and replicability of the results.

This is the general problem we tackle in this paper: defining a technique for assessing availability risks which is simple enough to be included in a real RA, while at the same time providing solid guarantees in terms of accuracy and replicability of the results it delivers.

The concrete problem that leads to the definition of the above general problem statement regards a large multinational company and the method the company uses to assess availability risks. While it is satisfied with the fact that using the present RA method they can perform RAs in time, the company aims at improving their RAs by assessing risks more precisely and reducing the dependency of the results on the personnel carrying out the RA (i.e., when determining the impact level of a threat). At the same time, the company wants to keep the method feasible in terms of both the amount and the detail level of the information required and of the time and resources needed to carry out an RA. In other words, any improvement of their current RA method and techniques should not require information that the team carrying out the RA cannot obtain and should ensure that the results of the RA can still be delivered on time to the requester. The natural choice to achieve these goals is to decompose the risk into its constituting factors such that the following two requirements are met:

(a) the decomposition is objective, i.e., has a true relationship with the complex risk to be assessed;
(b) data can be collected cost-effectively.

To solve this problem, in this paper we introduce the qualitative time dependency (QualTD) model and the technique associated with it. The QualTD model and technique allow one to carry out a qualitative assessment of availability risks based on the propagation of availability incidents in an IT architecture. Incident propagation is used to increase the accuracy of incident impact estimation. Likelihood estimation is not specifically addressed by our technique, but can be based on existing likelihood estimation models (see Sect. 2 for details).

To model the assessed system, we use a graph in which system components are represented by nodes, and the functional dependencies (along with time constraints) are represented by edges between nodes. Dependencies are derived from the IT system architecture. Most of the information can be found in the system specification/development documentation, which keeps the extra work required to use this technique to an acceptable minimum.

In order to evaluate the technique based on the QualTD model we

1. carry out an assessment of the availability risks on the global identity and authentication management system of the company (an availability-critical system) by following the company RA method together with the QualTD model to assess the impact of the threats and vulnerabilities present in the system, from now on we call this assessment RA_2;
2. compare the results on the impact estimation obtained from RA_2 with the results produced during a previous assessment carried out by the company using their internal RA method only, from now on RA_1 (to this end, we used the likelihood estimates from RA_1 to ensure that the results of the two RAs could be comparable);
3. identify some general factors that justify the adoption of our technique also in other cases based on the results of point (2);
4. indicate how to generalise the approach we followed in the present case to other assessments, carried out following other popular (standard) RA/RM methods; and
5. provide a brief review of other RA techniques based on dependency graphs which we found in the literature, and we discuss the results they deliver and their applicability to the present RA case.

Our results indicate that

1. the technique using the QualTD model satisfies requirement (b), i.e., it is feasible to embed the QualTD model with the company's RA method without requiring too much time or unavailable information;
2. the QualTD model constitutes an improvement towards requirement (a), i.e., according to the RA team of the

company, the technique using the QualTD model delivers better results in terms of accuracy (due to a more accurate impact estimation) and helps delivering more inter-subjective results (i.e., less dependent on the personnel carrying out the RA);

3. other RA techniques based on dependency graphs [2,10,13,20] do not satisfy requirement (b), i.e., they could not be applied to the present case, due to the fact that they require information that is unavailable or that requires too much time to be extracted.
4. the QualTD model can be used in combination with other existing standard methods, under some conditions, which include (1) the information available is enough to apply the QualTD model and (2) the compatibility of the target method with some key features of the QualTD model.

The last point deserves an additional explanation. A concern one has when introducing a new technique for assessing specific risks is whether this technique fits within more high-level RA methods. Intuitively speaking, a general (say company-wide) RA is usually carried out following a (high-level) *method* and a number of specific *techniques*. The high-level method specifies the global lines to set up the RA process and to embed it into the organisation. Examples of RA methods include CRAMM [36], IT-Grundshutz [40], OCTAVE [32] or the NIST SP 800-30 [27]; a more complete list can be found in Sect. 5.1. The RA method usually includes a number of tasks (like evaluating the availability risks), and does not fully specify how to implement them within a specific organisation. This gives organisations the flexibility of choosing an appropriate *technique*. Techniques include Fault and Event Tree Analysis [28], Attack Graphs [26] or HazOP [6]. Our contribution can be seen as a *technique* to assess availability risks. To establish to which extent the QualTD model can be embedded in present popular RA methods, we have made a taxonomy of them and pointed out the conditions that need to be satisfied for this embedding to be successful.

In our previous work we presented the time dependency (TD) model [31], which is meant to be used to analyse and evaluate availability risks in a quantitative way. Differently from qualitative risk assessments, in quantitative risk assessments likelihoods are represented by numeric frequencies or probabilities and impact is numeric and represents, e.g., money. The magnitude of the difference between risk values is therefore known. The TD model is based on the same general principle the QualTD model is based on, i.e., using the architecture to model time constraints and the functional dependencies among the components of the IT system. If provided with the expected frequency and the monetary loss associated with the system component disruption, the TD model allows one to calculate the risk associated with an incident as the average loss expectancy, and to rank incidents accordingly. Also, based on the incident average loss expectancy and on the cost of available mitigating controls, it allows one to select the subset of the available controls that mitigate the loss due to availability incidents at the lowest possible price. However, the TD model has a number of limitations that make it unsuitable to be used as a general-purpose RA technique: (1) being quantitative it requires information that in many industrial organisations can be difficult to obtain (e.g., incident probability or expected frequency): this makes it not compliant with requirement (b), (2) it assumes that if one component fails, all the components depending on it will automatically fail, which may not be the case when redundancy measures are in place, and (3) it assumes the incidents that can affect the target of assessment is already known by the risk assessor.

The QualTD model is more geared to industrial practice than the TD model, since it is fully qualitative and does not require information which is hard to gather: therefore, it overcomes limitation (1) of our previous TD model. To make the QualTD model qualitative we determine the impact and risk of availability incidents when the estimates about the likelihood of threats and vulnerabilities, the incident duration and the importance of the business functions supported by the analysed IT system are expressed by values in an ordinal scale. In an ordinal scale, only ordering among values are known (e.g., $High > Medium > Low$). The QualTD model also solves limitation (2) by introducing *AND* /*OR* dependencies to specify with more flexibility the behaviour of a component on the components it depends on when they fail. To solve limitation (3), we provide in this paper a framework to link threats and vulnerabilities with the components of the IT system under assessment and derive a list of incidents, thus increasing the usability of our model and technique.

This paper is structured as follows: in Sect. 2 we formally present the QualTD model and we explain how it works by means of a running example. In Sect. 3 we first introduce the industrial context in which we tested the QualTD model and then we present the technique we used to apply the QualTD model to this industrial case. In Sect. 4 we describe the design, the criteria and the assumptions we made to evaluate the QualTD model and technique, and we present the evaluation results. In Sect. 5 we first discuss the applicability of our technique in combination with standard RA/RM methods, and then we compare our technique with other dependency-based RA techniques in the literature. Finally, in Sect. 6 we draw the conclusions of the paper.

2 The qualitative time dependency (QualTD) model

We now introduce the model supporting our RA technique. To illustrate the ideas we provide a running example showing how the QualTD model can be employed in practice.

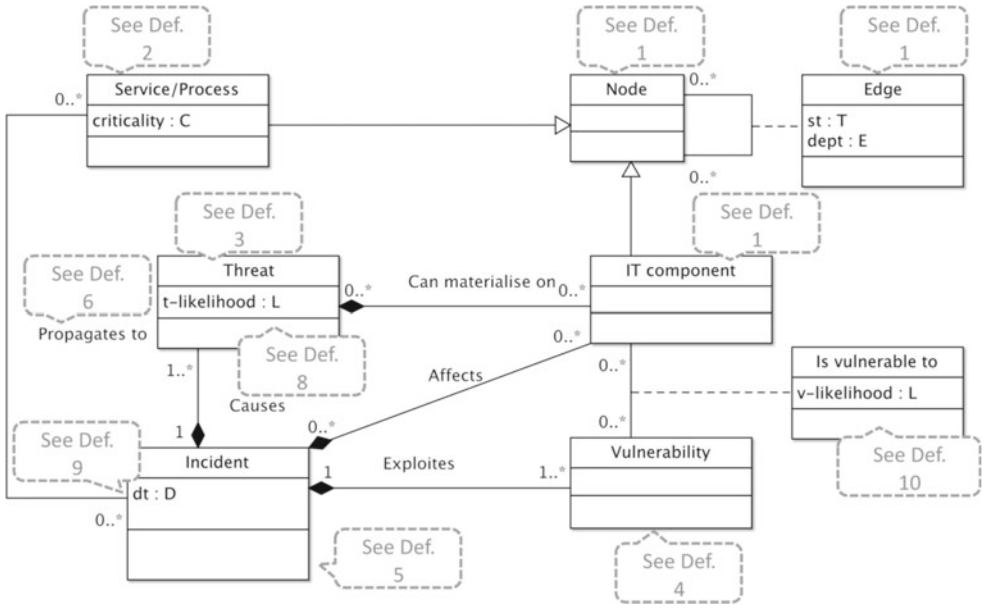

Fig. 1 UML class diagram of the QualTD model. In the diagram, the type name of the attributes (criticality, likelihood, downtime, survival time, dependency type) is referred to by their initial letter only

The QualTD model represents the system target of the assessment (ToA) by means of a graph in which nodes can be system components, services or processes supported by the system, and dependencies among nodes are the edges of the graph. Incidents that can affect the ToA are the results of a combination of threats and vulnerabilities and affect one or more nodes in the graph. So, for example, a threat can be a denial of service, a vulnerability can be a buffer overflow and an incident a denial of service on a specific application carried out by exploiting the buffer overflow vulnerability. The effects of an incident can propagate to another system component, service or process following the dependencies in the ToA. The model allows us to compute the global impact and the risk levels of the availability incidents hitting the ToA in the way we are about to explain. Figure 1 summarises the main concepts of the QualTD model: for each one of them we will provide a more detailed description in the sequel. Nodes and edges are the constituents of a dependency graph. In turn, a node represents an asset constituting the IT architecture, and it is modelled as a generalisation of IT components (e.g., network components, servers, applications) and IT services or processes, which can have a certain criticality for the organisation's business. Threats can materialise on IT components (with a certain likelihood). IT components can (with a certain likelihood) have vulnerabilities. Our definitions of threats and vulnerabilities are similar to the ones given in BS 7799-3 [5]. A combination of a threat and a vulnerability on a specific set of IT components constitutes a security event (see BS 7799-3), which we call incident, and can have a certain duration. Note that BS 7799-3 defines an incident as a security event with good probability of damaging the organisation's business. According to this definition, an incident would be a combination of a threat and a vulnerability on a specific set of IT components which have a *good* likelihood and impact. For the sake of the presentation, we do not report in the diagram the concepts of incident harm and risk, as well as incident risk aggregated by threat/vulnerability, as they are complex concepts which are produced as the output of the model.

We split the presentation of the model according to the three phases of an RA the model supports: (1) definition of the ToA, (2) risk identification and (3) risk evaluation. To simplify the exposition we use the following sets to indicate domains: $\mathbb{T}$ is the set of all the time interval lengths (expressed in minutes), $\mathbb{E}$ is the set of all the possible dependency (edge) types and it is defined as $\mathbb{E} = \{AND, OR\}$, $\mathbb{D}$ is the set of all the qualitative values expressing duration (e.g., `Short`, `Long`), $\mathbb{L}$ is the set of all the qualitative values expressing likelihood (e.g., `Likely`, `Unlikely`), $\mathbb{C}$ is the set of all the qualitative values expressing business value/criticality of an asset (e.g., `Critical`, `Unimportant`), $\mathbb{H}$ is the set of all the qualitative values expressing business harm (e.g., `Severe`, `Negligible`) and $\mathbb{R}$ is the set of all the qualitative values expressing the risk (e.g., `High`, `Low`).

2.1 Definition of the ToA

We model the ToA by means of an *AND*/*OR* graph which represents the components of the ToA and their functional/technical and organisational dependencies.

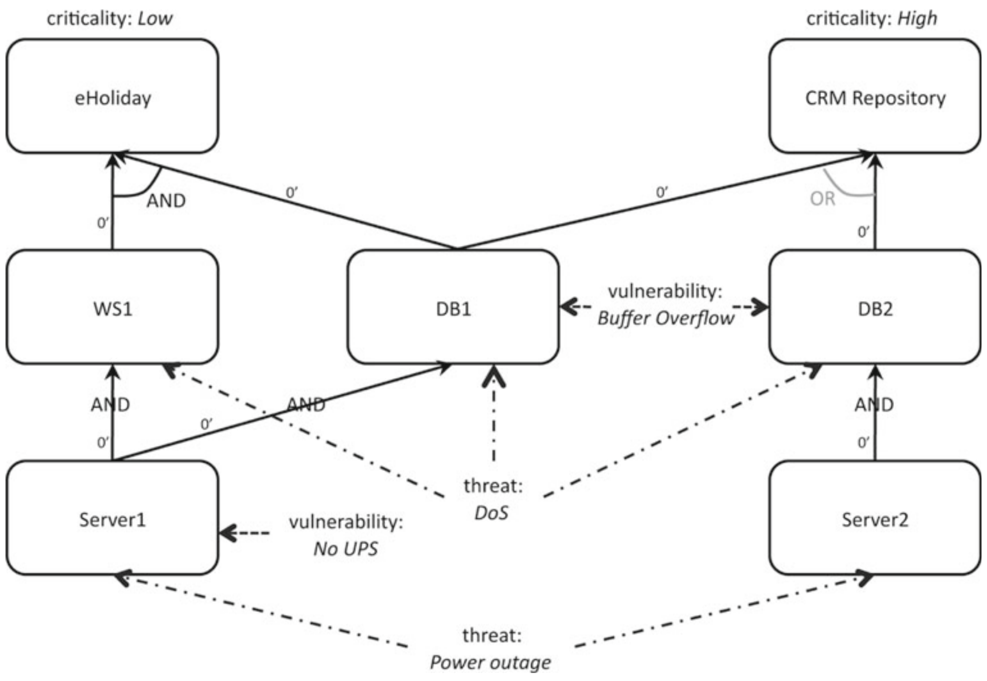

Fig. 2 The dependency graph representing the ToA in our running example. Nodes are the constituents of the (partial) IT infrastructure under exam. Services are annotated with their criticality level for the organisation. The figure also includes the vulnerabilities and threats which we will formally introduce later in this section and specifically describe in the running example

Definition 1 (*Dependency graph*) A dependency graph is a pair $\langle N, E\rangle$ where N is a set of nodes representing the constituents of the ToA, and E is a set of edges between nodes $E \subseteq \{\langle u, v, dept, st\rangle \mid u, v \in N, dept \in \mathbb{E}$ and $st \in \mathbb{T}\}$.

