GoCoMM: A Governance and Compliance Maturity Model*

Gabriela Gheorghe   Fabio Massacci
Stephan Neuhaus
Università degli Studi di Trento
Trento, Italy
firstname.lastname@disi.unitn.it

Alexander Pretschner
Fraunhofer IESE and TU Kaiserslautern
Kaiserslautern, Germany
firstname.lastname@iese.fraunhofer.de

ABSTRACT

Advanced methodologies for compliance such as CobiT identify a number of maturity levels that must be reached: first the existence of an *infrastructure* for the enforcement of security controls; second, the ability to continuously monitor and audit *quantifiable indicators* for the controls put in place; and third, the ability to *react* when a policy violation is detected. In this paper, we go further and define a governance and compliance maturity model (GoCoMM) that is *process-oriented*. As an instance of the highest level of governance and compliance, we suggest a method of *goal correlation* that provides measurable indicators of security and compliance by systematically refining business processes and regulatory goals. We also introduce a run-time architecture to support this model.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*Process metrics*; K.6.4 [**Management of Computing and Information systems**]: System Management—*Quality assurance*; K.6.5 [**Management of Computing and Information systems**]: Security and Protection

General Terms

Design, Documentation, Legal Aspects, Management, Measurement, Security, Standardization

Keywords

Compliance, Assurance Indicators, Security Indicators

1. INTRODUCTION

Recent years have seen regulatory bodies focusing on the protection of assets from (willful or sloppy) company mismanagement. The number and the complexity of requirements placed on IT processes has increased, as witnessed by the Sarbanes-Oxley Act [25], Basel II [3], the EU Directive 2006/24 on data retention, and many others. These trends have a profound impact on the trust models, security policies, security procedures, and security infrastructure that companies need to develop and maintain [13, 14]. Indeed, traditional security research has been concerned with the protection of data such as access control in its classical form [21] or in more sophisticated variants such as history-based [15], usage based [20, 8, 22], workflow-based [26, 6], or purpose-based [1] access control.

Yet, all these works are set out in a classical security model where a one-time failure of a security policy means that the security of the system has failed. In real systems, some failures of legally prescribed forms of usage or access control are always present and regulators are aware of it. What a company has to show to prove its compliance is that *it is in control* of its processes.

To this extent some methodologies that were initially used for auditing have been extended to IT [3, 10, 11]. The problem is that, with the exception of ITIL, such methodologies still clearly show their descent from their initial "auditor" origin. They tell perfectly how to check that you already are compliant; but they offer no hints on how to actually *become* compliant.

The same situation is present in the realm of software engineering and Capability Maturity Models [24]. While the objective of each maturity model may change, their hierarchy is pretty much the same: chaos, documentation, (manual) correlation between objectives and techniques, automation, measurements, and occasionally change management. Our starting point is a commonly acknowledged gap in these maturity models: While the *existence of metrics* is stipulated for reaching higher maturity levels (e.g., level 4 in the SE-CMM [9]), little is said about *how to derive concrete metrics* from existing artifacts in one particular context.

We contribute to closing this gap with a maturity model *and* the concrete elements needed to achieve the higher levels of maturity . To this extent we provide:

- A *maturity model* that allows companies to assess how effective their controls are when it comes to automating the verification of regulatory compliance;
- a *conceptual model* that is needed to achieve the first step of linking controls to objectives;
- an architecture for a *control IT infrastructure* that implements the conceptual model and is needed to achieve the maturity level that pertains to automation;
- measurable *indicators* that show to which extent a business process is compliant with applicable regulations (*key assurance indicators*, KAI) and to what extent controls are used to ensure the compliance of business processes (*key security indicators*, KSI).

*This work has been partly funded by EU under the project FP7-IST-IP-MASTER. We also thank all members of the A2 MASTER activity for fruitful discussions on the conceptual model.

