Risk-Based Confidentiality Requirements Specification for Outsourced IT Systems (extended version)

Ayşe Moralı
Distributed and Embedded Security Group
University of Twente
Enschede, The Netherlands
ayse.morali@utwente.nl

Roel Wieringa
Information Systems Group
University of Twente
Enschede, The Netherlands
roel.wieringa@utwente.nl

***Abstract*—Today, companies are required to be in control of their IT assets, and to provide proof of this in the form of independent IT audit reports. However, many companies have outsourced various parts of their IT systems to other companies, which potentially threatens the control they have of their IT assets. To provide proof of being in control of outsourced IT systems, the outsourcing client and outsourcing provider need a written service level agreement (SLA) that can be audited by an independent party.**

SLAs for availability and response time are common practice in business, but so far there is no practical method for specifying confidentiality requirements in an SLA. Specifying confidentiality requirements is hard because in contrast to availability and response time, confidentiality incidents cannot be monitored: attackers who breach confidentiality try to do this unobserved by both client and provider. In addition, providers usually do not want to reveal their own infrastructure to the client for monitoring or risk assessment.

Elsewhere, we have presented an architecture-based method for confidentiality risk assessment in IT outsourcing. In this paper, we adapt this method to confidentiality requirements specification, and present a case study to evaluate this new method.

***Keywords*-Confidentiality requirements; Outsourcing, Service level agreements; Risk assessment**

List of Abbreviations

SLA	Service Level Agreement
CRAC	Confidentiality Risk Assessment and Comparison
IFG	Information Flow Graph
APG	Attack Propagation Graph
CMA	Corporate Master Agreement
SA	Service Agreement
ERP	Enterprise Resource Planning
OMA	Outsourcing Master Agreement

I. Introduction

Current regulations, such as Basel II [2], SOX [25], ISO-17799 [13], and BDSG 42a [4], require companies to be in control of the security (i.e. confidentiality, integrity and availability) of their IT assets and to provide proof of this in the form of audit reports. In this paper we call this the *control requirement* and by implication the more detailed IT requirements derived from control requirements are also control requirements. Satisfying control requirements is perceived as not contributing to the company's products or services. Therefore, companies are always aiming at satisfying control requirements in the most cost-effective way.

Satisfaction of control requirements is further complicated because organizations outsource tasks that are not part of their core business, such as IT management, by which some of their IT is now actually under the control of *other* organizations. In this paper, we introduce and evaluate a method for identifying and specifying a particularly important control requirement in outsourcing, namely confidentiality of information. To note that here we also address privacy related information.

IT outsourcing requires connecting IT infrastructures of two organizations, and mutually giving access to this cross-organizational infrastructure. For example, some employees of the provider must be able to perform management services on the IT infrastructure of the client, and conversely some employees of the client must be able to grant or revoke permissions to the employees of the provider. In whatever way this is done, confidentiality risks arise that must be managed jointly [8]. Assessing the confidentiality risks of either organization requires knowledge of both organizations' IT infrastructure, and mitigation measures also often require actions in both infrastructures [11], [17]. However, this is challenging, because outsourcing providers are commonly large organizations that provide IT services to several customers and the confidentiality requirements of these customers deviate from each other. Furthermore, to maintain confidentiality and protect business secrets, and to satisfy their *own* control requirements, providers do not want to reveal more about their IT infrastructure than strictly necessary.

Providers usually show that they are trustworthy by

This research is supported by the research program Sentinels (http://www.sentinels.nl). Sentinels is being financed by Technology Foundation STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

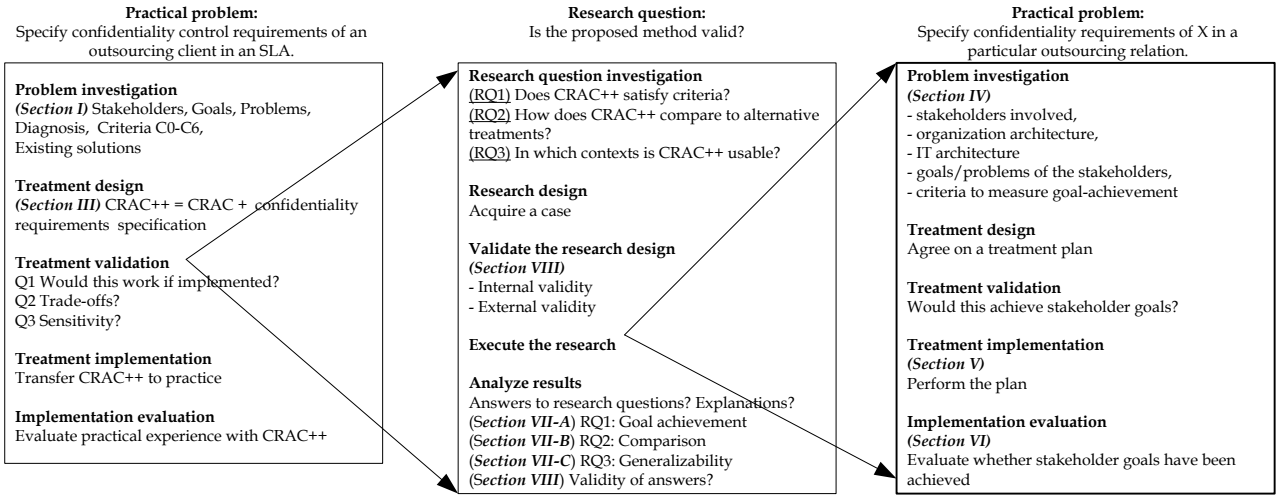

Figure 1. Structure of this research. The top-level problem is shown on the left.

showing compliance to regulations, e.g. SOX [25], and additionally by independent audits, who produce a report under Statement of Auditing Standards No. 70 Service Organizations (SAS 70). A client who thinks that these compliance reports alone are not enough, must additionally specify the content of the audits in the form of Service Level Agreements (SLAs) that define service-specific requirements.

An SLA is a mid- to long-term contract that specifies service quality levels for the outsourcing provider, and fines for failing to deliver these. SLAs usually specify quality levels for availability and response time, but so far in practice they do not specify quality levels for confidentiality. Yet today, outsourcing clients have to show that they satisfy the control requirement of treating their data confidentially, and so they now need to specify their confidentiality requirements in SLAs.