Running example: Part 1 The ToA in this example is the portion of the IT infrastructure of an organisation providing two IT services: `eHoliday`, the holiday reservation service for the employees of the organisation and `CRM-Repository`, the organisations Customer Relationship Management (CRM) repository service. These services are implemented by means of three applications: `WS1`, a web server, `DB1` and `DB2`, two databases. `DB1` and `DB2` contain replicas of the CRM data, but only `DB1` is used by `WS1` as a repository for `eHoliday`. Applications are running on two different servers: `Server1` and `Server2`. `eHoliday` is implemented by `WS1` and `DB1` and, if only one of them is off-line, the service will be off-line as well. `CRM-Repository` is implemented by `DB1` and `DB2`, but both applications must be off-line for the service to be unavailable. `WS1` and `DB1` run on `Server1`, while `DB2` runs on `Server2`. According to this description, we build the dependency graph $g = \langle N, E\rangle$ as follows:
$N = \{$`eHoliday`, `CRM-Repository`, `WS1`, `DB1`, `DB2`, `Server1`, `Server2`$\}$, and
$E = \{ \langle$`Server1`, `WS1`, $AND, 0\rangle$, $\langle$`Server1`, `DB1`, $AND, 0\rangle$, $\langle$`Server2`, `DB2`, $AND, 0\rangle$, $\langle$`WS1`, `eHoliday`, $AND, 0\rangle$, $\langle$`DB1`, `eHoliday`, $AND, 0\rangle$, $\langle$`DB1`, `CRM-Repository`, $OR, 0\rangle$, $\langle$`DB2`, `CRM-Repository`, $OR, 0\rangle \}$.
Figure 2 shows the dependency graph of this running example.

The nodes N of the graph are the constituents of an IT architecture, e.g., IT services, applications, servers, network components and locations, together with the business processes the IT supports. Different IT components can be represented by means of a single node in the graph, according to the abstraction level required by the RA. For example, in a company-wide assessment we could represent an IT service (i.e., a set of servers and all the applications running on them) by means of a single node, while for the assessment of a specific IT system we model each component as an individual node.

An edge from node b to node a indicates that a depends on b. The graph supports both *AND* and *OR* dependencies. In the former case this means that a becomes unavailable when any node it depends on is disrupted. In the latter case, a becomes unavailable when all nodes it depends on are disrupted. Each edge is also annotated with the survival time (st), which indicates the amount of time v can continue to operate after u is disrupted.

If a node a has an *AND* dependency on nodes b and c and an *OR* dependency on nodes d and e at the same time, we read this as a having an *AND* dependency on nodes b, c and x, with x having an *OR* dependency on nodes d and e. Similarly, the survival time of node a with respect to nodes

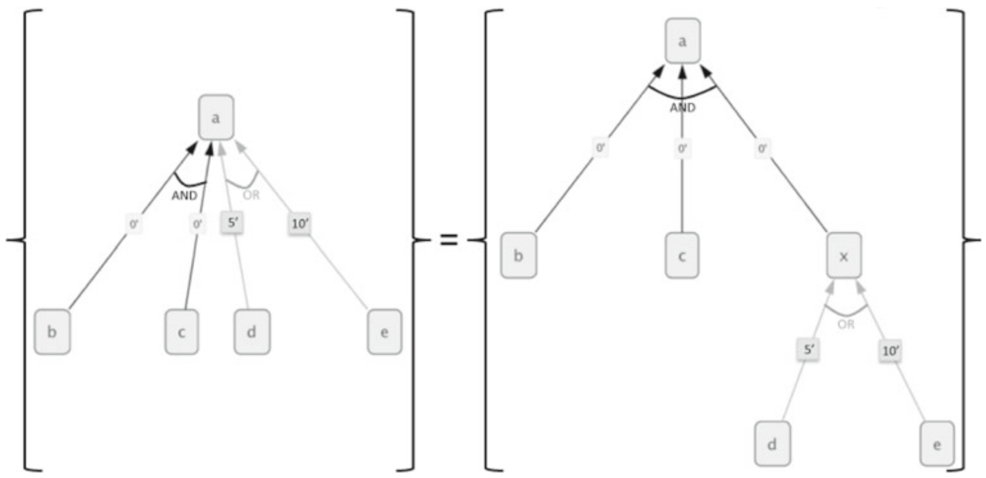

Fig. 3 Equivalence of a graph with mixed *AND* and *OR* dependencies

d and *e* becomes the survival time of node *x* with respect to *d* and *e*, and the survival time of node *a* with respect to *x* is set to zero. This concept is shown in Fig. 3.

To complete the description of the ToA we include in the model an estimate of the criticality of the business processes and of the IT services in the perspective of the RA requester.

Definition 2 (*Process/Service criticality*) Given a dependency graph $g = \langle N, E \rangle$, the criticality of a process/service is a mapping *criticality* : $N \rightarrow \mathbb{C}$.

Running example: Part 2 According to the business units of the organisation using the IT system, the criticality level of `eHoliday` and `CRM-Repository` is, respectively, `Low` and `High`.

Criticality is defined only for those nodes which represent IT services or business processes. It expresses the damage the company suffers if the node becomes unavailable. For example, in a production company, an IT service supporting a production line, which is a core business function, has a higher criticality than, e.g., personal e-mail for employees.

2.2 Risk identification

After modelling the ToA, we identify the vulnerabilities which are present on it, as well as the threats which could materialise on it, in particular the ones that compromise its availability.

Definition 3 (*Threat*) Given a dependency graph $g = \langle N, E \rangle$, a threat is a potential cause of an incident, that may harm one or more nodes of g. We call T the set of all the threats to the ToA.

Running example: Part 3 For the sake of simplicity, here we identify two threats to the ToA: a `Power outage` can bring the servers off-line and a Denial of Service (`DoS`) attack can cause the unavailability of the applications. Our set of threats is therefore $T = \{$`Power outage`, `DoS`$\}$.

This is a common definition of threat, similar to that given in BS7799-3 [5]; moreover, it is fully compatible with the concept of threat the Company has adopted in its internal RA method. The set of threats T our model addresses are only the ones which have an impact on the availability of the ToA.

Definition 4 (*Vulnerability*) Given a dependency graph $g = \langle N, E \rangle$, and the set of threats T, a vulnerability is a weakness of a node (or group of nodes) in N that can be exploited by one or more threats in T. We call V the set of vulnerabilities on the ToA.

Running example: Part 4 We identify two vulnerabilities which can be present on the nodes of the ToA: `Server1` does not have an Uninterruptible Power Supply (UPS) unit for power continuity in case of outage; moreover, `DB1` and `DB2` may crash after a buffer overflow attack. Our set of vulnerabilities is therefore $V = \{$`No UPS`, `Buffer overflow`$\}$.

Also in this case, our definition of vulnerability is consistent with both the definition given in RA standards, and with the concept of vulnerability the Company has adopted in its internal RA method.

We model an incident as a security event (as defined in BS7799-3 [5]) caused by a specific threat on a particular component of the IT architecture by exploiting a specific vulnerability. Differently from the definition of incident given in BS7799-3, we consider as incidents all security events, not only events “that have a significant probability of compromising business operations”.

Definition 5 (*Incident*) Given a dependency graph $g = \langle N, E\rangle$, a set of threats T and a set of vulnerabilities V, an incident i is a 3-uple $\langle M, t, v\rangle$ with $M \subseteq N$, $t \in T$ and $v \in V$, describing the combination of three events:

1. v is a vulnerability of each node $n \in M$
2. t is the cause of i on each node $n \in M$
3. t exploits v

We call I the set of all incidents generated from g, T and V. Moreover, we say a node n is directly affected by an incident $i = \langle M, t, v\rangle$ if $n \in M$.

Running example: Part 5 By combining g, T and V we identify four incidents that can hit the ToA: (i_1) A power outage causes `Server1` to stop because there is no UPS, (i_2) a DoS attack is performed on `DB1` by exploiting the buffer overflow vulnerability, (i_3) a DoS attack is performed on `DB2` by exploiting the buffer overflow vulnerability, and (i_4) a DoS attack is performed both on `DB1` and `DB2` by exploiting the buffer overflow vulnerability. Our set of incidents is therefore $I = \{i_1, i_2, i_3, i_4\}$ where

$i_1 = \langle\{\texttt{Server1}\}, \texttt{Power outage}, \texttt{No UPS}\rangle$, $i_2 = \langle\{\texttt{DB1}\}, \texttt{DoS}, \texttt{Buffer overflow}\rangle$,
$i_3 = \langle\{\texttt{DB2}\}, \texttt{DoS}, \texttt{Buffer overflow}\rangle$, $i_4 = \langle\{\texttt{DB1}, \texttt{DB2}\}, \texttt{DoS}, \texttt{Buffer overflow}\rangle$.

The last concept we introduce for risk identification is incident propagation.

Definition 6 (*Incident propagation*) Given a dependency graph $g = \langle N, E\rangle$ and an incident $i = \langle M, t, v\rangle$, we say that i can propagate to a node $n \in N$ if

1. $n \in M$, or
2. $\exists e \in E \mid e = \langle m, n, AND, st\rangle$ and i propagates to m, or
3. $\forall e \in E \mid e = \langle m, n, OR, st\rangle$, i propagates to m.

Running example: Part 6 We want to know if the incident $i_1 = \langle\{\texttt{Server1}\}, \texttt{Power outage}, \texttt{No UPS}\rangle$ propagates to `eHoliday`. Although `eHoliday` is not directly affected by the incident, it depends on `WS1` and `DB1`, which in turn depend on `Server1`. `Server1` is directly affected by the incident; therefore, we know that i_1 will propagate to `eHoliday`.

Definition 7 (*Nodes affected by the propagation of an incident*) Given a dependency graph $g = \langle N, E\rangle$ and an incident $i = \langle M, t, v\rangle$, $Prop_i = \{n \in N \mid i \text{ propagates to } n\}$.

Running example: Part 7 According to Definition 7, the set of nodes affected by the incident $i_1 = \langle\{\texttt{Server1}\}, \texttt{Power outage}, \texttt{No UPS}\rangle$ is $Prop_i = \{\texttt{Server1}, \texttt{WS1}, \texttt{DB1}, \texttt{eHoliday}\}$.

Note that, while the definitions of Threat, Vulnerability and Incident we give in this section could be generally used for confidentiality, integrity and availability, the definition of incident propagation is specific to availability. In fact, an availability incident propagates on the IT architecture because of the technical/functional and organisational dependencies that connect the constituents of the architecture. For example, a power outage on a datacentre will result in some servers being unavailable, as well as the applications running on these servers. This disruption causes the IT services depending on the disrupted applications to become unavailable in turn, and propagates from servers to the (key) business processes supported by the IT services. On the other hand, even if a confidentiality or integrity incident may propagate through the IT architecture following the same path, this is not always the case. For example, if the confidentiality of a network is compromised because someone is eavesdropping data that are carried by the network, this does not imply that the confidentiality of the applications using that network (and thus connected to it in the dependency graph) is compromised, as they may use encrypted traffic (and this feature is not represented in the graph). Therefore, our model is specific to availability only.

2.3 Risk evaluation

The last piece of information we include in the model regards likelihood and duration of incidents. In more detail, a threat is characterised by two indicators: (1) the threat likelihood and (2) the time needed to solve the disruption caused by the threat, e.g., a `Short` or `Long` disruption, or even more than two disruption lengths.

Definition 8 (*Threat likelihood*) Given the set of threats T, the threat likelihood is a mapping $t\text{-}likelihood : T \rightarrow \mathbb{L}$.

Running example: Part 8 Security analysts have assigned a likelihood to the threats in T using the following scale: `Very Likely`, `Likely` and `Unlikely`. The likelihood of `Power outage` is `Unlikely` and the likelihood of `DoS` is `Likely`.

The likelihood of a threat is an estimate of the probability of the threat materialising on the ToA. Here, we have made the (simplifying) assumption that the likelihood of a threat is a property of the threat itself, and it is independent from the IT component the threat occurs on. The assumption holds for most of the threats, but not for targeted attacks (i.e., attacks crafted for and directed to a specific IT component), since the likelihood of the attack is influenced by the value of the targeted component. In this case we split the threat into a number of new threats, each of them representing a specific IT component being targeted.

It is common practice in qualitative RAs to assess the likelihood of threats by means of so-called likelihood models. Each model combines different parameters, e.g., difficulty of the attack, resources needed, etc. to determine the final likelihood of a threat. However, it is out of the scope of this work to specify such a model. In the literature there exist works proposing models for specific contexts (e.g., eTVRA [25] for telco networks).

Definition 9 (*Incident duration*) Given a dependency graph $g = \langle N, E\rangle$ and a set of incidents I, the incident duration is a mapping $dt : I \times N \rightarrow \mathbb{D}$.

Running example: Part 9 According to the stakeholders of the IT system, an incident is classified as a `Long` disruption if it takes more than 3 h to be repaired, as a `Short` one otherwise. The contract signed with the power company guarantees that a power disruption is repaired on average in 6 h. Therefore, i_1 is classified as a `Long` disruption. Since restoring `DB1` or `DB2` after they crashed only requires a restart, incidents i_2, i_3 and i_4 are classified as `Short` disruptions.

dt(i,n) is an estimate of the (average) time a node *n* is out of service when incident *i* occurs. If we consider, for example, a buffer overflow attack which causes the stop of an application, the disruption time is the time needed to detect that the application is no longer running and to restart it. We do not take into account the time needed to fix the vulnerability exploited by the threat (e.g., the time to patch the system), unless this activity is needed to restore the functionalities of the system. To keep the model qualitative, and to match the Company method, we apply a discretisation of the disruption time in terms of `short` disruption (i.e., shorter than a given threshold) and `long` disruption (i.e., longer than a given threshold), which constitute our $\mathbb{D}$ set.

We now associate vulnerabilities with their likelihood.

Definition 10 (*Vulnerability likelihood*) Given a dependency graph $g = \langle N, E\rangle$, and the set of vulnerabilities V, the vulnerability likelihood is a mapping *v-likelihood* : $V \times \mathcal{P}(N) \rightarrow \mathbb{L}$, where $\mathcal{P}(N)$ is the power set of N.

Running example: Part 10 Security analysts have assigned a likelihood to the vulnerabilities in V using the following scale: `Very Likely`, `Likely` and `Unlikely`. The likelihood of `No UPS` and `Buffer overflow` is `Very Likely`.