2. A MOTIVATING CASE STUDY

The case study considered in this paper is based on a concrete process from the Hospital San Raffaele (HSR) in Milano, Italy. Private Hospitals with an officially recognized public function such as HSR are charged with administering drugs or with providing diagnostic services to patients that use their structure (e.g,, because the corresponding public services are overbooked) and then are authorized to claim the cost of drug dispensation or diagnostic provisioning from the regional state health administration.

As a consequence of their public function and because they treat sensitive data, the processes of HSR are highly *regulated*. In order to give a rough idea of the amount of regulation, the simple process of authorization and accounting for the dispensation and recompensation of drugs (called "File F") is subject to, e.g., Legislative Decree no. 196 of 30 June 2003 "personal data protection code", "Additional Measures Applying to Processing of Sensitive or Judicial Data" point 20; "Computerized Authentication System" points 1, 2, 3, 5; annex B, "processing by electronic means", "authorisation system" points 12, and 13, as well as several circulars including No. 17/SAN 3.4.1997.

We will have to consider *exceptions* as well. There can be situations when personnel can rightfully override certain behavior restrictions. For example, doctors must be able, in an emergency, to obtain any drug they need, even without going through proper channels first. In this case, requiring compliance with procedures could mean endangering patients. At the same time, it is important for the File-F process of HSR to be compliant with these goals also because failure to be so would mean to lose the status of an institution with public functions, and thus also the related revenues. Being able to prove that the process complies with the goals is hence an activity of utmost relevance.

Once we have an IT infrastructure that can enforce these policies, the quality of that enforcement must still be measured and, in the best case, quantified through suitable metrics. One problem here is that metrics are often selected on the basis of what can be easily computed from existing evidence, not on the basis of what would be useful for the organization. For instance, a typical security metric would be the number of computers with outdated virus signatures [12]. However, if this number rises by 10%, is HSR then exposed to 10% more risk? What if the computers with outdated virus signatures are not connected to the Internet and if they have no externally accessible interfaces except keyboard and mouse? This demonstrates the need for *indicators* with an immediate and intuitive meaning. We will get back to indicator measurement in §6.

3. THE GOVERNANCE AND COMPLIANCE MATURITY MODEL (GOCOMM)

A problem with maturity models such as BSIMM is that it is not clear precisely what an organization ought to do in order to upgrade from maturity level n to level $n\mathscr{C}1$. Our Governance and Compliance Maturity Model (GoCoMM) attempts to remedy this situation for the domain of governance and compliance.

The state of the art in assessing governance and compliance seems to be to hire a consulting company that will then *sample* parts of the system and then assess these parts through *review*. The problem with this approach is that it is selective: it is being carried out only at predefined moments in time, not necessarily correlated with the business needs. Also, it often requires what the BSIMM describes as a "fire drill" [17, CP 3.1], that is, taking reviewed components out of their production environment. However, company executives are interested in continuous assessments of production components, e.g., to adequately handle contingencies.

To see how this might be attained in practice, let us consider a simple example from the File F case study. Rather than checking once per year if drugs were prescribed by authorized personnel only, or sampling if reimbursement claims are consistent with what was actually administered, this checking could be done continuously. Possible reactions could then be taken immediately rather than at the end of the year. Ideally, it would be possible to not only *observe* non-compliance and to react, but to *ensure* it by implementing respective controls. In the example, the reimbursement process could be changed so that reimbursement forms are automatically checked for consistency with existing data, e.g., in the hospital's pharmacy, and be blocked if inconsistencies are detected.

In order to have continuous assessment one hence needs:

- *Control* of business processes; that is, the presence of certain mechanisms, called controls, at certain points during the execution of the business process. The purpose of controls is to ensure compliance (note that a goal can be both (1) to provide evidence of a situation S and then react, and (2) to guarantee that S does not happen).
- *Correlation* between controls and objectives, and also between indicators and controls; that is, the ability to tell which control is present because of which regulation, and which indicator relates to which control.
- *Automation* in collecting evidence and computing indicator values; that is, the freedom of human (manual) processes by which evidence for indicators is collected and combined; and
- *Measurements* to rate the effectiveness of the controls and other processes; that is, the evaluation of evidence to assess the degree of governance and compliance.