The problem with specifying confidentiality requirements in an SLA is that clients do not want to specify a quantified confidentiality level such as that on the average no more than 1% of the data will be lost per month. Furthermore, even if they would want to specify a confidentiality level like this, this attribute could not be monitored because typically, confidentiality incidents are not observable by the client, both because attackers keep their actions secret and because the provider would not allow any client to monitor the provider's IT infrastructure. So another approach to confidentiality level specification must be chosen, that satisfies at least three criteria that we have identified so far: (C1) it does not specify confidentiality levels as percentages of data loss; (C2) it is not based on monitoring incidents; and (C3) it does not require a provider to disclose confidential information.

Companies currently have checklists of confidentiality risks that they use to assess the risks of an outsourcing architecture. These checklists do not explicitly consider confidentiality, and they also do not provide sufficient insights in confidentiality risks to support negotiation with the outsourcing provider. Discussion with several companies taught us that the method should satisfy several criteria additional to the ones we mention above:

- C4 The method shall be usable with acceptable effort for the client. In particular, experienced risk assessors shall be able to use it without following a course and it shall not increase the time allowed for risk assessment. We call this criterion *ease of use*.
- C5 The method shall deliver results (confidentiality control requirements to be included in an SLA) that are independent of personal judgment by making less use of subjective estimates than the checklist based method. We call this criterion *repeatability*.
- C6 The method shall increase the client's understanding of confidentiality risks in this outsourcing relationship.

In this paper we propose and evaluate a method that meets these requirements sufficiently in the cases that we have investigated, and in a way that justifies the claim that it will meet these criteria in other cases too. It is based on specifying confidentiality requirements according to risk assessment results.

II. Research Method and Structure of the Paper

In this paper we follow a nested problem-solving approach as proposed earlier by Wieringa [26] (Figure 1). At the top level we have the practical problem of specifying confidentiality control requirements of an outsourcing client in an SLA. A *practical problem* is a difference between the real world and the way stakeholders would like it to be. To resolve it, some change must be applied to the real

world. In this paper we call this change a *treatment*.[1] In a rational problem solving cycle, the treatment is designed after an investigation of the problem and validated before implementation; and it is evaluated after implementation.

We have already presented our problem investigation in Section I. We will describe existing solutions in Section IX. In Section III we describe our treatment, which is an extension of the CRAC method [19] for assessing and comparing the confidentiality risks of IT architectures, by a step that specifies requirements for confidentiality risk mitigation measures. We call this extended method *CRAC++*.

Sections IV to VII describe a validation of CRAC++. This is the main topic of this paper.

The question whether a treatment is valid asks whether the treatment will have the desired effects. This is a research question. To answer a research question, we have to *do* something, and this is a new practical problem at a lower level of nesting (Figure 1, middle column). Standard treatment validation questions are what the effects of a treatment will be, and whether this will satisfy stakeholders' criteria (RQ1), how this compares to alternative treatments (RQ2) and whether this will work in other problem contexts too (RQ3). The middle column of Figure 1 shows a rational problem solving cycle in which the researcher investigates the research problem, designs research to answer the research questions, validates the research design, executes it and analyzes the results.

To validate a method, we eventually need a realistic context in which the method is applied. Applying it to a toy problem is fine for illustration, and testing in an experiment is good for improving our understanding of the method, but in order to know whether the method will work in practice, it has to be used in practice. This could be done by a field experiment, in which practitioners use the method to solve an experimental problem [24]. This is extremely expensive but not impossible. In our case, we opted for the more realistic option, given our budget, of using the method ourselves for a real world problem. In other words, we took an action research approach to validation [3].

We have acquired a case organization that needed to specify confidentiality requirements in an outsourcing relation (Section IV), and have used CRAC++ to specify confidentiality requirements that could be included in an outsourcing SLA (Section V). Analysis of this case allows us to find a first, approximate answer RQ1 (Section VII-A). Analyzing the mechanisms at work during our application of CRAC++ allows us also to assess generalizability (RQ3, (Section VII-C)), and comparison with what would happen when using other methods allows us to assess trade-offs with other treatments (RQ2, (Section VII-B)).

[1]Earlier we called it a solution [27] but this hides the fact that a treatment may not solve the problem completely but only bring the stakeholders closer to their goals, or may even make the problem worse, as when a doctor prescribes a wrong medicine.

The validity of our action research approach is the inverse of the risk that we are giving the wrong answers to our research questions. We discuss this risk in Section VIII.

III. CRAC++

The Confidentiality Risk Assessment and Comparison (CRAC) method [19] compares confidentiality risks of two alternative networked IT architectures by analyzing how information can flow through a network, and how unauthorized persons can move (get access to nodes) through the network. Possible information flow determines the information that can be present in a node of the network, and therefore allows us to assess the impact of a confidentiality breach (information disclosure) at that node. Analysis of possible movement of unauthorized persons through the network allows us to assess the risk of an unauthorized person accessing a node, causing a confidentiality breach. Combining this information allows us to assess the risk of confidentiality breach per node.

In CRAC++, we extend this method with a step to identify confidentiality requirements of the client that are *not* implied by the known confidentiality requirements of the provider, and which therefore are candidates for inclusion in an SLA with that provider. Because of the page limitation we could not include a formal description of the method here, but interested reader may refer to the technical report [18].

Step 0: Elicit Input Data

Relevant documents to consider are IT architecture specifications, existing SLAs, best practices, relevant recommendations, standards and laws that contain confidentiality control requirements, e.g. the NIST vulnerability list [20]. Relevant stakeholders may include the company's security officer, system architect, and security architect.

We use the following basic notation.

- V is the ordered set of all confidentiality values (e.g. {top-secret, confidential, public});
- N is the ordered set of all information asset quantity classes (e.g. {all, single, none});
- H is the ordered set of all information asset homogeneity classes (i.e. {homogeneous, non-homogeneous});
- I is the ordered set of all qualitative impact values (e.g. {high, medium, low, null});
- TI is the ordered set of all qualitative total impact values (e.g. {very-high, high, medium, low, null}); and
- E is the ordered scale of fractions between 0 and 1 indicating the ease of exploiting vulnerabilities and accessing the instances of information assets on a component.

At the end of this step the risk assessor has the following data, which is used in the following steps of the method:

Information assets: Functional or organizational data stored on the system components, and of value to the organization, such as user credentials, client data and functional specifications of the system. We classify these information assets based on their confidentiality-relevant properties, such as confidentiality value for the organization. Information assets are types that have instances. For example, if client data is an information asset, then each client record is an instance of this asset. A is the set of information assets we consider. To each information asset $a \in A$ we associate a confidentiality value $v : A \rightarrow V$. Furthermore, an information asset is homogeneous if the damage due to its disclosure can be considered proportional to the number of its instances that get disclosed. For example "social security numbers" are homogeneous, since the damage due to the loss of one hundred social security numbers is larger than the damage due to the loss of a single social security number. Conversely, an information asset is *non-homogeneous* if the damage due to the disclosure of one instance is as big as the damage of the disclosure of all instances. For example, if credentials of one user get disclosed, the damage to the company is the same as if credentials of 100 users with equal access writes would be disclosed. To model this we use the mapping $h : A \rightarrow H$.