The *v-likelihood(v, N_v)* is an estimate of the probability that the vulnerability v is present in the set of homogeneous nodes N_v, i.e., nodes which can suffer from the same vulnerability with the same likelihood. The simplest and most frequent case is when we determine the likelihood of a vulnerability being present on a single node of g. However, we might also need to consider the likelihood of a vulnerability being present on a set of homogeneous nodes which are involved in a specific incident. For example, consider the case in which some malware causes a number of servers to stop working by exploiting a vulnerability which is present in an application deployed on all of these servers: in this case we need to estimate the likelihood of the vulnerability being present on all of the servers running the application with the vulnerability, since the resulting incident would affect all of them at once.

In case of an accurate RA (e.g., when it is possible to do technical vulnerability verification such as penetration testing), the fact that an application is present on an IT component can be determined without uncertainty; for example by making sure a buffer overflow affects a web server by trying to exploit it. However, in most cases, due to lack of time, the RA team has to rely on indirect (and therefore uncertain) information, for example, by consulting the NIST National Vulnerability Database [43] to check if the web server may suffer from a specific buffer overflow vulnerability. *v-likelihood* is the expression of this uncertainty.

2.4 Output of the QualTD model

We use the information contained in the model to calculate the risk associated with an incident, which is influenced by the likelihood that the threat occurs in the ToA (which is a property of the ToA), the likelihood that a vulnerability is present in a node or a set of nodes (which expresses the uncertainty about whether the vulnerability is present in the nodes) and the estimated disruption severity. In more detail, an incident causes (by propagation) a disruption with a certain duration on some nodes of the dependency graph which have a certain criticality. We call this combination the *global impact* of the incident.

Intuitively, the more critical the processes/services affected and the longer the disruption, the greater the impact of the incident will be, i.e., the global impact of an incident is monotone.

Definition 11 (*Global impact*) Given a dependency graph $g = \langle N, E\rangle$, an incident $i = \langle M, v, t\rangle$, a monotone composition function *harm* : $\mathbb{C} \times \mathbb{D} \rightarrow \mathbb{H}$ mapping criticality and duration to business harm, and a monotone aggregation function *impact-agg* : $\mathbb{H} \times \cdots \times \mathbb{H} \rightarrow \mathbb{H}$; the global impact of i is defined by *global-impact* : $I \rightarrow \mathbb{H}$, such that

$$\text{global-impact}(i) = \text{impact-agg}_{n \in \text{Prop}_i} (\text{harm}(\text{criticality}(n), dt(i, n))) \tag{1}$$

Running example: Part 11 The RA team has decided that the global impact of an incident is calculated using the following rules:

a) the global impact is `Critical` if the incident causes the disruption of at least one service with `High` criticality;
b) the global impact is `Moderate` if the incidents causes a `Long` disruption on any service, or a `Short` disruption of at least a service with `Medium` criticality;
c) the impact is `Insignificant` otherwise.

For example, if we take the above definition (a), the *impact-agg* function is given by the "at least one service" statement, and the *harm* function is given by associating any disruption of a service with `High` criticality to the `Critical` impact. According to these rules the criticality of i_1, i_2, i_3 and i_4 is, respectively, `Moderate`, `Insignificant`, `Insignificant`, `Critical`.

Now that we have defined the incident global impact we can evaluate the incident risk, which is a composition of the likelihood of the threat, the likelihood of the vulnerability and the global impact of the disruption caused by the threat materialising.

Intuitively, this means that the more likely it is that a threat materialises on an IT component (or a set of them), or the more likely it is that the component is vulnerable to that threat, and the more harmful the threat is, the more reasons there will be to protect it against this incident. As for the global impact, also the incident risk is therefore monotone.

Definition 12 (*Incident risk*) Given an incident $i = \langle M, t, v\rangle$, the incident risk is a monotone composition function *i-risk* : $\mathbb{L} \times \mathbb{L} \times \mathbb{H} \rightarrow \mathbb{R}$ mapping *t-likelihood*(t), *v-likelihood*(v) and *global-impact*(i) to the risk level of i.

Running example: Part 12 As for the global impact, the RA team has decided that the risk level of an incident is calculated using the following rules:

(a) the risk level is `High` if either the incident has a `Critical` global impact and at least `Likely` threat and vulnerability likelihood, or if the global impact is `Moderate` and threat and vulnerability likelihood are both `Very Likely`;

(b) the risk level is `Medium` if either the incident has a `Critical` global impact and the threat and vulnerability likelihood are both at most `Likely`, or if the global impact is `Moderate` and either threat or vulnerability likelihood is `Very Likely`;

(c) the risk level is `Low` otherwise.

In this case, *i-risk* is implemented by means of these three rules, which associate the combination of global impact, threat likelihood and vulnerability likelihood to the correspondent risk level. According to these rules, the risk level of i_1, i_2, i_3, and i_4 is respectively: `Medium`, `Low`, `Low` and `High`.

An additional operation one would like to do is to aggregate the incident risk in terms of threats and vulnerabilities. Evaluating risk in terms of threats and vulnerabilities is important to determine both the risk profile of the ToA, i.e., which threat sources are the most harmful, and to prioritise vulnerabilities to be addressed (i.e., patched) first.

Definition 13 (*Incident risk aggregated by Threat/Vulnerability*) Given a dependency graph $g = \langle N, E\rangle$, a threat t and the set of incidents $I_t = \{i \mid i = \langle M_t, t, v_t\rangle\}$, a vulnerability v and the set of incidents $I_v = \{i \mid i = \langle M_v, t_v, v\rangle\}$ and a monotone aggregation function *risk-agg* : $\mathbb{R} \times \cdots \times \mathbb{R} \rightarrow \mathbb{R}$; the risk of a threat t is an aggregation of the risk level of all the possible incidents which can originate from that threat (I_t), i.e., the mapping *t-risk* : $\mathbb{R} \times \cdots \times \mathbb{R} \rightarrow \mathbb{R}$ such that

$$t\text{-}risk(t) = risk\text{-}agg_{i \in I_t}(i\text{-}risk(i)) \tag{2}$$

Similarly, the risk of a vulnerability v is the aggregation of the risk level of all the possible incidents in which that vulnerability has been exploited (I_v), i.e., the mapping *v-risk* : $\mathbb{R} \times \cdots \times \mathbb{R} \rightarrow \mathbb{R}$ such that

$$v\text{-}risk(v) = risk\text{-}agg_{i \in I_v}(i\text{-}risk(i)) \tag{3}$$

Running example: Part 13 If we use *Max* as the aggregation function *risk-agg* to calculate the risk level aggregated by threat/vulnerability, we assign each threat/vulnerability the maximum risk level of the incidents they are involved in. In this way, the risk level of `Power outage` and `DoS` is respectively `Medium` and `High`. Accordingly, the risk level of `No UPS` and `Buffer overflow` is, respectively, `Medium` and `High`.

The QualTD model supports the traceability of the RA results. For instance, suppose the RA has been carried out, and after some time we want to recall why a DoS is a `High` risk for our system, we can go through the records of the model and discover that

1. it is `Likely` that a `DoS` is carried out by exploiting a `Buffer overflow` on both `DB1` and `DB2`,
2. both `DB1` and `DB2` are `Very Likely` to be prone to a `Buffer overflow`
3. the resulting incident causes a `Short` disruption of the `High` critical service `CRM-Repository`,
4. according to points 1–3 and to the impact and risk level definitions, the risk of a `DoS` in the system is `High`.

When doing impact and risk evaluation we use the composition and aggregation functions *harm*, *impact-agg*, *i-risk* and *risk-agg*, which operate with qualitative values (e.g., `High` likelihood and `Low` impact): the definition of the composition and aggregation functions is outside the scope of our model and it is left to the choice of the RA team. However, these functions must be monotone and semantically sound with relation to the meaning that the qualitative values involved have for the stakeholders of the RA. For example, the definition of `Critical` impact we give in the running example part 11 is semantically sound, whereas it would not have been sound if we defined as `Critical` an incident causing a `Short` disruption on a service with `Low` criticality. In the running example and in Sect. 3.2 we describe two possible implementations of *harm*, *impact-agg*, *i-risk* and *risk-agg*, based on descriptive tables which define all the possible combinations of input and output values.

Rationale for a QualTD model It is legitimate to argue whether the model is *sound* or not. It is sound iff disruptions in the model propagate in the same way as in the real system. Regarding soundness, the system we propose has three intrinsic "limitations": (a) it has only *AND* and *OR* nodes, (b) it does not consider the "recovery time" of the single components, and (c) it works only if the graph is acyclic. The first limitation is in our opinion not a problem, as it is simple to model even very complicated dependencies with the use of only *AND* and *OR* nodes. The second limitation is a design choice which keeps the model simple, and in our experience does not affect the fidelity of the model. In any case, it is straightforward to extend our system to also take the individual recovery time into consideration, for example by assigning the recovery time to the nodes and then adding it to the incident downtime during incident propagation. The third limitation is in our opinion the only true limit of the system. Our experience says that acyclic graphs are perfectly suitable to model practical IT architecture. However, it is possible to contrive examples in which this is not the case. For such examples, either one is able to "abstract away" the cycles (for instance by analysing them separately and modelling them with a single node), or our model is simply not applicable. Once one accepts the above three intrinsic limitations, then soundness follows from the soundness of the *AND* and *OR* basic nodes: assuming that (1) the nodes of the dependency graph include all the components of the ToA, and that (2) for every component the availability dependency of this component on other components is correctly and completely included in the graph by means of *AND* /*OR* edges, then the fact that an incident on a certain (set of) components will propagate in the ToA as predicted by the QualTD model can be proved by using standard graph theory. We skip the demonstration for space reasons.

It is the task of the risk assessor using the technique based on the QualTD model to make sure that hypotheses (1) and (2) are reasonably verified in a specific case. In Sect. 3 we will show the technique we used to build the dependency graph as completely and correctly as possible.

3 Case study

In this section we show how the QualTD model can be used in a practical RA by describing the case study we carried out with it. We will also use this case study to evaluate our technique. Let us start by describing the context in which it was carried out.

3.1 The industrial context

The organisation We carried out the case study at a large multinational company with a global presence in over 50 countries (from now on we call it the Company) counting between 100,000 and 200,000 employees. The Company IT unit supports the business of hundreds of internal departments by offering thousands of applications accessed by approximately 100,000 employee workstations and by many hundreds of business partners. The IT facilities for the European branch are located at one site: our RA was conducted at that site. IT services are planned, designed, developed and managed at the Company's headquarters; those services, such as e-mail or ERP systems, are part of the IT infrastructure which is used by all the different Company's branches all over Europe.

The stakeholders of the IT service are (1) the Company's global IT infrastructure (GIT) management department, (2) the risk management and compliance (RMC) department, (3) users: the Company's units using IT services (including GIT and RMC) and (4) an outsourcing company managing parts of the IT infrastructure on behalf of GIT.

GIT provides basic IT infrastructure services such as desktop management, e-mail and identity management. IT services are designed internally by GIT and then partly outsourced for implementation and management to another company. The outsourced tasks include specialised coding, server management, help-desk and problem solving services.

RMC supports the compliance to internal policies and best practices of the Company IT services; part of the tasks of RMC is to perform on-demand security RAs for the IT services of GIT. An RA is usually requested by the owner of the IT service each time a new service is developed or a new release of an existing one is about to be deployed.

The other business units of the Company rely on these IT services for the continuity of their business. Some of these IT services are developed and managed by the business unit itself (e.g., if they are specific to the competence area of the unit), while global company services (e.g., authentication, e-mail system) are provided by GIT. For efficiency reasons, like in most other large organisations, business units exchange services by means of a "enterprise internal market": one business unit pays another one for the use of a given service and the service provider unit finances its

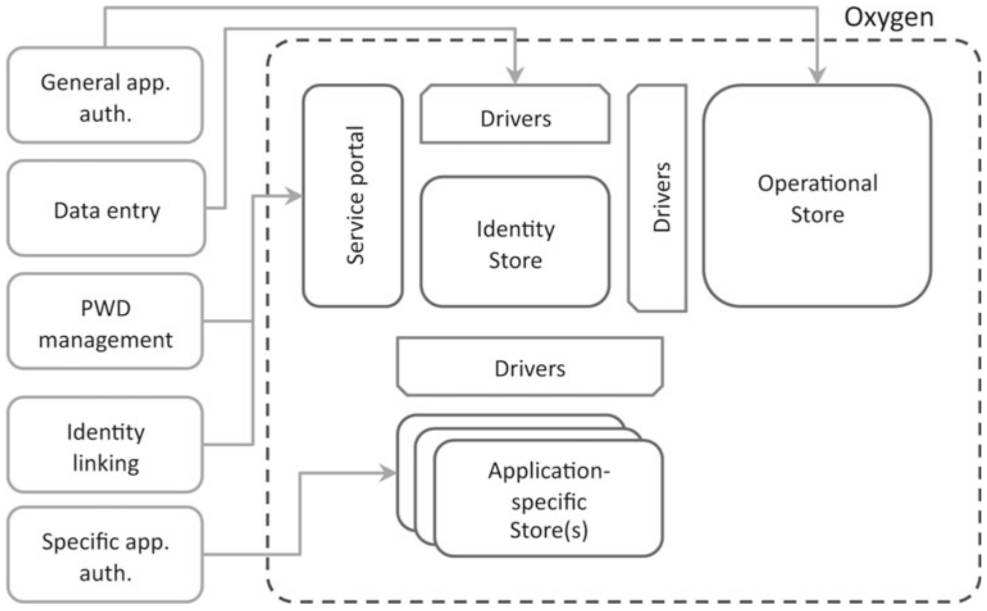

Fig. 4 An overview of the Oxygen architecture

activities by means of these funds. This mechanism increases the efficiency of internal service management.

The implementation and the management of some IT services are outsourced to another company, which we call the Service Provider. Although the servers running the IT services are owned by the Company and physically kept within its data centres, the Service Provider manages the OS and the software running on them. Moreover, for some services, the Company outsources also the development (e.g., coding, deployment) of the custom applications to the Service Provider. The Service Provider has signed contracts with the Company which include service level agreements (SLAs) regarding both the security of the information managed by the outsourcing company and the availability of the outsourced services.