Using these four requirements as a baseline, we can define the five levels of maturity in GoCoMM:

Level 0: Chaos. No adequately documented level of control over business processes. A traditional audit is likely to fail.

Level 1: Control. The business has controls in place, but they are inadequately correlated with objectives. When objectives change, there is no process in place that could say which controls need to change because of the changed objective. We suspect that most organizations are at level 1.

Level 2: Correlation. The controls are correlated with objectives, but they do not (all) operate automatically.

Level 3: Automation. Controls operate automatically, but no indicator judges the performance of processes and controls.

Level 4: Measurement. There are automatically and consistently computed indicators in place that allow continuous assessment of governance and compliance.

4. CONTROL AND CORRELATION (L.1+2)

Value-generating processes (i.e., processes that are directly concerned with making money for the organization) are called *business processes*. Processes that are concerned with making sure that business processes behave according to some set of security or regulatory rules are called *control processes*. Similar to the boundary between non-functional and functional requirements, the line between both types of processes is fuzzy and a precise labeling may not always be achievable in practice.

Control objectives describe the security or regulatory goals that an organization needs to comply with. They are assigned to a specific business process or a sub-processes of a business process and are often formulated on a very high level of abstraction. For instance, HSR control objectives include "deliver the right drug to the right patient", and "obtain the correct prescription for the correct patient by the right operator."

Control objectives need to be broken down (refined) until the organization is satisfied that the control objectives are met when corresponding controls are in place and are working correctly. In other words, they must be refined until they can directly be related to business processes. This refinement can be achieved using security requirements engineering methodologies like [2, 5]. For instance, the control objective "access to the patient database must be regulated" can be decomposed into the sub-objectives "there must be role-based access control for accessing the database", "access to the database must be logged", and "access to the database must be protected by an authentication mechanism", among others.

Control objectives that are not further refined are achieved by *control activities*. A control activity is a means of managing risk, including policies, procedures, guidelines, practices or organizational structures, which can be of administrative, technical, management, or legal nature. These will usually consist of one sentence written in English, and will usually be named in accordance with a standard such as ISO 27001. Control activities are often simply called *controls*. As an example, ISO 27001 defines the controls "10.1.3—Segregation of duties", "10.10.1—Audit logging", and "11.2.3—User password management", all of which apply to the HSR case study.

Control activities are implemented by *control processes*. For instance, looking up a user's authentication credentials is part of a control process that is used to implement the ISO 27001 control "11.2.3—User password management." Because of the "atomic" nature of control activities, control processes and control activities can usually be directly mapped to each other. Having controls in place is necessary to reach maturity level 1, *control*.

Assuming that control objectives are refined in a systematic tree-like way and that there is documented relationship between control processes and control activities, we are ready to satisfy the requirements for level 2, *correlation*: for every control process, we identify the corresponding control activity and then follow the refinement backwards until we see for which control objective(s) it is relevant. Conversely, if a control objective changes, it is easy to compute the control processes that are affected by that change.

5. AUTOMATION (LEVEL 3)

The compliance run-time architecture is depicted in Fig. 1.

The *Policy Management Component and Policy Repository* manages policies and their evolution. Policies are used both at the specification and at the implementation level. At the specification level, they describe desired properties of the overall system—it is with respect to these policies that compliance is defined. At the implementation level, (different) policies are used to configure the other components. If needed, policy changes are pushed to the respective infrastructure components in a consistent and atomic manner. To manage the policies, it uses a policy repository.

The *Signaling Infrastructure* observes raw events in or at the interface of the service, and possibly processes them to generate more abstract events required by the monitoring policy. Signaling then hands them over to the monitoring infrastructure.