IT architectural components: These can be hardware (servers, terminals, routers, USB-sticks, a physical location (e.g. buildings), software (e.g. applications, operating systems, firewalls), or a network location (e.g. a network segment), where instances of information assets may be present. C is the set of all IT architectural components. A component c can obtain multiple instances of a given information asset a. The mapping $n : A \times C \rightarrow N$ is a quantitative estimate of the number of instances of a that can be retrieved from component c at once. Note that there are components, such as a router, that have no instance that can be stored on them.

Threat agents: These are potential attackers (e.g. hackers) or people who may intentionally or accidentally access information assets that they are not authorized to access (e.g. malicious insiders or outsourcing providers). T is the set of all threat agents in the system. We classify threat agents based on their estimated capabilities, such as system knowledge and hacking skills.

Relevant vulnerabilities: A vulnerability is a condition of the IT infrastructure or its organization that facilitates confidentiality attacks on architectural components. For instance if "reuse of storage media without proper erasure" is a vulnerability a threat agent may exploit it to access information stored on an external hard disk. Furthermore, we call vulnerabilities that need to be mitigated according to the confidentiality requirements of the outsourcing client *relevant vulnerabilities*. V is the set of all relevant vulnerabilities in the system. We represent that v is a condition of the IT architectural component c, or facilitates attacks on that component, by means of the mapping $w : V \times C \rightarrow \{true, false\}$.

Confidentiality requirements: We make lists of both the confidentiality requirements of the outsourcing client and those of the outsourcing provider. R is the set of all confidentiality requirements of the outsourcing client and outsourcing provider.

Step 1: Assessing Total Impact of Disclosure per Component

First, for each information asset and each component that the asset can reside on, we make an *Information Flow Graph* (IFG) that shows how this information can flow through the network (Figure 2-(a)). An $IFG = \langle C, ED \rangle$ is a directed and rooted graph in which C is the set of vertices representing IT architectural components and ED is the set of all edges $ED \subseteq C \times C$. $(c_1, c_2) \in ED$ iff there exists a physical or logical connection between c_1 and c_2 such that instances of an information asset can flow. For an information asset a the vertices of IFG_a represent IT architectural components from which instances of a can be accessed by a threat agent. The root of an IFG is an information source, such as a database. To note that each component c can appear only once in an IFG. We use the maximum number of instances that may flow to a component c from its connected components to determine the number of instances an attacker can access by gaining access to c. The combination of IFGs tells us which information can be present in an architectural component.

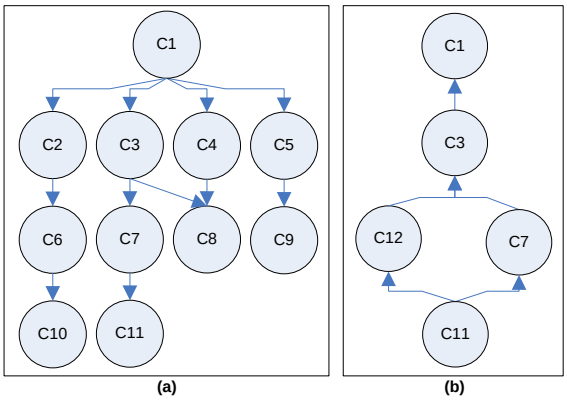

Figure 2. (a) Information Flow Graph; (b) Attack Propagation Graph

Confidentiality expert with system architect determine the IFGs by analyzing desired or undesired retrieval of instances of each information asset. For instance if the IFG represents the flow of client data, a client record could flow from the client database over the network to the terminal PC.

Then, we determine for each a, c and *ifp*, the impact of the disclosure of the instances of a which are present in c according to *ifp* using the function (*ifp-imp* : $A \times C \times IFP_a \rightarrow I$). To this end, we take into consideration the number of instances of an information asset which can be extracted to

a component at once, as well as their confidentiality level and homogeneity. Recall that $l(a)$ and $h(a)$ are respectively the confidentiality level and the homogeneity property of information asset a and that $n(a, c)$ is the maximum number of instances of a that can be extracted from c.

$$\textit{ifp-imp}(a, c, \textit{ifp}) = \begin{cases} l(a) \odot n(a, c, \textit{ifp}) & \text{, if } h(a) = \textit{homogeneous;} \\ l(a) \odot \textit{all} & \text{, if } n(a, c, \textit{ifp}) \neq \textit{none;} \\ \textit{null} & \text{, else.} \end{cases}$$

Where $\odot : L \times N \rightarrow I$ is a monotone composition operator for ordinal scale qualitative values in L and N. $\odot$ should be agreed on with the risk assessment stakeholders to guarantee that everybody understands how values are composed.

Now, we are able to compute the *impact* of the disclosure of each information asset a on each component c, *imp* : $A \times C \rightarrow I$, as the maximum impact with respect to all the possible flow paths in FP_a. We determine impact according to the following equation:

$$imp(c, a) = \textit{max}_{\textit{ifp} \in IFP_a} \textit{Ifp-imp}(a, c, \textit{ifp}) \quad (1)$$

Next, for each component, confidentiality expert together with the security officer assess the total value of information on the component. We call this value *total impact*, because it indicates the impact of disclosing all information assets on a component c. If c contains only one information asset a, then $imp(c) = imp(c, a)$. On the other hand, if c contains two or more assets (say a_1 and a_2) then we "add" $imp(c, a_1)$ and $imp(c, a_2)$. To this end we use the monotone operator $\oplus : TI \times I \rightarrow TI$ (as for $\odot$, $\oplus$ shall be agreed on with the stakeholders). Since we do not have a ratio scale of information value, this "addition" is not a summation operator but a way of expressing the opinion of security officer about the combined value of all information that can reside on a component. Since we do not have a ratio scale of information value, this "addition" is not a summation operator but a way of expressing the opinion of security officer about the combined value of all information that can reside on a component. More formally, the total impact of c is given in the following definition. Given a component c and a set of information assets A, we call *total impact of* c expressed by the function *imp* : $C \rightarrow TI$ the cumulative loss caused by the disclosure of all confidential information available on c. $imp(c)$ is given by the following equation:

$$imp(c) = \oplus_{a \in A} imp(c, a) \quad (2)$$

In the real-world cases that we have done so far, it is not known by the security officer what the monetary value of each information asset of the company is. Security officers prefer to assess the confidentiality value in terms of ordered non-ratio values such as *very high – high – medium – low – very low*. Therefore, in each real-world case, together with the security officer the confidentiality expert have defined a qualitative summation operator that allows us to "add up" the confidentiality value of the information asset that can be in the component, which is therefore the estimation, by security officer, of the total impact of information disclosure per component.