The target of assessment The system on which we focus our case study is called Oxygen. Oxygen is the global Identity Management for employees and sub-contractors of the Company. The goals of the system are

1. *Identity Management*: to provide enterprise-wide standard identities for all employees and contractors of the Company, integrate identities with the different identity authoritative sources (e.g., the Human Resources information system) and manage them through a governed process and ensure regulatory and privacy compliance.
2. *Identity/Account Linking and data synchronisation*: to provide a holistic view of the many accounts possessed by a person, enforce account termination when a person leaves the Company, enable data synchronisation among identity provider and identity consuming systems for data accuracy and provide credential mapping, a foundation for Single Sign-On.
3. *Identity Service for authentication and authorisation*: to provide operational directory services for general applications to be used for authentication and authorisation, to provide unique, standard, organisation-wide identifiers for employees and contractors, and to provide a foundation for advanced authentication and authorisation in the future.

Oxygen is designed and implemented by the GIT department, while the management of the servers running it is outsourced to the Service Provider.

Figure 4 depicts the design of Oxygen: the system is composed of a number of *identity stores*, which are identity databases implemented by means of directory services. The main Identity Store keeps information about all of the identities and their attributes. The Operational and the Application-specific stores contain a (partial) replica of this information and are accessed by the different applications which require identities for authentication and identification. Replication of the identity stores is required for performance reasons.

Oxygen collects identity data from different authoritative sources, such as the information system of the Human Resources department. Data acquisition is performed by means of drivers, which also take care of synchronising data between the different identity stores.

In addition to the identity stores, Oxygen exports also a service portal, which allows employees of the Company to manage part of their identity record (e.g., updating their home address, changing password).

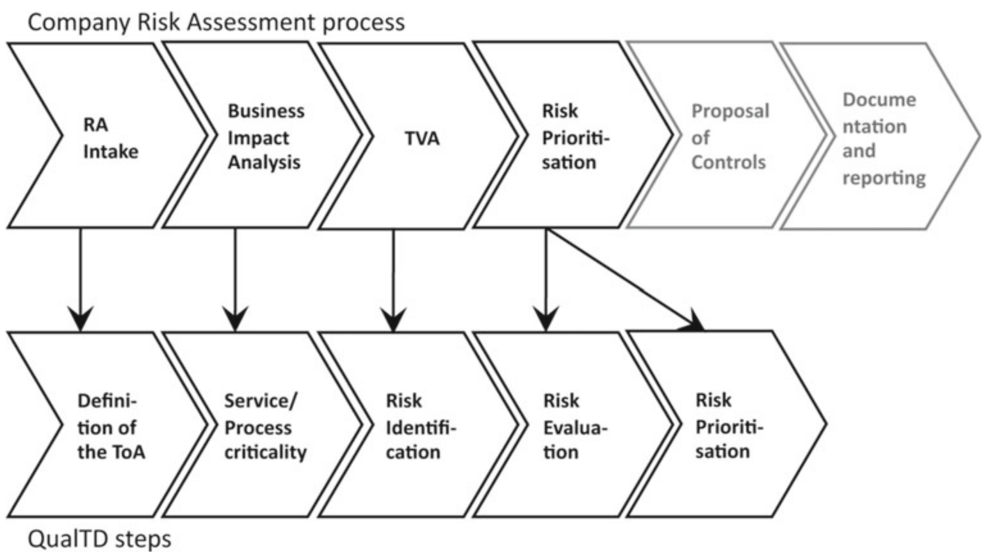

Fig. 5 The internal RA process (*above*) linked to the steps of the QualTD technique which complement the process (*below*)

The existing RA method In 2008, the RMC department carried out an RA on the Oxygen system following its internal RA process, which is mainly based on the guidelines provided by BS7799-3 [5], while the official security control policy is compliant with the ISO 27002 [18] standard.

The upper part of Fig. 5 depicts the process usually followed by RMC. In the following list we describe the six tasks composing the RMC process and we link them with the steps of the QualTD technique.

1. *RA intake*: the RA team (composed of people from the RMC department) and the requester project responsible agree on the scope of the RA and the Target of Assessment (ToA). The requester also submits proper documentation about the IT service to the RA team. This task corresponds to the definition of the ToA (see Sect. 2.1) in the QualTD technique.
2. *Business Impact Analysis (BIA)*: the RA team, together with the owner of the ToA, determines the desired levels of Confidentiality, Integrity and Availability for the ToA (e.g., HIGH integrity and availability and LOW confidentiality). They do this by analysing the impact that a breach of one of the three security properties on the information managed by the ToA would have on the business unit in a realistic worst-case scenario. They also determine which legislation or regulation requirements the ToA has to comply with (e.g., SOX [45] compliance). During this task the definition of the service/process criticality in the QualTD technique (see Sect. 2.1) should be made.
3. *Threat/Vulnerability Assessment (TVA)*: the RA team analyses the ToA and determines which threats/vulnerabilities the ToA is exposed to. Risk identification is based on a fixed list of threats/vulnerabilities which has been derived from a number of existing RA standards (e.g., BS7799-3, ISO 17799, BSI IT-Grundshutz [5,16,40]) and customised to fit the needs of the Company. The BIA influences the TVA in the sense that the threat list is customised according to the required levels of confidentiality, integrity and availability of the ToA: the higher the security level, the more detailed the list. The list is then used to check if the main components of the ToA (e.g., network communication, user interface, etc.) are exposed to the threats/vulnerabilities. At this stage, threats/vulnerabilities are flagged as applicable/not applicable to the considered component of the ToA, and as covered/not covered according to the fact that controls that could mitigate them are already deployed. This task corresponds to the risk identification step (see Sect. 2.2) in the QualTD technique.
4. *Risk prioritisation*: it consists in the evaluation of likelihood and impact of the threats/vulnerabilities which have been marked as applicable and not covered during the TVA. The risk assessors estimates the likelihood of a threat/vulnerability based on the company likelihood model, which takes into account several factors, e.g., resources, technical skills and time needed, or attacker motivation. They estimate the impact of a threat/vulnerability, based on the possible incident scenarios that the threat/vulnerability could determine in the ToA. These scenarios are figured out by the RA team based on their personal skills and their knowledge of the ToA. Likelihood and impact are then combined to determine the resulting risk, based on a risk aggregation matrix very similar to the one of Table 2. Threats and vulnerabilities are then prioritised based on their risk level: the higher

the risk, the higher the priority for controls. This task corresponds to the risk evaluation and to the output of the QualTD technique steps (see Sects. 2.3, 2.4).

5. *Proposal of Controls*: the RA team proposes a plan to cope with the identified risks, and identifies controls to mitigate the likelihood of the threats or to protect the ToA from the identified vulnerabilities. Examples of proposed controls include password policies, authentication mechanisms or Intrusion Detection/Prevention Systems.
6. *Documentation and reporting*: the RA team presents the results of the RA to the requester. It is not mandatory for the requester to communicate with the RA team about follow-up actions taken as a consequence of the RA.

The average time needed for an RA is approximately 240 man-hours (2 people for 3 weeks), depending on the size of the ToA (usually, RMC carries out RAs on ToAs which are comparable in size with Oxygen). Roughly, the first 80 man-hours are spent on steps 1 and 2 and for reading all the relevant documentation, another 80 man-hours are spent on steps 3 and 4, and the remaining 80 man-hours are spent on step 5 and to prepare the final report to be exposed during step 6. The RA team consists of two people performing the same task independently and then peer-reviewing each other's findings to come to a more objective final result.

The RA team uses three main sources of information: (a) documentation provided by the requester, (b) interviews with the requester and (c) vulnerability scans and other forms of direct investigation of security weaknesses.

Documentation includes results from previous assessments (i.e., RAs and security auditing activities), all the design and development documents (i.e., functional specifications, security design, technical architecture design and software design) and SLAs and outsourcing contracts.

Interviews with the requester are carried out after reading the documentation to clarify doubts and to set the boundaries of the RA. Another interview is carried out to address the BIA and, after step 4, to discuss about the main risks identified.

Optionally, the RA includes active forms of investigation of security weakness. The general principle RMC follows is *trust but verify*, which means that documentation about security measures implemented is trusted, but verified in its main aspects by means of, for example, vulnerability scanners.

3.2 Availability RA using the QualTD model

In this section we describe how we employed the QualTD model together with the RA method of the Company for the new RA of Oxygen. The main difference of a RA carried out following the Company internal RA process only with one carried out following our technique is that we build a dependency graph of the ToA and link threats and vulnerabilities with each other and with the nodes of the graph to better estimate impacts. As we discuss in more detail in Sect. 4, we used likelihood estimates carried out by the Company RMC personnel, since the QualTD model does not specifically address this topic.

We combined the QualTD model with four tasks of the Company internal RA process, as we show in the lower part of Fig. 5. First, we included in the *RA Intake* the activity of building the dependency graph. We spent 80 man-hours to perform this task. We also re-performed part of the BIA: instead of only defining the security requirements for Confidentiality, Integrity and Availability, we also assessed the criticality level of the main IT services of the ToA. We spent one man-hour on this. Finally, we carried out the *Threat Vulnerability Analysis* and *Risk prioritisation* by using the QualTD model as we explained in Sect. 2. We spent 72 man-hours to perform this task.

To build and run the QualTD model for Oxygen we relied on two sources of information: technical documentation and interview sessions. In practice we used the same documentation the RA requester provided for RA_1, as we describe in Sect. 3.1. In more detail, four documents were made available for the RA:

1. *The functional specification document*: this document describes the functionalities provided by Oxygen and how the functional architecture is designed, i.e., software components, what is their task and how they relate to each other.
2. *The security architecture and design document*: this document describes which security measures are implemented, e.g., server redundancy, and how they are implemented, e.g., which services are redundant and where they are located.
3. *The internal SLA document*: this document describes the quality of service parameters which are guaranteed to the users of Oxygen. In the context of availability, this document describes the availability figures for the different services provided by Oxygen, e.g., the authentication service is guaranteed to be available 99% of the times.
4. *The network diagram*: this document describes which are the actual servers running the different components of the Oxygen system, which software they are running and in which datacentre they are being managed.

We now describe in detail the activities we performed. For the sake of exposition we split the description according to the tasks that compose the Company RA process. Each task is further split according to the related step of the QualTD model of Sect. 2, as shown in Fig. 5.

3.2.1 RA intake

Defining the ToA The first step is building the dependency graph for Oxygen. According to the level of abstraction required for this RA, we modelled the following node types:

1. *Datacentres*: from the security architecture document and the network diagram we extracted the two buildings hosting the datacentres in which the servers are split for redundancy purposes.
2. *Network components*: from the security architecture document and the network diagram we extracted the firewalls protecting the different servers and enabling access to the Oxygen services from the internal network.
3. *Servers*: from the security architecture and the network diagram we extracted which servers are used.
4. *Applications*: from the security architecture, the network diagram and the functional specification documents we extracted the applications running on each server.
5. *IT Services*: from the functional specification and the internal SLA document we extracted the services exported by Oxygen, linking them to the applications implementing them.

The most challenging task in building the dependency graph was determining the dependencies among the nodes. The dependencies among buildings, network components, servers and applications could be inferred from the network diagram and the security architecture. Unfortunately, the functional specification document, which should link software to IT services, only referred to "logical" software components, which are not directly linked to the servers and the applications running on them. For instance, the functional component which acquires identity information from the different authoritative identity sources is actually implemented by three different applications: a Java-based web service, a Directory service and a DBMS; in turn, the DBMS also supports other functional components. To determine these dependencies we proceeded by refinement: whenever in the documentation we found that a certain application runs on a certain server, or that the application implements a certain service, we drew a new dependency among these nodes. Then, we cross checked the information from the functional specification and the network diagram documents to make sure the dependencies we found were consistent throughout all the documents. When we found an inconsistency, we updated the model and iterated the process. We reached a "stable" version of the model after the third iteration of this process.

To support this step we developed a graphical tool. The tool allowed us to draw the dependency graph, show it and modify it quickly during the interview sessions. The resulting graph is made of 65 nodes and 112 edges. Among the nodes we count 13 IT services, 32 applications, 14 servers equally distributed between 2 datacentres and connected simultaneously to 2 different network segments by means of 2 different firewalls. Building the first prototype version of the graph took us approximately 40 man-hours, using only the four documents we described as a source.

After building this prototype version of the dependency graph we checked it with the RMC personnel during an interview session: we showed the graph and explained the reasons motivating each dependency drawn; we then asked for possible missing ones. For example, we showed that a failure in the DBMS would lead to the unavailability of the identity data acquisition service and we asked if this conclusion was consistent with their knowledge of the system. The answer was positive; no inconsistencies were found during this session. Finally, we performed another interview session with the developers of the system to further check for consistency and completeness of the dependency graph. During this session we focused our explanation of the graph on the reasons motivating the choice of modelling a dependency between two nodes. For example, we motivated the choice of drawing a dependency from the DBMS to the application server since the Web Service uses the DBMS to store configuration parameters, and the unavailability of the DBMS would cause the Web Service to be unable to operate in turn. We found some discrepancies between our model and the behaviour of the system which is currently implemented. These discrepancies were due to inaccurate or outdated information in the functional specification document: we decided to keep the graph coherent with the actual implementation of Oxygen, instead of the one present in the documentation. RA_1 did not spot these discrepancies, as the analysis of the ToA required to build the dependency graph is much more detailed than the analysis required for an assessment which does not require to build any formal model.

Figure 6 shows an anonymised version of the dependency graph we obtained at the end of this task. During the task, although we did not know anything about Oxygen before our RA, we were able to build the dependency graph based on the available documentation. We only relied on interviews to confirm the correctness of the graph, not to build the graph itself. This ensures the method can be used by any risk assessor, who must not be an expert of the ToA.

3.2.2 Business impact analysis

After we built the dependency graph, we considered the business impact analysis (BIA), which consists of determining the required level of availability for the whole Oxygen system and the criticality level of all the IT services exported by Oxygen. We did this by interviewing the GIT department board, together with a member of the RMC department.