The *Monitoring Infrastructure* collects events that are emitted by the signaling infrastructure. It is hence capable of collecting and interpreting events from different signaling components. It checks

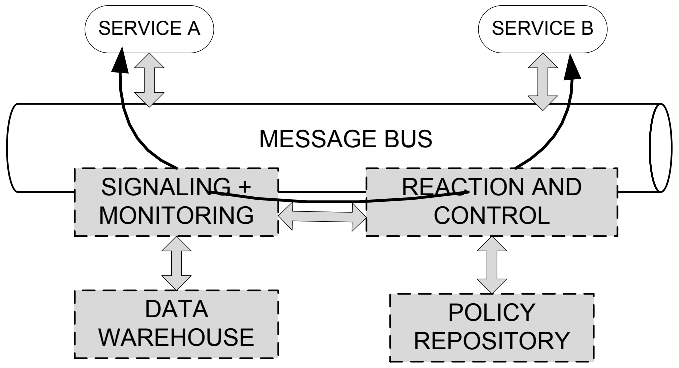

Figure 1: Run-time ESB architecture

whether specific conditions are satisfied and hence performs a further aggregation of signalled events (layered aggregation can provide significant speed-ups). These aggregations are passed, firstly, to the diagnostic infrastructure for the computation of KAIs and KSIs (§1). Secondly, they are passed to the enforcement infrastructure (see below) so that relevant policies can be enforced.

The *Diagnostic Infrastructure* computes KAIs/KSIs and to provide a control cockpit. Based on the computed indicators, the cockput allows humans to assess the security or compliance status, intervene in security governance, and to coordinate interventions in cross-organizational settings. To this end, it receives events from the monitoring component and displays them on the cockpit in a way that is meaningful to the business.

The enforcement of controls is done by the *reaction and control infrastructures*. The control infrastructure is used whenever a service provider allows the enforcement of a policy by technical means. This can happen by blocking events, modifying them (e.g., anonymize records by replacing name and birth dates with blanks), waiting for a condition to become true, or by executing additional actions (notify data owner upon transmission of personal data). In contrast, the reaction infrastructure concerns enforcement by observation and compensation, and is used when a policy violation has been detected, but not prevented. To this end, the reactive infrastructure receives events from the monitoring infrastructures and analyzes them w.r.t. possible policy violations.

6. INDICATORS (LEVEL 4)

We want to measure the degree of compliance of a process with its specified control objective. Being compliant with a control objective means that a process can emit only some traces, but not others: "being compliant" is really specifying which traces are allowed. We call the set of compliant traces the *ideal process*. For example, if the control objective is "only authorized personnel may access the patient database," then only traces that contain an authorization event before the access event are allowed. Traces that have no authorization or traces that contain authorization only after the access has already happened are not allowed.

Once we have specified the ideal process, we can view the actions of the control process as making sure that compliance is achieved. To this end, the control process is wrapped around the business process and vets and possibly modifies its inputs and outputs; see Fig. 2. This—conceptual—wrapping gives rise to a natural categorization of events that we later use for the definition of indicators. Events that enter the business process from the outside are category 0; events that come from the outside through the control process are category 1; events that come from the control process to the busi-

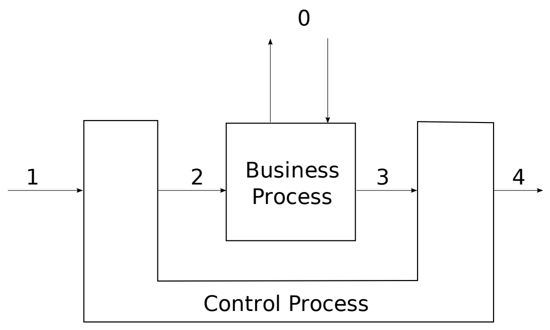

Figure 2: Channels between control and business processes

ness process are category 2; events that come from the business process to the control process are category 3; finally, events that leave the control process to the outside are category 4.