At the end of this step we identify the components for which unauthorized access would create a total impact higher than a certain value (criticality threshold) that is determined by system owners. We call these components *confidentiality-critical components*.

Step 2: Assessing Protection Level per Component

Having assessed the total impact of information per component, we must now assess the likelihood that a component will be accessed by an unauthorized agent. Frequencies of access by unauthorized agents are not available, so we cannot assess this likelihood numerically. Instead, confidentiality expert will assess, with the security officer and security architect, the protection level of each component for each class of threat agent and will use this to estimate the risk of information disclosure per component.

Ease of exploiting vulnerabilities: For this the confidentiality expert together with the security officer and security architect, assess for each threat agent the *ease* of exploiting vulnerabilities, based on the agent's capabilities. This assessment only depends on an assessment of the agent's capabilities and of vulnerabilities, and does not require knowledge of the IT architecture. The absolute numerical value of the fractions expressing ease has no meaning, but their relative ordering expresses the experts' opinion about which exploit is usually more difficult for a threat agent. We assume here that these opinions can be totally ordered. We represent the ease that a threat agent t exploits a vulnerability v by the mapping $e : T \times V \rightarrow E$

Ease of accessing one component: Together with security architect the confidentiality expert assess the vulnerabilities of each component, and the effectiveness of any preventive measures taken for these vulnerabilities per component. Measures are technical or policy leveltools that mitigate vulnerabilities of components. M is the set of all measures in the system. We represent the fact that m is a measure in the system. We represent the fact that m is a measure thaat mitigates vulnerability of v of component c by means of the mapping $m : V \times C \rightarrow \{true, false\}$. This is then combined with the previous analysis of ease of exploiting vulnerabilities by a threat agent. This provides us with an assessment of the ease of accessing a component for each threat agent. Again, the ease is qualitatively expressed in terms of a totally ordered set of values. We represent the ease of a threat agent t accessing a component c by the mapping $e : T \times C \rightarrow E$.

Protection level of component in network: If there were only one component in the network we would be done after assessing the ease of accessing components. However, each component is part of the architecture and an attacker can

take many paths through the network. To analyze the effect of this on each component, for each threat agent we make an *Attack Propagation Graph* (APG) that represents all paths the attacker can take to a valuable node. An APG is a finite directed graph with one or more terminal nodes (nodes without outgoing edges). The nodes represent components of the system and the edges represent attack steps (Figure 2-(b)). The edges are annotated with the ease of this step for the attacker. Similar to IFGs, we say a attach propagation graph $APG = \langle C, ED \rangle$ in which C is the set of vertices representing IT architectural components and ED is the set of edges, $ED \subseteq C \times C$. $(c_1, c_2) \in ED$ iff there exists a physical or logical connection between c_1 and c_2 such that a threat agent who has access to c_1 can gain access to the instances of information assets on c_2 or bypass c_2.

We construct an APG by first drawing nodes representing the entry points of the system and then gradually connecting further components by considering all possible propagations, until we reach a component that contains all instances of an information asset. These are the terminal nodes, because we assume that the threat agent will be satisfied when he reaches these nodes, either because this was his goal or because he is pleasantly surprised by what he finds there.

For each path from an entry node to a terminal node, we define the *bottleneck* of the path as the node that is hardest to access for a threat agent t. We represent a path in the APG for the threat agent apg_t by an ordered list $p = [c_1, ..., c_n]$ where $c_1 ... C_n \in C$ with no repeated occurrences of c_i. We call $P_t = \{p_1, ..., p_m\}$ the set of paths a threat agent t can follow. The bottleneck may cause the threat agent t to stop pursuing this path. For each terminal node c, we then select the path with the easiest bottleneck. The ease of access of this bottleneck is then by definition the protection level of c against this threat agent. Finally, for all APGs in which c is a terminal node, we define the easiest bottleneck as the *protection level* of c. Given a terminal node c, the protection level of this component $pl : C \rightarrow E$ is given by the ease of the easiest bottleneck with respect to bottlenecks of all the possible paths, as follows:

$$pl(c) = max_{t \in T}(min_{p \in P_t}(e(t, c))) \quad (3)$$

Components with low protection levels need attention. In particular, the outsourcing client may want to require the provider to increase the protection level of this component.

Step 3: Determining Candidate Confidentiality Requirements

In this step we identify confidentiality requirements of the client that are not implied by known confidentiality requirements of the provider. Those identified requirements that affect the ease of exploiting vulnerabilities of confidentiality-critical components are candidate requirements to be included in an SLA.

First, we identify vulnerabilities against which the client wants to defend itself. For this we identify the outsourcing client's confidentiality requirements that are not implied by known confidentiality requirements of the provider. We assume that the related vulnerabilities are not mitigated by measures of the provider and call these *unmitigated vulnerabilities*.

In Step 2 we have identified protection levels under the assumption that the clients confidentiality requirements were satisfied. Now, we have two scenarios: Either the client asks the provider to step up its own confidentiality requirements so that all of the unmitigated vulnerabilities will be mitigated sufficiently (*best case*), or we do nothing (*worst case*). Step 2 has already been done for the best case, so now we redo it for the worst case.

Finally, the confidentiality expert compares the protection levels of critical components in the best and worst cases and identifies the confidentiality requirements that the provider must satisfy. These could be all of the requirements needed to realize the best case. More realistically, the security officer have to deal with finite budgets, and it is not possible to realize the best case. Each additional requirement in the SLAs will increase the cost of outsourcing, and from this point on, confidentiality requirements specification will be a negotiation between client and provider. CRAC++ has provided the information security officer of the client with sufficient architectural information to conduct these negotiations, namely by allowing him to reason about what would happen if a requirement is included or dropped from the SLA. CRAC++ is therefore a method to support decisions about confidentiality requirements.