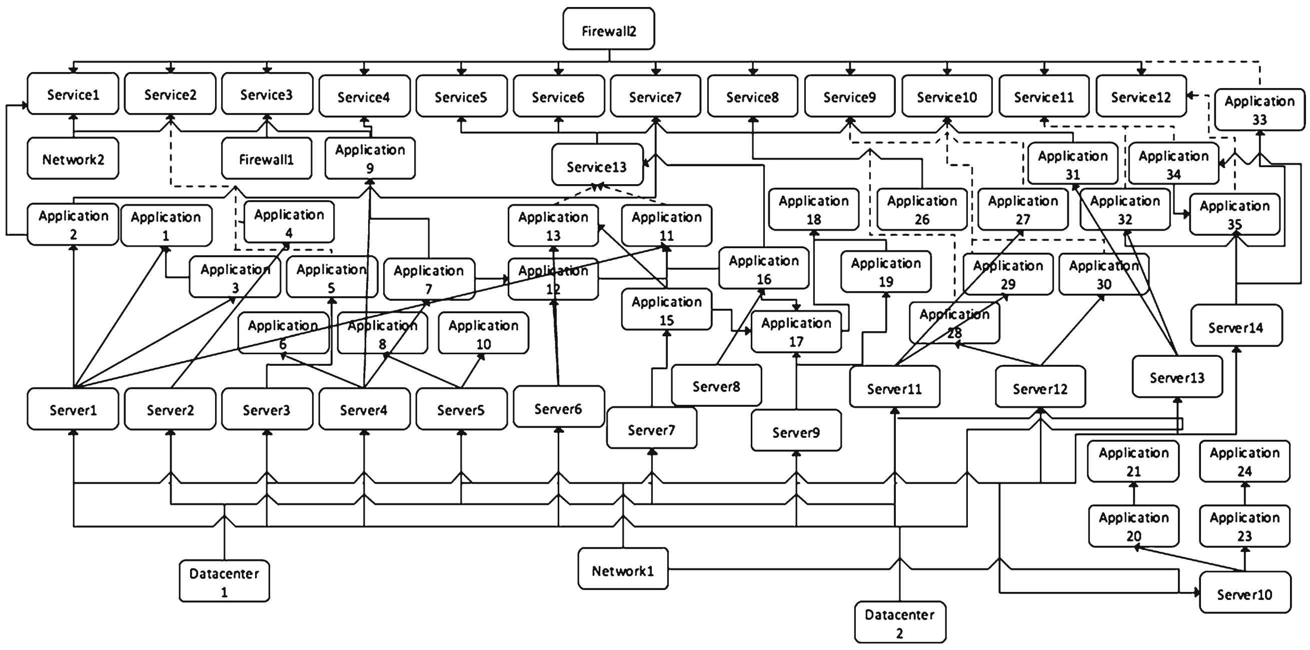

Fig. 6 This dependency graph resembles the one actually built for Oxygen. We observe from the *bottom*: datacentres, network components, servers, applications and IT services. *Solid edges* are *AND* dependencies, while *dashed edges* are *OR* dependencies

Since the required level of availability for Oxygen had already been assessed during RA_1, we only made sure that that part of the BIA was still valid. The GIT personnel confirmed that Oxygen requires a `High` level of availability. We then used this parameter during the risk identification phase for the selection of the threats and vulnerabilities to be used, as we describe in Sect. 3.2.3.

The new step of the BIA required by the QualTD model, which is not part of the RA method of the Company, consists of assessing the criticality of the IT services. For each IT service in the dependency graph we asked the GIT personnel if it had a `High`, `Medium` or `Low` criticality. In this way we defined the *criticality* function (see Definition 2).

After this last interview we had a final (approved) version of the dependency graph representing the ToA.

3.2.3 Threat/vulnerability analysis

Risk identification Recall that the RMC department adopted a threat/vulnerability list for their RAs, which was extracted from a number of standard RA methods and customised to fit the needs of the Company. To be able to compare the results of RA_2 with RA_1 we used the same threats and vulnerabilities. We will describe in more detail the reasons why we chose to do this in Sect. 4.

The list comprises a total of 121 threats and vulnerabilities. Since we only assess availability risks, we selected the subset of this list with an impact on availability, relying on the classification done by the RMC which determines for each entry if it has an impact on confidentiality, integrity or availability. Consequently, the set T was composed of 22 threats and the set V of 39 vulnerabilities. Moreover, according to the Company RA method, threats and vulnerabilities are selected based on the required level of Confidentiality, Integrity or Availability for the ToA. Since the level of availability of Oxygen has not changed in the two RAs we are allowed to use the same availability threats and vulnerabilities.

The next step we carried out was to link threats with vulnerabilities. During RA_1 threats and vulnerabilities were assessed separately, while the QualTD model requires us to link threats with vulnerabilities (thereby making explicit the reasoning that was implicitly done during RA_1). We did this by selecting, for each of the 22 threats, which one of the 39 vulnerabilities the threat can exploit to materialise. To validate our threat-vulnerability mapping we explained our choices to the RMC personnel during an interview session, and we integrated our mapping based on their opinion. Although no major inconsistency was found, we had to change a small number of mappings, because of a misinterpretation of the description of some threats.

Subsequently, we determined which nodes of the dependency graph were targeted by threats and in which nodes a certain vulnerability was present. To do this we evaluated which kind of node the threat/vulnerability applies to; for example, a power disruption can only affect a datacentre and a DoS attack can only affect software nodes.

Finally, we enumerated the availability incidents following Definition 5. This task was performed automatically by intersecting threats with the nodes they target, vulnerabilities with the nodes they are present in and threats with the vulnerabilities they can exploit. We inserted all this information in a database. Therefore, listing incidents was nothing more than building a view on the existing table schema. We checked our results with the RMC personnel, to detect inconsistencies in our mapping, but we found no discrepancy, as mapping threats and vulnerabilities to asset types was quite an unambiguous task.

3.2.4 Risk prioritisation

Risk evaluation We used the estimates of the likelihood of threats and vulnerabilities from RA_1, (for the definition of the *t-likelihood* and *v-likelihood* functions see Definitions 8 and 10). The estimate was done in terms of `High`, `Medium` and `Low` likelihood level, according to the likelihood model adopted by the RMC team, which is based on eight different parameters (e.g., time needed for the attacker, technical skills needed, etc.). The reason why we did not do our own estimate of the likelihood is twofold: first, we needed to ensure that the results of the two RAs could be comparable, and since our model only implies a different way in estimating the impact, likelihood had to be kept fixed. Second, since the results of this second RA are meant to be used by GIT, we wanted the likelihood estimates to be based on the professional judgement of the RMC personnel, instead of ours.

To assess incident duration (i.e., the dt function of Definition 9) we first used the Company-internal SLAs to set the threshold between a `Short` and `Long` incident duration. The Company-internal SLAs give an availability figure for the IT services provided by Oxygen. For example, they guarantee that the identity data acquisition service will be available for a certain fraction of time in a month. We set the threshold as the longest amount of time (in hours) the service can be out of service while remaining compliant with its SLA. For example, if the availability figure is 99.5% in a month (i.e., 30 days), we set 4 h as our threshold. We choose this measure, since, in this case, the SLAs were set to give an indication about how long a certain service can be disrupted without causing excessive problems to the Company's business. In this way, we distinguished between `Short` incidents (i.e., those shorter than the maximum tolerated disruption time in a month) and `Long` ones (i.e., those which last longer than the maximum tolerated disruption time in a month). Subsequently, we analysed the time needed to solve each of the incidents. We considered both the time needed to detect the disruption and the time needed to fix the problem. The resulting total disruption time, which we compared with the threshold, is the sum of these two parameters. We performed this analysis based on both the information we gained from the SLA document the Company has signed with the outsourcer, and the opinion of the developers of the Oxygen

Table 1 Global impact level determination

Impact level	Definition
`Critical`	At least one service/process with `High` criticality is disrupted for a `Long` period of time
`Serious`	At least one service/process with `High` criticality is disrupted for a `Short` period of time
`Significant`	At least one service/process with `Medium` criticality is disrupted for a `Long` period of time
`Moderate`	At least one service/process with `Medium` criticality is disrupted for a `Short` period of time
`Marginal`	At least one service/process with `Low` criticality is disrupted for a `Long` period of time
`Insignificant`	No service/process is disrupted or only service/process with `Low` criticality are disrupted for a `Short` period of time

system. The SLA document contains the maximum response time for incidents happening in the portions of the system for which management has been outsourced. For all the remaining parts of the system we relied on the judgement of the GIT developers.

With this we had acquired all the information needed to run the model and obtained the *global impact* of the incidents and their risk. For each incident i we used the dependency graph to determine the set $Prop_i$ of the processes and services which were affected by the incident given the IT components the incident directly targets as we described in Definition 7. Subsequently, we used Table 1 to determine the global impact level. The definitions we used are based on the requirements for availability the GIT has set on Oxygen during the meeting in which we assessed the criticality of services/processes. These definitions are an implementation of the combination of the composition function *harm* and the aggregation function *impact-agg* of Definition 11.

We then used the definitions of Table 2 to determine the risk level associated with every incident. The definition of the risk level we give was built on the indications of the RMC personnel, and it is an implementation of the function *i-risk* of Definition 12.

The choice of using these two tables to evaluate the global impact and the risk level was driven by two main motivations: first, the functions defined by the tables are monotone; therefore, they are compliant with the requirements of Definitions 11 and 12, and they allow one to trace back the reasons causing the assignment of a certain risk level to a certain incident (see Running example 13). Second, the alternative choice of assigning a numerical value to each qualitative one (e.g., `High` = 3, `Med` = 2 and `Low` = 1) and then performing mathematical operations on them (e.g., sum, multiplication or average) would not work in our case. In fact, although this is a very popular and widely adopted technique in RAs

Table 2 Incident risk level determination

Risk level	Definition
`High`	Impact is `Critical`, both threat and vulnerability likelihood are `Medium`. Impact is `Serious`, both threat and vulnerability likelihood are `High`
`Med-High`	Impact is `Critical`, either threat or vulnerability likelihood is `Low`. Impact is `Serious`, both threat and vulnerability likelihood are `Medium`. Impact is `Significant`, both threat and vulnerability likelihood are `High`
`Med`	Impact is `Serious`, either threat or vulnerability likelihood is `Low`. Impact is `Significant`, both threat and vulnerability likelihoods are `Medium`. Impact is `Moderate`, either threat or vulnerability likelihood is `High`
`Med-Low`	Impact is `Significant`, either threat or vulnerability likelihood is `Low`. Impact is `Moderate`, both threat and vulnerability likelihood are `Medium`. Impact is `Marginal`, both threat and vulnerability likelihoods are `High`
`Low`	In other cases

(e.g., see Cunningham et al. [7]), it only provides meaningful results if we know the exact *ratio* among the qualitative values (e.g., if we knew that `High` is exactly three times `Medium` we could assign 9 to `High` and 3 to `Medium`). Since our RA was carried out in a completely qualitative manner, we only know that `High` is bigger than `Medium`, but we do not have any indication on how big the ratio is between them; therefore, we cannot perform any mathematical operation on these values. In other words, we work with values in an *Ordinal scale*, while the other approach would at least require values in an *Interval scale*, as shown by Herrmann [12].

Having determined the risk level, we ranked availability incidents according to their risk. However, to complete the outcome of the threat/vulnerability assessment step, we also needed to rank the most dangerous threats and vulnerabilities for Oxygen. We did this by assigning each threat/vulnerability the risk of the incident they cause, which has the highest level associated. In other words, we used *max* as the aggregation function *risk-agg* of Definition 13.

4 Case study evaluation

In this section we make an evaluation of our case study. To this end, the methodology we follow is the one introduced by Wieringa et al. [29,30] for technical research, which is based on the following two statements:

1. `solution & context` *produces* `effects`
2. `effects` *satisfy (to an acceptable extent)* `stake-holder-motivated criteria`

Wieringa et al. observe that each technological solution which is applied in a context produces some effects on it. The effects may (or may not) contribute to satisfy some goals defined by the stakeholders of the research context. The evaluation criteria set by the stakeholders must be in a measurable or comparable form, so that if two different solutions are applied to the same context, they can be evaluated and compared with relation to these criteria. The reasoning scheme can be applied when a solution is specified but not yet implemented [11] or after a solution is implemented [24].

In our case, the technical solutions to be evaluated are the RAs performed on the Oxygen system: the first is done following the RA method of the Company and the second made by integrating the same method with the QualTD model. The context in which we apply these solutions is described in Sect. 3.1.

4.1 Stakeholders, goals, and criteria

First, we present the stakeholder's goals and the derived evaluation criteria, which we have already briefly introduced in Sect. 3.1. These goals regard both specifically (the security of) Oxygen and the quality of the general RA process. Methodologically, we derive the goals by analysing the description of the activities GIT provided us during the interviews; subsequently we defined the criteria to measure those goals. Finally, we validated the goals and criteria by means of interviews with the stakeholders. For the sake of the presentation we only report the results of this activity in the list below. Although our case study will not allow us to evaluate all the criteria, we report them all to give an overview of stakeholder's objectives.

Goals and criteria regarding Oxygen:

– *GIT*

1.1 The goal *Ensure cost/effective mitigation controls and timely mitigation plans* is measured by the quality criterion *Cost for managing High/Medium/Low risks*.
1.2 The goal *Implement controls with the least possible contractual and financial impact* is measured by the quality criterion *number of controls with contractual and financial impact*.

– *Services depending on Oxygen*

1.3 The goal *Have the authentication/identity service for their application available when needed* is measured by the quality criteria *number of times authentication was not available in one month* and *number* *of times identity management was not available in one month.*

– *The Service Provider*

1.4 The goal *Manage systems with the least possible effort and by remaining compliant with SLAs* is measured by the quality criteria *Euro/resources employed for managing hardware/software and to guarantee SLAs* (*including consequences for not fulfilling contractual obligations*).

Goals and criteria regarding the RA process:

– *RMC*

2.1 The goal *Ensure good quality of the RA Service* is measured by the quality criterion *number of important risks for the RA requester identified during an RA vs. number of unimportant risks*.
2.2 The goal *Make the RA process more efficient* is measured by the quality criterion *number of man-hours employed for an RA by the members of the team*.
2.3 The goal *Make the RA process less subjective* is measured by the quality criterion *number of choices let to the risk assessor*.

– *GIT*

2.4 The goal *Use global* (*shared*) *solutions to solve the same problem in different systems* is measured by the quality criteria *number of months to implement controls* and *number of different solutions employed to solve the same problem in different systems*.

4.2 Design of the evaluation process

Given the stakeholders goals and criteria, we use them to analyse and compare the results of RA_2 with those of RA_1.

First, we briefly discuss the procedure we followed. In this analysis we assume that, given a method to calculate the risk in an RA, the quality of an RA is only determined by the knowledge of the risk assessor about (a) the ToA, (b) threats and their likelihood, (c) vulnerabilities and their likelihood and (d) how threats, vulnerabilities relate to each other and impact the ToA. We choose not to include all the social/organisational factors, e.g., the relationships among the stakeholders and their commitment to IT security, the alignment of all the stakeholders with respect to the organisation business goals, etc. These factors are indeed very important for the success of an RA but, for the sake of this evaluation, we assume them to have remained steady in the Company throughout the two RAs, and therefore to have no impact. For more examples of other IT RA social/organisational success factors, please refer to [9,35]. The experiment we carried out compares the results of two RAs, performed sequentially by different people on the same IT system. For these reasons, to keep the experiment under control, we needed to make sure that (1) the order in which the RAs were carried out does not influence their results, and (2) the quality of the results does not depend on the security skills of the people carrying out the RAs. To accomplish these conditions we conducted RA_2 before having access to the results of RA_1, but using the same sources of information. We used the same list of threats and vulnerabilities, as well as the same likelihood estimation, in both the RAs and we made sure the technique we employed to relate threats, vulnerabilities and nodes did not depend on the particular security skills of the risk assessor.

Table 3 summarises the conditions that we enforced to ensure the two RAs are comparable.

In the next sections we compare the results of RA_2 with relation to RA_1. To do that we use four evaluation criteria from the list of Sect. 4.1. These parameters are (2.1) the number of important risks for the RA requester vs. the number of unimportant ones (recall that in RA_1 a risk is the combination of the likelihood and impact of a threat/vulnerability), (2.2) the number of man-hours employed to carry out the RA, (2.3) the number of choices that the RMC personnel have to take and (2.1) the cost of managing availability risks. The other criteria of the list are not decidable by a risk assessor but would be observable after the system has been in use for a while. An RA will have an impact on how the system scores on these criteria but based on our evaluation alone we cannot tell what the impact of our technique will be.

For the sake of presentation, we summarise the results: (1) the QualTD model has improved the (perceived) accuracy of RA_2 by increasing the number of identified important risks for the RA requester, (2) it introduced an overhead in the number of hours employed, (3) it helped reducing the subjectivity of impact estimates in RA_2 and (4) thanks to the effects of points (1) and (3), the QualTD model supports a better risk prioritisation, which is one of the requirements for optimising the cost of risk mitigation.

To further substantiate our findings, our technique should be tested by people who did not participate in its development. We plan to have this test done by the RMC personnel of the Company.

4.3 Evaluation of the criteria

Evaluation of Criterion 2.1: number of important risks for the RA requester vs. number of unimportant risks The first evaluation criterion is given by the number of important risks for the RA requester with respect to the less important ones and it expresses the `result quality` of an RA method. With *important risks*, here we mean the threats/vulnerabilities which have a high or medium risk level (i.e., the ones that will be taken into account when deciding the risk profile of

Table 3 RA comparison control variables

	(1) RA Order	(2) Security skills
(a) ToA	Used the same documentation in the two RAs. RA_2 is blind to the results of RA_1 (see Sect. 3.2.1)	Build the dependency graph does not require to be an expert of the ToA (see Sect. 3.2.1)
(b) Threats & likelihood	The same threat list and likelihood estimation was used for RA_1 and RA_2 without any change (see Sect. 3.2.3)	Only the security skills of the RMC team have been employed in the two RAs for threat identification and likelihood estimation (see Sect. 3.2.3)
(c) Vulnerabilities & likelihood	The same vulnerability list and likelihood estimation was used for RA_1 and RA_2 without any change (see Sect. 3.2.3)	Only the security skills of the RMC team have been employed in the two RAs for vulnerability identification and likelihood estimation (see Sect. 3.2.3)
(d) Combining threats, vulnerabilities and nodes	RA_2 does not use any information of RA_1 about this (see Sect. 3.2.3)	We combined threats with vulnerabilities in accordance with the personnel who carried out RA_1. Linking threats/vulnerabilities with nodes does not depend on particular security skills (see Sect. 3.2.3)

the system and the risk mitigation strategy) which are judged to have been assessed accurately. In this case the number of relevant risks identified in the two RAs is not influenced by the number of threats/vulnerabilities identified or by their likelihood, as the list of threats/vulnerabilities remained the same in both assessments as well as their likelihood estimation. On the other hand, the risk of a threat/vulnerability can be overestimated or underestimated in case certain incidents and their impact are not taken into account in the RA. In this case, we would have important risks which are not considered when the risk level of the corresponding threat/vulnerability has been underestimated, or less important risks considered as important, when the risk level of the corresponding threat/vulnerability has been overestimated. We focus our evaluation on this aspect.

To determine the performance in identifying important risks of RA_2 with respect to RA_1, we compared and analysed the results of the two RAs together with the RMC personnel.

First, we made sure that risks were evaluated following the same criteria in both RAs, i.e., given the same threat and vulnerability likelihood and impact levels, the resulting risk level is the same.

Second, we analysed the cases in which the two RAs gave different results and we analysed the reasons for the difference. Table 4 summarises our findings. The RMC personnel acknowledges that in all cases, the risk estimation made in RA_2 is more accurate than the one previously made in RA_1. For this reason, in Table 4 we set the estimation given by RA_2 as a reference for RA_1 and we say RA_1 overestimates the risk level of a threat/vulnerability when the risk level given by RA_1 is higher than the one given by RA_2 for that threat/vulnerability. The same applies when the risk level given by RA_1 is lower than the one given by RA_2, in this case we say RA_2 underestimates the risk level of a threat/vulnerability.

In seven cases, the reason of the difference was due to external causes that do not involve the use of the QualTD model. For example, in RA_1 the vulnerabilities regarding the configuration of the Company network were usually underestimated on purpose. This because the final report of the RA carried out without the model was directed to the GIT board, who is not directly managing the Company network. Consequently, the judgement of the RMC team was that it was not useful to point out the obvious in the report, since the RA requester had no way of managing that kind of risk. The remaining 14 differences are due to a better quality of RA_2.

According to our analysis, the success of RA_2 is due to the fact that the QualTD model enables the risk assessor to estimate with more precision the impact of a threat materialising, and also to determine the impact of the vulnerabilities, by explicitly linking them to the incidents they can cause: all these operations are hard to perform without an architecture model that allows one to reason about the availability impact. For example, the impact of malware (e.g., worms) spreading across the Company network and infecting the (few) Windows servers of Oxygen were underestimated due to the lack

Table 4 Summary of the number of differences between the two RAs

	Threats	Vulnerabilities	Total
Related to Availability	22	39	61
RA_1 overestimates risk level	1	2	3
RA_1 underestimates risk level	5	13	18
Differences caused by factors not related to the QualTD model	1	6	7
Differences caused by using the QualTD model	5	9	14

of awareness of the risk assessors during RA_1 about the connections between these servers and other core components of Oxygen.

Evaluation of Criterion 2.2: number of man-hours employed for an RA We split the analysis on time consumption of the two RAs according to the four steps of the Company RA process supported by the QualTD model.

1. *RA intake*: the time needed to accomplish this step with the Company method is 80 man-hours (by two people) on average. Building the dependency graph certainly constitutes an overhead, since it requires to formalise the knowledge acquired from the documentation and it also required at least one additional meeting with the developers of the system. In our case, we spent approximately 80 man hours (by one person) to finish the RA intake step using the QualTD model. About 40 man-hours were needed to gain knowledge of the Company, which would not have been necessary by an experienced RA team in the Company itself. So we think that one person of the RA team of the RMC department, experienced as we are, would have needed about 50 man-hours to build the dependency graph. Currently, the RA intake takes 80 man-hours (by two people), so the overhead introduced by our model would be of approximately 10 man-hours. Whether this is worth the investment depends on the benefits to be gained from this in terms of a more accurate RA and in terms of the reusability of this graph for future RAs of this or other (related) systems.
2. *BIA*: including the estimation of the service/process criticality into the business impact analysis is an inexpensive task, since it is already included in the procedure followed by the RMC personnel, only in an informal way. Moreover, we experienced that it was easy for the GIT to rank the services by criticality, since this knowledge is part of their everyday business. Formalising service/process criticality took less than one man-hour.
3. *TVA*: differently to the Company method, the QualTD model explicitly requires to link threats and vulnerabilities to the nodes of the dependency graph to evaluate the risk. This task took us approximately 30 man-hours more than the time normally employed by the RMC personnel. However, this is partly due to the fact that we had to "learn" and get used to the definitions of the threats and vulnerabilities of the list provided by the Company. We estimate that, should we have known them better we would have done the same job in half the time. Moreover, another good part of the work was that of manually linking threats and vulnerabilities to nodes; we did this step by hand and it was very time consuming: a proper GUI would have saved us other time.
4. *Risk prioritisation*: using the QualTD model does facilitate this step. In fact, following the Company RA process, the RA team has to perform a (time-expensive) peer review of the risk evaluation performed by each member of the team, i.e., the team members have to go through their personal estimation of likelihood and impact for each threat/vulnerability and, in case they find any discrepancy, determine the reasons motivating each decision and reach a final agreement on the proper likelihood/impact levels. The QualTD model allows one to automatically prioritise threats and vulnerabilities. Moreover, as risks are evaluated in a more detailed level (i.e., incidents instead of threats/vulnerabilities), the QualTD model facilitates the discussion on the final impact level of threats/vulnerabilities. For example, during the discussion with the RMC personnel on the final results of RA_2, we used the model to explain why a certain threat or vulnerability had a certain risk level by going into detail on the incidents that these threats and vulnerability are involved in. This technique was judged very useful and practical by the RMC personnel. It is also possible to reuse most of the work of linking threats, nodes and vulnerabilities for future RAs on the same ToA, this would reduce to zero the difference with the original method in the time consumption on the TVA step.

Evaluation of Criterion 2.3: number of choices let to the risk assessor Making the RA results more inter-subjective (i.e., shared among the RA stakeholders) is one of the original goals of the RMC department, which aims at (a) delivering better quality results by identifying as many potential and relevant risks as possible, and (b) being able to justify the reasons why a certain threat or vulnerability was given a certain risk level.

The QualTD model supports the first objective by "forcing" the risk assessor to systematically explore all the possible combinations of threats and vulnerabilities, thus reducing the risk of mis-estimating the importance of a certain threat or vulnerability.

Regarding the second objective, since the QualTD model requires to enumerate explicitly availability incidents, it is easier for the risk assessor to trace the reasons why a threat/vulnerability was given a certain risk level (recall that we can calculate an aggregated risk level for both threats and vulnerabilities from the incident risks). Moreover, a member of the RMC department has to give (explicitly or implicitly) four subjective estimates to evaluate a single incident: the likelihood of the threat, the likelihood that the vulnerability is present in some nodes, the duration of the incident and the criticality of the services/processes it hits. By applying the QualTD model, the global impact of an incident is based on the criticality of the nodes involved, which is given by the RA requester. In this way, we reduce by one fourth the number of choices to be taken by the RMC personnel (alone) for each incident, and increases the inter-subjectivity of the results, as the criticality of the services has to be agreed upon before the RA starts. In other words, even if the subjectivity of the estimates is still present, it depends less on the expertise of the single risk assessor, and it is shared with the risk assessment requester, who is the final user of the assessment results.

Evaluation of Criterion 1.1: Cost of risk management The budget for managing risks is always limited. In this perspective, optimising the costs of RM means achieving at least the same security level for at most the same price. To achieve this goal, it is important to adequately prioritise the risks one wants to manage in terms of (a) the risk level, and (b) the cost to mitigate that risk. By providing a more precise risk prioritisation based on (a), the QualTD model supports part of the decision process of prioritising risks for mitigation purposes. At present time, however, the model does not include any means of prioritising risks with respect to (b). Actually, our model bears similarities with the quantitative TD model [31] which, on the other hand, does include countermeasures and enables one to run an optimisation algorithm which select the best risk mitigation strategy taking into account (a) and (b). We believe that the same approach is applicable also to the QualTD model with few modifications. This is, however, beyond the scope of this paper, and left as future work.

4.4 Applicability to other scenarios

Based on the experience of the case-study, we observe that there are two main factors which determined the success of the QualTD model.

First, the model forces the RA team to follow a more systematic approach; this means that there is less space for human errors and that the model provides an affordable way to deal with the complexity of the ToA. The QualTD model shares this characteristic with many other model-based approaches, as for example model checking techniques. This also means that, as other model-based approaches, it requires a preliminary investment in terms of time and resource to build the model. With this case-study we showed that the time investment does not exceed 50% of the time spent in an RA carried out without the model, and the resources commonly available for an RA are sufficient to build the model. In general, this investment can be very worthwhile (because, e.g., it allows one to reuse the information gathered or it allows one to identify problems that would remain undetected with other techniques), or just a waste of resources. In our case, as confirmed by the RMC team, a QualTD model built for an RA can be widely reused in the following RAs of the same ToA; the resource investment can thus be compensated by reusing the model in successive RAs. This makes it particularly suitable when the ToA is periodically subject to RAs. Moreover, we believe our model-based RA approach should only be used when it either allows one to save resources in the long run (as explained above) or when the need for accurate results is worth the effort of using it. In the case analysed here, Oxygen is an availability-critical system for the Company, and therefore the need for accuracy in the assessment justified the time overhead it introduced. Also, the need to optimise the budget for risk mitigation could be a leading factor for choosing the QualTD model and afford its initial time overhead. Another scenario in which using the QualTD model could be convenient is when the dependency graph can be built automatically (e.g., when a configuration management database is already present and can be used to build the graph), since in this case there is almost no time overhead.

A second success factor of the QualTD model is that it links the knowledge about security with the components of an IT architecture, their technical and functional dependencies and their importance. With this case-study we showed that the QualTD model structures information in a way that is simple enough to be used and complete enough to cover all the aspects that are important for a security RA. In fact, we did not find any uncovered risk area in RA_1 which was not covered in RA_2. For this reason, we think that the QualTD model is particularly suitable to be used to assess the availability risks of an IT infrastructure or of parts of it.

5 Related work

This section is divided into two parts. In the first part we make a taxonomy of standard RA methods, and we single out the methods, or the characteristics of these methods, that are compatible with the technique we presented in this paper.

In the second part, we do a literature review of the techniques that use dependency-based models to improve the quality of an RA; we compare them with the technique we presented in this paper and we discuss their applicability to the paper's industrial case.

5.1 Combining the QualTD model to standard RA methods

In this part we look at general RA methods, and we discuss under which circumstances the QualTD model can be used in combination with them. To do this, we first make a taxonomy of RA methods.

5.1.1 A taxonomy of RA methods

To provide a snapshot of the state-of-the-art within RA methods we follow the survey by the European Network and Information Security Agency (ENISA) [39]. The survey consists of a list of sixteen RA methods currently in use. Among these methods we only consider international standards, i.e., those which are available in English and which are actually in use in more than one country. According to these criteria, we reduce the initial list of sixteen methods to ten: CRAMM [36], EBIOS [38], ISAMM [37], ISO 13335-2 [14], ISO 17799 (now ISO 27002) [18], ISO 27001 [17], IT-Grundshutz [40], MEHARI [42], OCTAVE [32], NIST SP 800-30 [27]. The remaining six methods are dropped because of two reasons: Austrian IT Security Handbook, Dutch A&K Analysis and MARION because only available in a single language (German or Dutch), while ISF, MAGERIT and MIGRA because of lack of relevant documentation. Finally, since the list on the ENISA survey is not admittedly complete, we augment it with another popular method, the Australian/New Zeland standard for RM AS/NZS4360 [33], and with CORAS [8], the method resulting from the EU-funded project IST-2000-25031. We explicitly choose to exclude Common Criteria [15] from this list as it is not properly an RA method, even if it requires some risk analysis to be performed.