IV. The Case: Problem Investigation

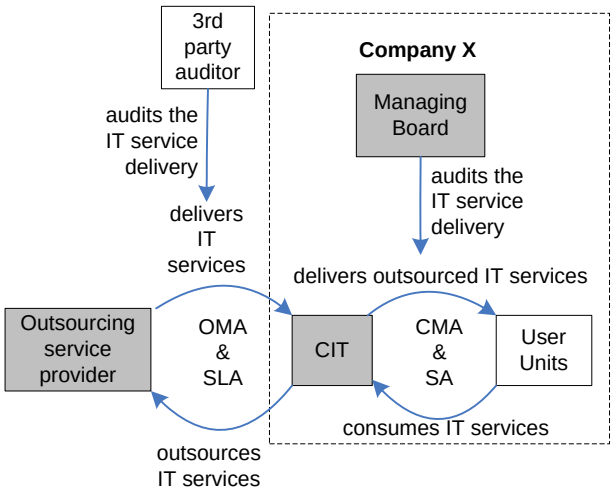

Figure 3. Stakeholders and their inter-relations w.r.t. the action case. Boxes represent stakeholders, arrows represent business relations, the dashed box indicates the boundary of X. Gray boxes are the stakeholders whose confidentiality goals affect the content of the OMA and SLA.

A. Stakeholders

The case to be described is a large multinational industrial company, which we refer to as X, with a total of 23,500 employees and divisions in 49 countries. Figure 3 shows the outsourcing relation that we will consider.

CIT is a competency center of X responsible for providing information and communication services to the system units. The confidentiality requirements of these services are defined in a Corporate Master Agreement (CMA) and if necessary detailed by Service Agreements (SA). The CMA contains control objectives that are extracted from the corporate rules. A *control objective* is a measure that indicates fulfillment of a control requirement. For example, the control requirement

> "The organization's approach to managing information confidentiality and its implementation shall be reviewed independently at planned intervals ..."

is operationalized in the SA by the control objective

> "CIT shall provide yearly a compliance statement ... declaring compliancy to corporate regulations on confidentiality of service providing as contracted. ..."

The Managing Board of X is responsible for managing and protecting the benefits of all competency centers. Competency center managers report yearly to the Managing Board on the fulfillment of the requirements in the CMA.

One set of services provided by CIT to users in X is Enterprise Resource Planning (ERP). Furthermore, CIT has outsourced ERP data center hosting services to an outsourcing provider. An Outsourcing Master Agreement (OMA) describes the quality attributes of the services that the outsourcing provider delivers to CIT and an SLA details the case specific requirements. There is one rule in the OMA that describes the confidentiality-related quality attributes:

> "In protecting Confidential Information, [Provider] will take all necessary precautions and the confidential information will be treated in the same manner and with the same degree of care as [Provider] applies with respect to its own confidential information. [Provider] shall keep all Confidential Information disclosed to it by X and [further clients of the Provider] in secure places, under strict access and use restrictions."

The current SLA between the outsourcing provider and client contains no confidentiality requirements.

B. IT architecture

The ERP system and the hardware that it runs on are owned by CIT but most of it is located in data centers owned by the provider. The ERP database contains four business confidential information assets:

- *Application information* is the business-related information of X, e.g. customer records and product prices.
- *Functional information* is monitoring-related information, e.g. access logs and IDS rule set.
- *User information* is information on the system users, e.g. roles and credentials.
- *Technical information* is the IT infrastructure-related information of the service, e.g. tunnelling data, of the ERP environment.

Figure 4 shows the architecture of the outsourcing IT infrastructure. Components with the same functionality that are located in the same network segment, e.g. employee PC (CPC), are represented by a single symbol. The firewalls between network segments provide IP-based access control. The gray rectangles represent physical buildings owned by the provider or by X.

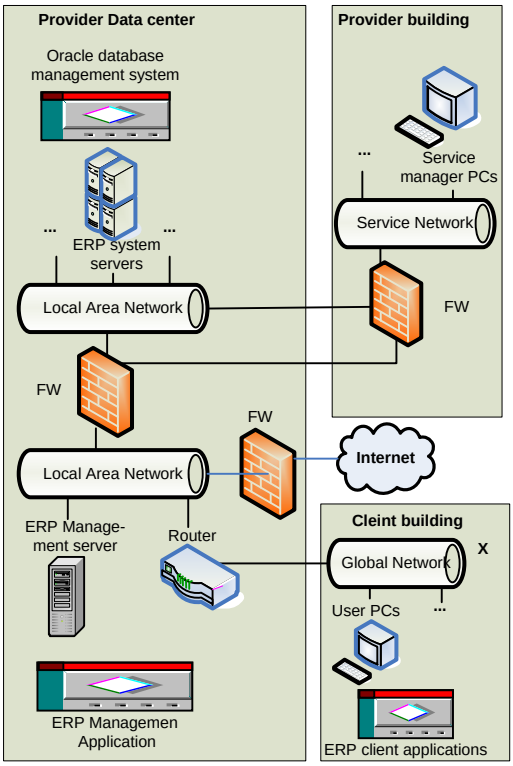

Figure 4. The IT infrastructure that supports the services in the scope of the case study.

C. Stakeholder Goals

Table I summarizes the stakeholder goals, and obstacles to goal achievement. (Our use of goals and obstacles is similar to KAOS [7].) Our aim is to find the confidentiality requirements that will help X to mitigate the effect of these obstacles.

As a response to recent changes in governance requirements, the Managing Board aims to be compliant with Corporate Governance Code (G1). One consequence of this is that corporate units such as CIT have become responsible for the quality of the services that they deliver to users in the company. To audit this, the Managing Board requires the corporate unit managers to periodically present reports on the quality of the services that they deliver to the System Users. However, the outsourcing provider does not allow

Table I
CONFIDENTIALITY GOALS AND PROBLEMS.

Stakeholders	Goals	Obstacles
Managing board	**(G1)** To be compliant with Corporate Governance Code.	**(O1)** The provider does not give direct insight into confidentiality of its systems to X.
CIT	**(G2)** To deliver user units of X CIT services that are compliant with CIT confidentiality requirements.	**(O2)** The SLAs and the OMAs do not contain confidentiality indicators, therefore it cannot be measured how well systems of the outsourcing provider meet the confidentiality requirements of the X.
Outsourcing provider	**(G3)** To deliver ERP Data Center Hosting Services as specified in the OMA and SLA and convince X that the confidentiality level of the services they deliver is enough for the requirements of X. **(G4)** To remain compliant with SOX [25] and SAS70 [1].	**(O3)** Confidentiality requirements are changing dynamically. **(O4)** Outsourcing clients are not communicating their confidentiality requirements clearly

CIT to directly analyze the confidentiality properties of the ERP system. The provider periodically delivers third-party audit reports based on their own confidentiality requirements to CIT (O1). This is an obstacle to G1 because these audit reports do not reflect the confidentiality requirements of X but of the provider.