For the sake of the presentation, we classify the 12 methods by means of three parameters: (1) the scale used to evaluate risk and risk factors (quantitative or qualitative), (2) which factors are proposed in the method to evaluate the impact level and (3) the underlying view on how risk is evaluated.

Parameter (1) determines if the risk level measures something that can be (meaningfully) expressed in numbers (e.g., money), or something which can only be expressed with labels (e.g., high, medium, low). In other words, a qualitative method measures the level of a risk factor in an ordinal scale (i.e., only ordering among values are known), while a quantitative method uses measures in interval or ratio scales (i.e., the magnitude of the difference between two values in the scale is known; ratio scales also define an absolute and non arbitrary zero point).

Parameter (2) indicates which factors influence the impact of a security event (i.e., a threat, a vulnerability or an incident), and to which extent the method is constrained by these factors. Some methods only give general guidelines (e.g., the damage to the organisation), while others strictly define a particular set of parameters (e.g., the money loss, or the affected business processes).

Parameter (3) investigates what determines the risk level of a security event and how different properties are combined. To this end we elaborated five different profiles (`Type 1` to `Type 5`):

1. `Type 1`:
   $Risk(Threat, Asset) = Likelihood(Threat) \otimes Vulnerability(Threat, Asset) \otimes Impact(Threat, Asset)$
   In `Type 1` methods, risk is analysed with relation to a threat and an asset, or a group of assets and it is evaluated as the combination of the likelihood of the threat, the vulnerability level of the asset(s) to the threat and the impact of the threat on the asset(s). We argue that this approach can be applied both to fine-grained assessments (i.e., taking into account single assets and asset-specific threats) and to more high-level assessments (i.e., taking into account only classes of assets and high-level threats).
2. `Type 2`:
   $Risk(Threat, Asset, Needs) = Impact(Threat, Needs) \otimes Vulnerability(Threat, Asset)$
   In `Type 2` methods, risk is analysed with relation to a threat, an asset and some security needs on the system and it is evaluated as the combination of the vulnerability of the asset and the impact of the threat on the security needs. We argue that this approach is suitable where security requirements are clearly specified, for example for software products developed by following a rigorous software engineering process.
3. `Type 3`:
   $Risk(Threat, Asset) = AnnualLossExpectancy(Threat, Asset) = Probability(Threat, Asset) \otimes AverageLoss(Threat, Asset)$
   In `Type 3` methods, risk is analysed w.r.t a threat and an asset, is intended as the annual loss expectancy (in monetary terms), and it is evaluated as the combination of the probability of the threat affecting the asset and the average loss of the resulting incident. We argue that this approach is suitable in all the situations in which decisions are taken based on a financial cost/benefit analysis (e.g., insurance companies), and in which quantitative data are available (e.g., for critical infrastructures).
4. `Type 4`:
   $Risk(Threat, CriticalAsset) = Impact(Threat, CriticalAsset) \otimes Vulnerability(CriticalAsset)$

Table 5 Classification of the RA methods

Method	Evaluation scale	Impact evaluation	Risk evaluation
CRAMM	Qualitative	Based on open damage scenarios	Type 1
EBIOS	Qualitative	Based on security needs	Type 2
ISAMM	Quantitative	Based on monetary loss	Type 3
ISO 13335-2	Both	Based on the business harm	N/A
ISO 17799	Qualitative	Based on the business harm	N/A
ISO 27001	Qualitative	N/A	N/A
IT-Grundschutz	Qualitative	Based on open damage scenarios	Type 5
MEHARI	Qualitative	Based on fixed damage scenarios	Type 1
OCTAVE	Qualitative	Based on critical assets	Type 4
NIST SP 800-30	Qualitative	Based on open damage scenarios	Type 1
AS/NZS 4360	Both	Based on a balance between business harm and business advantages	Type 5
CORAS	Both	Based on open damage scenarios	Type 5

In `Type 4` methods, risk is analysed with relation to a threat and an asset that has previously been identified as critical, and it is assessed as the combination of the impact of the threat on the critical asset and the vulnerability of the asset. We argue that this approach is suitable where there are critical assets to be protected (e.g., for utility network infrastructures).

5. `Type 5`:
$Risk(Incident, Asset) = Likelihood(Incident) \otimes Consequences(Incident, Asset)$
In `Type 5` methods, risk is analysed with relation to and incident (i.e., a combination of a threat and some vulnerabilities) and an asset, and it is evaluated as the combination of the likelihood of the incident and the consequences of the incident itself. Unlike for the `Type 1` approach, this approach attributes risk levels only to security incidents (i.e., a threat exploiting a vulnerability) to assess their risk. We argue that this means that it is more suitable to be applied to fine-grained RAs and it is harder to apply to the high-level ones.

Table 5 reports the results of the classification. Most of the methods are meant to be used with qualitative measurements, and this confirms the fact that most RAs today are carried out in a qualitative way, mainly due to lack of reliable quantitative data or to time constraints [3].

Regarding impact level evaluation, we observe that ISO 13335-2 and ISO 17799 only specify that the impact of a security event is tied to the business harm suffered from the organisation. Furthermore, AS/NZS 4360 also specify the possibility of a business advantage of undertaking a certain risk, e.g., leaving servers unpatched may lead to a quicker time to market for the organisation. CRAMM, IT-Grundshutz, NIST SP 800-30 and CORAS specify more precisely how the impact level should be assessed, since they introduce the concept of damage scenarios: the RA team should identify different impact scenarios (e.g., from `Catastrophic` to `Marginal`) which describe the negative consequences of a risk event on the organisation. We say that these scenarios are "open" as these methods do not specify a particular set of scenarios or they do not require to use the ones they propose. On the other hand, MEHARI is based on a "fixed" impact scenario, i.e., the description of the consequences is fixed, and the risk assessor can only rank them. EBIOS imposes that the impact level of a security event is assessed in terms of the security needs (i.e., a security requirement on the IT assets) that the event violates. Similarly, in OCTAVE the impact level is measured in terms of how "hard" the security event is hitting a mission-critical asset (e.g., a server which has been pre-determined to be critical for the organisation). Finally, ISAMM measures impact by means of the money the organisation can loose because of a security event.

Regarding risk level evaluation, we observe that CRAMM (which mostly implements the principles given in BS7799-3 [5]), MEHARI and NIST SP 800-30 share the same common view on risk, i.e., they all consider risk as a combination of the likelihood and the impact of a threat to hit a group of assets and the vulnerability level of this group of assets. Similarly, IT-Grundshutz, AS/NZS 4360 and CORAS consider risk as the combination of the likelihood of an incident (i.e., a threat exploiting some vulnerabilities) and the consequences (positive or negative) of this incident happening. On the other hand, `Type 2`, `Type 3` and `Type 4` profiles are intrinsically tied to a particular approach to RA, since `Type 2` and `Type 4` rely on qualitative concepts for defining risk (e.g., critical assets, security needs) and `Type 3` relies on the quantitative concepts of probability and average monetary loss. Finally, we observe that the methods of the ISO family do not adopt any risk analysis profile. This is due to the fact that, according to ENISA [39], ISO 13335-2 is a very general guideline to set up a RM framework, while ISO 17799 and ISO 27001 are actually not real methods for RM, but rather compliance standards, reporting a list of controls for good security practices and the requisites that an existing method should have to be standard-compliant, respectively.

5.1.2 Applying the technique based on the QualTD model together with other RA methods

With applying the QualTD model-based technique to an RA method we mean carrying out some specific parts of the RA process (i.e., definition of the ToA, BIA, risk identification, risk evaluation and risk prioritisation) for availability risks by using the QualTD model.

According to our classification scheme, the original RA method followed by the Company is qualitative, based on the business harm and `Type 1` with relation to the risk level evaluation. The new RA method which integrates our technique based on the QualTD model remains qualitative, but it is based on open damage scenarios and has a basic risk level evaluation of `Type 5`. We also define a procedure to aggregate the evaluation of incident risks per threat and vulnerability, making the evaluation scheme compliant to the original `Type 1`.

From the perspective of the risk level scale, the QualTD model can only be used together with a qualitative RA method (on the other hand the TD model we proposed in [31] can only be used with quantitative ones).

From the perspective of impact level determination, we showed in the present case how the QualTD model is compatible with methods evaluating the impact in terms of business harm.

On the other hand, for methods adopting damage scenarios, the integration with our technique is only possible if the scenario descriptions used by the organisation undertaking the RA can be associated with the unavailability of a node in the dependency graph.

For methods in which the impact level is based on critical assets, e.g., OCTAVE, the QualTD model cannot be applied as it is, since in the current specification we do not give a definition of critical assets. However, one possible way of adapting the model to this purpose consists in first determining the most critical processes/services and then using the dependency graph to find the nodes supporting those processes/services.

We also observe that it is hard to integrate our technique with methods based on security needs, such as EBIOS. Innerhofer-Oberperfler and Breu [13] introduced an approach, which shares some similarities with ours and is suitable to be used in combination with these methods: we will present this approach in more detail in Sect. 5.2.

Finally, in the present specification of the model, we do not consider the business advantage of a certain risky factor, as required by AS/NZS 4360: this is the only obstacle we see for the integration of the QualTD model with this standard.

Regarding risk level evaluation, the QualTD model can be integrated with any method adopting the `Type 5` approach. For example, our model could be used in combination with CORAS as an additional, availability-specific, technique to determine the consequences of threats, in substitution of the traditional HazOp, FTA and FMECA techniques.

We showed in the present case how we integrated the QualTD model with a `Type 1` method by means of a threat and vulnerability (aggregated) risk level definition table. We believe that this approach is applicable in general if it is time and information-wise feasible for the risk assessor to explicitly enumerate the vulnerabilities are present in the ToA.

Integration with a `Type 4` is instead more challenging, as it would require an approach similar to the one we described previously for OCTAVE.

Finally, RAs following `Type 2` and `Type 3` methods cannot be integrated with our model, due to the fact that `Type 2` methods already (implicitly) take into consideration the consequences of incident propagation in the definition of the security needs for each asset in the ToA, while `Type 3` methods are quantitative.

5.2 Dependency-based techniques for RA

Some academic researchers propose to use dependencies to improve the quality of security RAs. They have addressed this topic from multiple perspectives, such as information security, business administration and software engineering. In the literature of security RA we find three kinds of dependencies: security dependencies, software dependencies and organisational and technical/functional dependencies. In this section we will examine previous literature on these three fields which matches our work. Moreover, since our method considers the third kind of dependencies, in the final part of this section we also enumerate some techniques to build technical/functional dependency graphs.

Security dependencies Baiardi et al. [2] propose a framework for RA of information infrastructures by building a hyper-graph of security dependencies, i.e., dependencies on the security properties of the system: confidentiality, integrity and availability. The dependency graph is a form of attack graph in which nodes are the components of the infrastructure, and edges between nodes represent the dependency of a component on some security properties of the component it is linked to. Threats are represented as users of the infrastructures possessing some security properties on some contents, while vulnerabilities are conditions allowing the extension of security rights from one component to another. The framework allows one to rank countermeasures and create risk mitigation plans. A countermeasure can reduce the vulnerability level of a component, update dependencies, update the initial properties of a threat or increase the resources needed for an attack. Attack graph-based approaches are known to have scalability problems (e.g., see Lippmann et al. [21]) in terms of the number of hosts under assessment. This is due to the fact that building such graphs requires a large amount of work which can be only partially automated. Moreover, they require extensive and difficult to obtain attack details: this information was not available in the Oxygen RA and we believe it would not be readily available in most RAs. On the other hand, our approach is in principle less precise, but also works when attack details are limited, as the propagation of an availability incident is mostly dependent on the architecture of the ToA, and this information is in many cases readily available.

Software dependencies Goseva-Popstojanova et al. [10] present a semi-quantitative approach for assessing reliability and availability related risks at early phases of a software life cycle by using the UML representation of the ToA. In this work, the authors use dependencies between software components to assess the likelihood of a fault propagating from a component to the other. In more detail, they use the following UML constructs: software architecture diagrams, use case diagrams, sequence diagrams and state charts of software components. By means of this information, they estimate the probability of failure of a software component, and the probability of failure of two software components interacting with each other. They consider the complexity of a software component in order to calculate the probability of its failure, and the number of messages exchanged by components to determine the probability of an interaction failure. They give the impact of a failure in a qualitative scale ranging from *Minor* to *Catastrophic*. Then, they calculate the risk level distribution of each UML use case scenario by building a Markov model from the scenario sequence diagram. Finally, they average all the single use case risk distributions to determine the overall system risk. This approach, however, is not readily applicable to all IT RAs. First, it specifically targets the assessment of risks to software components, but it is less suitable to be used for a whole IT system which includes not only software but also hardware, network components and their interaction. Secondly, as threats only software and communication failures are taken into account. In the RA of a whole IT system one is interested in assessing incidents caused by other threats (e.g., DoS attacks) and this approach does not provide a way to do this. Finally, in the Oxygen case we did not have any UML representation of the ToA and no quantitative figures about the likelihood of threats.

Organisational and technical/functional dependencies Innerhofer-Oberperfler and Breu [13] propose a model-driven approach for assessing IT-related risks using an enterprise architecture as the basis of the model. They group entities of the enterprise architecture in four hierarchical layers: business, application, technical and physical layer. They derive—by refinement—business security objectives and requirements from this enterprise architecture and from the dependencies among its constituents. The refinement process follows a top-down approach starting from high-level business units to technical and physical devices. Then, they identify and analyse risks to the security requirements by selecting threats and vulnerabilities from standard security methods, e.g., BSI IT-Grundshutz [40]. Once risks are identified, they do a bottom-up aggregation of risk scenarios to make sure risks become clearly understandable at each level of the organisation (i.e., from technical to business levels). The approach is qualitative and not linked to a specific threat-list, with a risk analysis technique very similar to the one presented in the EBIOS [38] method. In our view, the strong point of this approach is that RA is fully embedded on the organisation at all levels, from the technical level to the business management level. On the other hand, it imposes that the whole organisation is aligned and has agreed on security requirements at all levels, before the assessment can be done. This is a strong assumption for normal enterprise organisations in which such a cooperation among the many business units and the IT department is hardly achieved. For example, in our case we had little information regarding the high-level goals of the organisation and the main difficulty in applying this method would have been deriving the full list of security requirements from (unknown) high level goals. Our approach on the other hand, only requires the business owner(s) to give a relative value to the different IT services involved in the RA, which is much easier to gather from business-oriented people.

Kim et al. [20] propose a model to assess and prioritise security risks and their treatment in the context of a communication infrastructure. They do this by determining the magnitude of damages produced by a threat to the assets of the ToA, also taking into account incident propagation. To model incident propagation, they use technical and functional dependencies among the assets of the communication infrastructure: for each threat they create a workflow (graph) of the incident propagation, with the assets as nodes and the relevant dependencies as edges. They annotate each edge with the probability that the destination node is affected by the damage on the source node. Finally, they model the vulnerability level of an asset by considering the “age” of the asset. Starting from the assumption that systems age over time, and because of the increased level of knowledge attackers gain on the weaknesses of the asset, attacks are supposed to have a greater probability of success over time if the system is not timely patched. Using incident propagation graphs and likelihood distribution functions, the authors are able to calculate the risk of an infrastructure over time, and to prioritise the actions to be taken to control those risks. This approach is substantially quantitative, and this makes it harder to apply due to lack of information: the data available for Oxygen was insufficient to estimate the level of weakness of the system over time. Moreover, it only considers the component's age to determine its vulnerability level, which is limiting in many situations. For example, according to the SLAs with the outsourcing company, patching is performed quite regularly on Oxygen; therefore, the weakness (vulnerability) level of the assets is almost constant.

Building a dependency graph Every technique using dependencies for RA is based on the possibility of constructing a dependency graph describing the ToA. Building the dependency graph is an“extra” step which is not required by traditional RA methods, as they mainly rely on the same information but in an implicit form. For this reason, building the dependency graph in a time-effective way is essential for the applicability of dependency-based RAs.

A technical/functional dependency graph can be built either manually or automatically. Manual methods involve acquiring information by functional and technical documentation and from interviews, like we did in the Oxygen RA. On the other hand, *Static Dependency Analysis* [19] and *Active Dependency Discovery* [4] are two automatic techniques to automatically create the graph.

The former method is based on using application configuration files to derive dependencies, e.g., the `web.xml` file for Java web applications. The main drawback of this method is that it does not generate a full, cross-domain dependency graph. This is due to the fact that some dependencies are never derivable from a configuration file, and to the high number of different formats configuration files can take.

The latter method consists of measuring the variation of certain QoS parameters (e.g., availability or response time) of the ToA after some of its components are deliberately perturbed. For example, by simulating a network traffic overload it is possible to measure the dependency of the response time of a software component with relation to the network service it relies on.

An example of an Active Dependency Discovery technique was proposed by Bagchi et al. [1] for the availability of e-commerce environments. The authors propose to inject faults on the test/benchmark environment of the ToA and detect availability dependencies; the same dependencies are then also assumed to hold on the production system. This technique allows one to quickly build a dependency graph without the need to know perfectly the implementation details of the ToA. However, to build a reliable dependency graph, the test/benchmark system must be identical to the production system, which is not the case for Oxygen.

6 Conclusions

In this paper we introduce the new QualTD model and technique for the qualitative assessment of availability risks based on the propagation of availability incidents in an IT architecture. We apply the model and technique to a real-world case by carrying out an RA on the authentication and authorisation system of a large multinational company. We compare the results of this RA with the ones obtained from a previous RA carried out internally by the company on the same system. We then evaluate the results with respect to the goals of the stakeholders of the system.

Our results show the feasibility of the QualTD model and technique, and indicate that the model provides better results in terms of accuracy in terms of impact estimates and reduces the number of subjective decisions taken by the risk assessor. The reasons of success are mainly due to the systematic nature of the approach and to the completeness of the information the model includes. These factors help the risk assessor to deal with the complexity of the ToA in such a way that no relevant risk factor is neglected. Our analysis also shows that the QualTD model is particularly suitable to assess the availability risks of IT infrastructures or parts of them, when RAs are carried out regularly on the same target and when the final results of the RA are used to prioritise the risk mitigation strategies.

In addition, we analyse 12 RA standard methods, and we discuss which characteristics of the standard methods are compatible with the QualTD model-based technique. Our analysis shows that the QualTD model can be used in combination with many of the most popular RA standard methods. This indicates a wide range of applicability of the technique, also in organisations not using the same RA method we used in this case.

Finally, we make a review of academic works we found in the literature which apply dependency analysis to RA. We show the type of risk analysis these techniques allow and we discuss their applicability to our real-world case. Our analysis shows that none of the techniques examined are directly applicable to our case either because they require information that was not readily available, or because they cannot satisfy the requirements of the stakeholders.

References

1. Bagchi, S., Kar, G., Hellerstein, J.: Dependency analysis in distributed systems using fault injection: application to problem determination in an e-commerce environment. In: DSOM '01: Proceedings of 2001 International Workshop on Distributed Systems: Operations & Management. http://www.research.ibm.com/PM/DSOM2001_dependency_final.pdf (2001)
2. Baiardi, F., Suin, S., Telmon, C., Pioli, M.: Assessing the risk of an information infrastructure through security dependencies. Crit. Inf. Infrastruct. Secur. **4347**, 42–54 (2006)
3. Bennet, S.P., Kailay, M.P.: An application of qualitative risk analysis to computer security for the commercial sector. In: Eighth Annual Computer Security Applications Conference, pp. 64–73. IEEE Computer Society Press. http://ieeexplore.ieee.org/xpls/abs_all.jsp?isnumber=5913&arnumber=228232&count=25&index=15, April 1992
4. Brown, A., Kar, G., Keller, A.: An active approach to characterizing dynamic dependencies for problem determination in a distributed application environment. In: IM '01: IEEE/IFIP International Symposium on Integrated Network Management, pp. 377–390 (2001)
5. BS 7799-3: Information Security Management Systems. Part 3: Guidelines for Information Security Risk Management (2006)
6. BSI: BS IEC 61882:2001: Hazard and Operability Studies (HAZOP studies). Application Guide. British Standards Institute (2001)
7. Cunningham, B., Dykstra, T., Fuller, E., Gatford, C., Gold, A., Hoagberg, M.P., Hubbard, A., Little, C., Manzuik, S., Miles, G., Morgan, C.F., Pfeil, K., Rogers, R., Schack, T., Snedaker, S.: The Best Damn IT Security Management Book Period. Syngress Publishing. November 2007
8. den Braber, F., Hogganvik, I., Lund, M.S., Stolen, K., Vraalsen, F.: Model-based security analysis in seven steps—a guided tour to the CORAS method. BT Technol. J. **25**(1), 101–117 (2007)
9. Evangelidis, A., Akomode, J., Taleb-Bendiab, A., Taylor, M.: Risk assessment & success factors for e-government in a UK establishment. In: Electronic Government, vol. 2456/2002, pp. 93–99. Springer, Berlin (2002)
10. Goseva-Popstojanova, K., Hassan, A., Guedem, A., Abdelmoez, W., Nassar, D.E.M., Ammar, H., Mili, A.: Architectural-level risk analysis using UML. IEEE Trans. Softw. Eng. **29**, 946–960 (2003)
11. Gunter, C.A., Gunter, E.L., Jackson, M.A., Zave, P.: A reference model for requirements and specifications. IEEE Softw. **17**(3), 37–43 (2000)
12. Herrmann, D.S.: Complete Guide to Security and Privacy Metrics. Auerbach Publications, Boston (2007)
13. Innerhofer-Oberperfler, F., Breu, R.: Using an enterprise architecture for IT risk management. In: ISSA '06: Proceedings of Information Security South Africa Conference. http://icsa.cs.up.ac.za/issa/2006/Proceedings/Full/115_Paper.pdf (2006)
14. ISO/IEC 13335:2001: Information Technology—Security Techniques. Guidelines for the management of IT security (2001)
15. ISO/IEC 15408:2006: Common Criteria for Information Technology Security Evaluation. http://www.commoncriteriaportal.org/thecc.html, September 2006
16. ISO/IEC 17799:2000: Information Security. Code of Practice for Information Security Management (2000)
17. ISO/IEC 27001:2005: Information Technology. Security Techniques: Information Security Management Systems—Requirements (2005)
18. ISO/IEC 27002:2005: Information Technology. Security Techniques: Code of Practice for Information Security Management (2005)
19. Kar, G., Keller, A., Calo, S.: Managing application services over service provider networks: architecture and dependency analysis. In: NOMS '00: Proceedings of the 7th IEEE/IFIP Network Operations and Management Symposium, pp. 61–75. IEEE Press (2000)
20. Kim, I.-J., Jung, Y.-J., Park, J.G., Won, D.: A study on security risk modeling over information and communication infrastructure. In: SAM '04: Proceedings of the International Conference on Security and Management, pp. 249–253. CSREA Press, June 2004
21. Lippmann, R.P., Ingols, K.W.: An annotated review of past papers on attack graphs. Technical report, Defense Technical Information Center OAI-PMH Repository. http://stinet.dtic.mil/oai/oai (United States), 1998. http://en.scientificcommons.org/18618950
22. Morali, A., Zambon, E., Houmb, S.H., Sallhammar, K., Etalle, S.: Extended eTVRA vs. security checklist: experiences in a value-Web. In: ICSE '09: Proceedings of the 31th IEEE International Conference on Software Engineering, IEEE. IEEE Computer Society Press (2009)
23. Muntz, R.R., de Souzae Silva, E., Goyal, A.: Bounding availability of repairable computer systems. SIGMETRICS Perform. Eval. Rev. **17**(1), 29–38 (1989)
24. Pawson, R., Tilley, N.: Realistic Evaluation. Sage Publications, Beverly Hills (1997)
25. Rossebo, J.E.Y., Cadzow, S., Sijben, P.: eTVRA, a threat, vulnerability and risk assessment method and tool for eEurope. In: ARES '07: Second International Conference on Availability, Reliability and Security, pp. 925–933. IEEE Computer Society Press. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4159893, April 2007
26. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. In: IEEE Symposium on Security and Privacy, p. 273 (2002)
27. Stoneburner, G., Goguen, A., Feringa, A.: NIST SP 800-30: Risk management guide for information technology systems. Technical report, NIST National Institute of Standards and Technology (2002)
28. Vesely, W.E., Goldberg, F.F., Roberts, N.H., Haasl, D.F.: Fault tree handbook. Technical report, US Nuclear Regulatory Commission NUREG-0492 (1981)
29. Wieringa, R.J., Heerkens, J.M.G.: Designing requirements engineering research. In: CERE '07: Workshop on Comparative Evaluation in Requirements Engineering, pp. 36–48. IEEE Computer Society Press. http://eprints.eemcs.utwente.nl/13002/, October 2007
30. Wieringa, R.J., Maiden, N., Mead, N., Rolland, C.: Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. Requir. Eng. J. **11**, 102–107 (2006)
31. Zambon, E., Bolzoni, D., Etalle, S., Salvato, M.: Model-based mitigation of availability risks. In: BDIM '07: Second IEEE/IFIP International Workshop on Business-Driven IT Management, Munich, pp. 75–83. IEEE Computer Society Press. May 2007

Web References (Last Accessed: May 2010)

32. Alberts, C.J., Dorofee, A.J.: OCTAVE criteria. Technical report ESC-TR-2001-016, Carnegie Mellon-Software Engineering Institute. http://www.cert.org/octave/, December 2001
33. Risk management: AS/NZS 4360:2004. http://www.riskmanagement.com.au/, October 2004
34. CISCO Systems: Cisco 2007 Annual Security Report. http://www.cisco.com/web/about/security/cspo/docs/Cisco2007Annual_Security_Report.pdf (2007)
35. CobiT 4.1: Control objectives for information and related technology. http://www.isaca.org (2007)

36. CRAMM v5.1 Information Security Toolkit. http://www.cramm.com (2009)
37. Deladrière, A., Morrison, M.: The risk management challenge. http://www.bankingfinance.be/40915/default.aspx, March 2008
38. EBIOS: Expression des Besoins et Identification des Objectifs de Sécurité. Section 2: Approach. http://www.ssi.gouv.fr/en/ (2004)
39. ENISA: Risk management: implementation principles and inventories for risk management/risk assessment methods and tools. Technical report, European Network and Information Security Agency (ENISA). http://www.enisa.europa.eu/rmra/rm_home.html, June 2006
40. BSI Standard 100-1: Information Security Management Systems (ISMS). http://www.bsi.de/english/gshb/ (2005)
41. McAfee: In the Crossfire—Critical Infrastructure in the Age of Cyber War. http://resources.mcafee.com/content/NACIPReport (2010)
42. MEHARI 2007: Risk analysis guide. http://www.clusif.asso.fr/en/clusif/present/, April 2007
43. NIST National Vulnerability Database. http://nvd.nist.gov/ (2009)
44. PriceWaterhouseCoopers: BERR Information Security Breaches Survey 2008. http://www.pwc.co.uk/pdf/BERR_ISBS_2008(sml).pdf (2008)
45. Sarbanes-Oxley Act of 2002: http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=107_cong_bills&docid=f:h3763enr.tst.pdf (2002)

Author Biographies

Emmanuele Zambon is a PhD candidate at the University of Twente since 2006, where he researches on Information Risk Management. He graduated in 2005 in Computer Science at the Cá Foscari University of Venice with a thesis on Intrusion Detection Systems. During and after his studies he has been employed in the Information Risk Management group of KPMG and in Telecom Italia as a consultant, doing ethical hacking, network vulnerability/assessment, and software development/performance tuning.

Sandro Etalle is professor of Embedded System Security at the Technical University of Eindhoven where he leads the chair of computer security, and part time professor at the University of Twente. In 2002 he started working on the protection of confidential data, a line of study which lead in 2004 to the definition of the first logic for accountability. His research interest include Trust Management, Policy Compliance, Intrusion Detection and Risk Management.

Roel J. Wieringa is professor of Information Systems at the University of Twente. His research interests lie in methods for software system requirements engineering and architecture design, and in the application of these methods to distribute information systems, varying from groupware to workflow and e-commerce systems. His research includes the development of advanced tools to support design methods, including tools to simulate and explore possible system designs. He has published books and over 50 papers on software design methods.

Pieter Hartel received his Master degree in Mathematics and Computer Science from the Free University of Amsterdam in 1978 and his PhD degree in Computer Science from the University of Amsterdam in 1989. He has worked at CERN in Geneva (Switzerland), the Universities of Nijmegen, Amsterdam (The Netherlands), and Southampton (UK). He is currently a full Professor of Computer Science at the University of Twente. His research interest is Information Security.