CIT aims to deliver the system users ERP services that are compliant with the corporate requirements of X (G2). However, the OMA and the SLA that specifies the ERP Data Center Hosting Service originate from a period before the new Governance Code, and therefore they do not cover the confidentiality requirements that follow from this code (O2).

The provider aims to deliver ERP Data Center Hosting Services as specified in the OMA and SLA and convince X that his confidentiality baselines satisfy the confidentiality requirements of X (G3); but also to remain compliant with SOX [25] and SAS70 [1] (G4).

The provider has difficulty keeping track of the technical changes related to delivered services and changing confidentiality requirements of its customers (O3). The provider says that this is because the outsourcing clients are not communicating their confidentiality requirements clearly (O4).

V. THE CASE: APPLYING CRAC++

Together with company X, we made a plan for applying CRAC++ and validated the plan with the decision makers to check whether this would help them reach their goals (treatment design and validation in the lowest-level cycle of Figure 1). After obtaining approval, we executed the plan. Here, we briefly report on the results.

Step 0: Eliciting Input Data

At the end of this step we obtained the following information:

- a list of information assets (Application Information, Functional Information, User Information, and Technical Information) and their confidentiality values in a range of low to high;
- a list of components of the IT infrastructure of the system (basically, Figure 4);
- a list of confidentiality requirements and control objectives of the outsourcing client and a list of control objectives of the outsourcing provider;
- a list of known relevant vulnerabilities (Some of these vulnerabilities are Unprotected Communication Lines, Possibility To Access The Applications Remotely, Weak Authentication and Inadequate Patch Management Process); and
- a classification of possible threat agents (Insider, Malicious Insider, Outsourcer, Subcontractor, and Outsider) and a classification of their competencies (Physical Access, System Knowledge, Technical Knowledge, Social Knowledge, Social Hacking Skills, Hacking Skills, and Motivation To Damage).

Step 1: Aggregating Total Impact

We produced four IFGs, one for each information asset, and the total information disclosure impacts of components composing them. For instance, the ORACLE Server component is in the IFGs modeling the flow of Application Information (with confidentiality value high), Functional Information (with confidentiality value low), User Information (with confidentiality value *high*) and Technical Information (with confidentiality value *low*). We determine that the total impact of ORACLE Server as *very high* by aggregating the confidentiality values of instances of these information assets that are stored on ORACLE Server.

All in all, we identify that 27% of the components have *very high* total impact, 20% of the components have *high* total impact, 7% of the components have *low* total impact and the rest of the components have *null* impact. CIT set the criticality threshold as medium. Accordingly we say that 47% of the components are confidentiality critical.

Step 2: Assessing Protection Levels

We constructed five APGs, one for each threat agent, and assessed the protection levels of components that comprise them. For instance the terminal node ORACLE Server is in the APG for Insider, Malicious Insider, Outsourcer, and Subcontractor with the respective protection levels $1/6, 1/2, 4/9$

and $1/15$. Consequently we define the protection level of ORACLE server as $1/2$, which indicates the easiest exploit.

Step 3: Determining Candidate Confidentiality Requirements

In our case we did not have access to the list of confidentiality requirements of the provider but we did have access to his control objectives, which operationalize providers confidentiality requirements. We therefore first specified the control objectives of the client that are related with his confidentiality requirements. Then we checked which of these were *not* implied by control objectives of the provider. There were nine of those. We then assessed the protection levels for the critical components mentioned in these objectives in the worst case and found that 20% of the critical components are affected by at least one of those nine objectives. Vulnerabilities of these critical components (unmitigated vulnerabilities) can be mitigated by adding control objectives that mitigate these vulnerabilities to the SLA.

For example, the requirement of X "Removal of Property" is operationalized by the control objective

> "All items of equipment containing storage media shall be checked to ensure that any sensitive data and licensed software has been removed or securely overwritten prior to disposal."

This is not implied by any control objective of the provider and so "Use of removable media is allowed" is one of the unmitigated vulnerabilities. It can be exploited by a threat agent to access the component ORACLE Server. According to the worst case scenario we determined that the protection level of ORACLE Server is $9/18$. In the best case the protection level of ORACLE Server is $8/18$. The outsourcing client may now use this information as an argument to include "Removal of Property" in the SLA.

VI. Evaluation of Stakeholder Goal Achievement

Evaluation of achievement of stakeholder goals with the security officer of X and a representative of the outsourcing provider led to the following conclusions.

G1: By applying CRAC++, the necessary control requirements can be included in the SLAs. Consequently the audit reports of CIT to Management can improve their compliance to Corporate Governance Code.

G2: Including in the SLA confidentiality requirements that are currently not satisfied by the provider allows CIT to provide services to units of X that comply with CIT confidentiality requirements.

G3: The provider cannot be held accountable for requirements not stated in the OMA and SLA, which takes away O3. Furthermore, since the necessary control requirements are a part of the new SLA, the provider is able to convince X that the confidentiality level of the services he delivers satisfies the requirements of X, which also takes away O4.

G4: CRAC++ does not require the provider to disclose any confidential information to the risk assessor or to X, that he is not currently sharing. In return for this, the provider must implement further confidentiality controls as specified in the new SLA; these do not negatively affect the provider's compliance to SOX or SAS70.

VII. Answering the Research Questions

A. RQ1: Does CRAC++ satisfy the criteria?

(C1) Confidentiality level specified as percentage of data disclosure: In Step 1, CRAC++ uses an estimation of the relative value of disclosure of instances of information assets to determine the impact and total impact of components. Furthermore, in Step 2, we estimate the relative protection level of terminal nodes to determine the ease of disclosing instances of information assets. We use these in Step 3 to determine confidentiality levels, not to define levels of "acceptable" percentages of information disclosure. Therefore, CRAC++ satisfies C1.

(C2) Incident monitoring: CRAC++ does not depend on monitoring incidents but on domain-specific knowledge of security officer of the capabilities of threat agents and the presence of vulnerabilities in components.

(C3) Not disclosing confidential system information: To identify the vulnerabilities of components and compare protection levels in best and worst cases, only shared knowledge about the architecture of the IT infrastructure of outsourcing provider is used.

(C4) Ease of use: In the field study the first author needed one week to understand X's problem (lowest level problem investigation in Figure 1) and conduct Step 0 of CRAC++, and one additional week for developing the spread sheets and executing the other steps. X has so far no experience with confidentiality risk assessments, but our practical experiences show that checklist based security risk assessments for a system with a similar size take commonly one to two weeks. Also, the security officer of X said that the steps and results were easily understood and if tool support is provided then they could be able to use it. For further evaluation of satisfaction of C4, we are planning to conduct a usability study.

(C5) Repeatability: We estimated the repeatability by comparing the meta models of CRAC++ with the check list based approach. For this we analyze the concepts they use with respect to whether they require subjective estimation. To note that for this purpose we have excluded the concept "risk" from the check-list based approach (because CRAC++ does not aim to present risk) and the concept "unmitigated vulnerabilities" from CRAC++ (because check-list based approach does not aim to elicit requirements). All in all, the meta model of CRAC++ contains 20 concepts, of which 15, such as "component" or "threat agent", can be observed objectively or at the meta level so that all stakeholders agree and all risk assessors uniquely use. 15% of the concepts,

such as vulnerabilities of a component or how to form an APGs, are subjective. On the other hand 67% of the concepts that the check-list based approach contains require subjective opinion of the risk assessor, such as how well a certain measure mitigates a certain threat. We consider this an indication that the method is less subjective than the check-list based approach. Interested readers may find a comparison of the meta concepts of CRAC++ and checklist based approach in the Appendix.

(C6) Increased understanding: After applying CRAC++ to the case CIT reported increased understanding of the effects of confidentiality requirements on the confidentiality levels of the components and was able to prioritize them according to the impact of incidents.

B. RQ2: How does CRAC++ compare to alternative treatments?

As an alternative treatment to achieving G2, X suggested to monitor the outsourced IT systems with a Security Incident and Event Monitoring (SIEM) tool. However, SIEM tools generate logs with confidential data and possibly increases the criticality of components, so they increase the confidentiality risks for X. And they also would require the provider to disclose confidential information, which violates C2.

As another alternative treatment to achieve G2, X executed a third party audit based on the control objectives of CIT. However, this treatment did not succeed either. Although the audit report indicated some incompliance, X did not have a mechanism to enforce the provider to implement measures. Furthermore, the control objectives of X were not linked to risks. So, X also did not have an identification of how to mitigate the risk by applying measures on the part of the system that he has control over.

In [19] we compare the subjectivity of CRAC to that of other risk assessment methods. There, we showed that the CRAC method is more repeatable than CRAMM [5] and checklist-based risk assessment. For instance, if the risk assessment would be conducted with the CRAMM-method, then in total 76% of the variables would be non-subjective. So we conclude that CRAC++ is more repeatable than assessing confidentiality risks with CRAMM and specifying control objectives as we described in Step 3 of CRAC++.

C. RQ3: In which contexts is CRAC++ usable?

CRAC++ makes a number of assumptions about its context of use. These assumptions govern its reusability in different contexts. We assume (A1) that the provider does have confidentiality control objects and that the provider satisfies these—the CRAC++ method does not contain a step to check this. Furthermore, CRAC++ does not assume that the provider discloses confidential information or that the client has quantified the value of information assets or the likelihood of unauthorized access per component. By implication, (A2) we do assume that there are security officers who have informed opinions about this, and the method then helps in drawing conclusions from these opinions.

Large outsourcing providers are subject to control requirements and will satisfy A1. Large outsourcing clients with a security staff and chief security officer will satisfy A2.

So far, we have applied CRAC++ twice, both in multinational industrial companies where confidentiality was not a critical requirement until external regulators enforced it. Operating in highly competitive markets, these companies are very cost-sensitive and they will therefore not aim at maximum confidentiality. This might well be different in privacy-sensitive organizations such as health care or insurance companies, or in high confidentiality organizations such as the military. We do point out though that the qualitative assessments in CRAC++ could be replaced by more quantitatively informed techniques without changing the overall logic of the method. Nevertheless, as a third assumption for use we hypothesize that (A3) in the context of use, confidentiality is not the highest-priority requirement.

All of this indicates reusability to any context that satisfies the three assumptions.

VIII. Threats to Validity

In the previous section we proposed answers to three questions relevant to the validation of CRAC++. Now we must consider the validity of these answers themselves: What is the likelihood that we answered the questions incorrectly? The higher this likelihood is, the lower the validity of our answers.

Answering RQ1, we found that CRAC++ satisfies C1, C2, C3, C5 and C6; analyzing the reasons for these answers we find no reasoning errors or observational mistakes so we claim these answers are valid. C4 could not really be checked, since the user of the method (Morali) is also the inventor of the method. More systematic usability studies would require tool support, which currently is absent.

CIT reported increased understanding (C6), but we did not apply a formal test (e.g. an exam) to test this, nor did we analyze to which extent this is due to CRAC++.

The comparison with other approaches (RQ2) does not introduce new threats to validity that we can think of.

We answered the reusability question (RQ3) by identifying the conditions under which CRAC++ can be used, and actually showing that it could be used in another case satisfying these assumptions. Like all inductive conclusions, our conclusion that CRAC++ can be used in other cases is uncertain, but because we used analytic reasoning rather than statistical reasoning, we cannot quantify this uncertainty. In any case, our generalization claim shall be subjected to further tests by applying CRAC++ to other cases that satisfy the assumptions.

IX. Related Work

Several methods have been proposed for managing security when outsourcing IT management [12], [15], [21], [23]. Data Protection Agreement (DPA) [22] specifies what a provider may and may not do with the client's data. CRAC++ can be used to identify relevant confidentiality requirements for a DPA. Insurance Contracts (IC) [8] defines security requirements based on past incidents, which, for confidentiality, is not realistic. Protection Level Agreement (PLA) [14] specifies metrics to define protection levels. This can be used in combination with CRAC++.

Haley et al. [10] define a method for defining security requirements as constraints on functional requirements. This differs from CRAC++ because we focus on confidentiality, which is independent of functional requirements of the system and serve the control objectives that are imposed by regulators. We do make explicit trust assumptions as Haley et al. [9] do, because we assume that the provider can be trusted to satisfy its own control requirements.

A further tool that the organizations use to present that they are in control of the security of the products they deliver is Common Criteria [6] evaluation. However these evaluations consist of merely comparing two sets of requirements and do not enforce true verification and assure effectiveness and correct implementation of requirements. Due to its IT-architecture centered character, CRAC++ provides traceability between the requirements and the security features of the system components. Thus it assures effectiveness and correctness of requirements. Furthermore, in case of changes in the system components, it allows easily updating the evaluation results. Mellado et al. [16] introduce a security requirements engineering method that is based on reusing the results of previous evaluations. However, they express the accuracy and veracity of requirements in terms of incident propagations.

X. Conclusions and Future Work

This paper is based on two ideas: (1) Confidentiality requirements specification cannot be based on incidents, but must be based on an assessment of the risk of disclosure of confidential information; (2) Requirements specification in an outsourcing relation is budget-constrained. It is not feasible to simply list all confidentiality requirements of the client in an SLA with the provider: The outsourcing relation would then become too expensive, and possibly the provider cannot even satisfy all requirements because it has also to satisfy other, possibly conflicting requirements. So the client and provider must negotiate about the confidentiality requirements to be included in the SLA, based on a risk assessment of the most critical components of the outsourcing infrastructure.

Our case studies and analysis so far indicates that CRAC++ can satisfy our criteria (C1-C6), but satisfaction of some criteria such as ease of use and repeatability need further research. Currently, CRAC++ has been applied using a series of linked spreadsheets. To allow testing usability and repeatability, future work will include the development of tool support for CRAC++.

References

[1] American Inst. of Cert. Publ. Accountants. Auditing Standards Board. Statement on auditing standards; 70 Auditing Standards (SAS70), 2008. http://umiss.lib.olemiss.edu:82/record=b1038093.

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

[3] R. Baskerville. Distinguishing action research from participative case studies. *J. of Syst. and Info. Techn.*, 1(1):25–45, March 1997.

[4] Bundesdatenschutzgesetz: §42a Informationspflicht bei unrechtmiger Kenntniserlangung von Daten. http://bundesrecht.juris.de/bdsg_1990/__42a.html.

[5] British Government's Central Computer and Telecommunications Agency. CRAMM: Risk Analysis and Management methodology, 2008.

[6] ISO 15408:2007 Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 2, CCMB-2007-09-001, CCMB-2007-09-002 and CCMB-2007-09-003, September 2007.

[7] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Sci. Comput. Program.*, 20(1-2):3–50, 1993.

[8] S. Gritzalis, A. Yannacopoulos, C. Lambrinoudakis, P. Hatzopoulos, and S.K.Katsikas. A probabilistic model for optimal insurance contracts against security risks and privacy violation in it outsourcing environments. *Int. Journal of Information Security*, 6(4):197–211, 2007.

[9] B. Haley, C. Laney, D. Moffett, and B. Nuseibeh. Using trust assumptions with security requirements. *Requir. Eng.*, 11(2):138–151, 2006.

[10] C. Haley, R. Laney, J. Moffett, and B. Nuseibeh. Security requirements engineering: A framework for representation and analysis. *IEEE Trans. Softw. Eng.*, 34(1):133–153, 2008.

[11] C. Huang, R. Behara, and Q. Hu. Managing Risk Propagation in Extended Enterprise Networks. *IT Professional*, 10(4):14–19, 2008.

[12] C. Huang and J. Goo. Rescuing IT Outsourcing: Strategic Use of Service-Level Agreements. *IT Prof.*, 11(1):50–58, 2009.

[13] ISO/IEC 17799:2005 Information Security - Code of Practice for Information Security Management, 2000. http://www.iso.org.

[14] Y. Karabulut, F. Kerschbaum, F. Massacci, P. Robinson, and A. Yautsiukhin. Security and trust in it business outsourcing: a manifesto. *Electronic Notes in Theoretical Computer Science*, 179:47 – 58, 2007. Proc. of the 2nd Int. Workshop on Security and Trust Management (STM 2006).

[15] K. Mayer and N. Argyres. Learning to Contract: Evidence from the Personal Computer Industry. *Organization Science*, 15(4):394–410, 2004.

[16] D. Mellado, E. Fernández-Medina, and M. Piattini. A common criteria based security requirements engineering process for the development of secure information systems. *Computer Standards & Interfaces*, 29(2):244–253, 2007.

[17] R. Miura-Ko, B. Yolken, J. Mitchell, and N. Bambos. Security Decision-Making among Interdependent Organizations. *Computer Security Foundations Symposium, IEEE*, 0:66–80, 2008.

[18] A. Morali and R. J. Wieringa. Risk-based confidentiality requirements specification for outsourced it systems (extended version). Technical Report TR-CTIT-10-09, Centre for Telematics and Information Technology, University of Twente, 2010.

[19] A. Morali, E. Zambon, S. Etalle, and R. J. Wieringa. CRAC: Confidentiality Risk Analysis and IT-Architecture Comparison of Business Networks. Technical Report (Submited) TR-CTIT-09-30, Centre for Telematics and Information Technology, University of Twente, 2009.

[20] NIST: National Vulnerability Database, 2008. http://nvd.nist.gov/.

[21] L. Poppo and T. Zenger. Do formal contracts and relational governance function as substitutes or complements? *Strategic Management J.*, 23:707–725, 2002.

[22] E. Power and R. Trope. Averting security missteps in outsourcing. *IEEE Security and Privacy*, 3(2):70–73, 2005.

[23] R. Sabherwal. The role of trust in outsourced is development projects. *Commun. ACM*, 42(2):80–86, 1999.

[24] D. Sjøberg, B. Anda1, E. Arisholm1, T. Dybå, M. Jørgensen1, A. Karahasanovicacute1, and M. Vokáccaron. Challenges and recommendations when increasing the realism of controlled software engineering experiments. In R. Conradi and A. Wang, editors, *Empirical Methods and Studies in Software Engineering*, pages 24–38. Springer, 2003. LNCS 2765.

[25] Sarbanes-Oxley Act of 2002, 2002. http://www.sarbanes-oxley.com/.

[26] R. Wieringa. Design science as nested problem solving. In *Proc. of the 4th Int. Conf. on Design Science Research in Information Systems and Technology*, pages 1–12. ACM, 2009.

[27] R. Wieringa, N. Maiden, N. Mead, and C. Rolland. Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. *J. Req. Eng.*, 11(1):102–107, 2006.

APPENDIX

In the following we plot the tables showing the meta level concepts of CRAC++ and corresponding check-list method. An empty cell indicates that the method does not have a corresponding concept. The bold typed concepts are either objectively determined or are based on an information set that is accepted by all stakeholders and the risk assessors for all related assessments. (Interested users may contact Morali for further information on the check-list method.)

Table II
ASPECTS OF THE CRAMM METHOD.

Concepts of CRAC++	Concepts of Check-List
Confidentiality requirements	**Measures**
Confidentiality threshold	
Information assets	**Data**
Confidentiality value	**Data type**
Homogenity	
IT component	**Interfaces**
Information Flow Graph	
Number of instances	**Percentage of data**
Impact	Impact
Total impact	Business impact
Critical components	
Vulnerabilities	Vulnerabilities & Threats
Threat agents	**User type**
Competencies	
Attack Propagation Graph	
Ease of exploiting vulnerabilities	
Effectiveness	Mitigation level
Ease of accessing a component	
Bottleneck	
Protection level	Severity
	Sensitivity period
	Final Business impact level
	Percentage of user
	Threat type