Indicators now tell us how well our business and control processes are working. They are based on evidence, collected from traces of events. Recall that a requirement for compliance is to use indicators to give quantitative answers to the questions "Is my business working correctly (compliantly)?" and "To what extent is this due to correctly functioning controls?" Recall that we have termed the first kind of indicator *Key Assurance Indicator* (KAI) and the second kind *Key Security Indicator* (KSI). A detailed discussion of how these two indicators differs can be found in [13]. Also, while we have formalized our indicators by essentially counting and comparing sets of traces that do and that do not comply, we have to omit this formalization here for lack of space.

For the first kind of indicators (KAI, correct operation of business process), we need to find out whether the traces that we get from a business-plus-control process are consistent with the ideal process. This is a security predicate in the sense of [7, 16]: an oracle tells us whether a trace is consistent with the ideal process or not. This indicator only looks at the input/output behavior of a process, so we will not need to know how the process works internally. In terms of Fig. 2, we only need to look at events of categories 1 and 4. Now that we have separated the non-compliant traces from the compliant traces, we can extract a meaningful metric from them. This metric usually comes as a natural answer to the question of when we consider an execution of this business process to be *good*, i.e., compliant with a control objective. For example, one criterion for a "good" business process execution results in all dispensed drugs being fully reimbursed. Once we identified the traces where that was not the case, one possible indicator would be "amount of money lost due drug issuances that were not reimbursed." Similarly, another KAI could be the "ratio of cost of non-reimbursed disbursements to the total amount for which a reimbursement application was made.". These metrics follow naturally from an understanding of the process's purpose and can be created at any level of refinement. (§4)

Showing evidence of compliant operation is sufficient for assurance indicators. It is not sufficient for control processes, however, because a control process that is never invoked does not increase the security or compliance of the controlled process. Therefore, key security indicators come in two flavors: KSI-correct for the correct operation of controls, and KSI-coverage for the coverage of controls. For KSI-correct, we can use the same scheme as for KAIs, comparing the control process with an *ideal control process*. An ideal control process is one that guarantees compliance of the combination of business and control process for all control processes that correctly implement the ideal control process. In terms of events, this time we also need to look at events of categories 2 and 3, since we are concerned with the input/output behavior of control processes. For instance, if the policy is "all data that leaves the system must be anonymized," one possible KSI-correct could be "number of instances where personal data went through the control process unchanged."

For KSI-correct we are only interested in whether the control process works correctly if it is invoked at all. In other words, we are ignoring category 0 events. For this, we need the second flavor of KSI, KSI-coverage. This indicator relates traces that were vetted by control processes to traces that were not vetted. For instance, if the policy is "all data that leaves the system must be anonymized", one possible KSI-coverage could be "number of instances where the anonymizer was supposed to be invoked, but wasn't."

7. PROTOTYPE AT ESB LEVEL

The above architecture can be implemented on top of any communication infrastructure for software components. In this paper, we will focus on service-oriented architectures. In these architectures, the Enterprise Service Bus (ESB) has become the favorite integration infrastructure, especially with the arrival of an integration standard called Java Business Integration (JBI) [1]. Architecturally, the JBI ESB is a bus onto which services or business processes are deployed. The bus virtualizes the endpoints of the services and mainly provides message mediation, routing, protocol transformations. The infrastructure mechanisms do not discriminate between the deployed services simply because their purpose is to integrate and not to isolate. It is thus natural to implement compliance analysis at the level of the ESB: not only that it is the state-of-the-art service integration technology, but it is also of a more 'tangible' level of abstraction than the business orchestration level.

At the JBI level, we decided to instrument the message bus in such a way that it intercepts all communication flows between all those services (or business processes) that are regulated by policies in the *policy repository*. A combined *signaling and monitoring* component decides on the relevance of an intercepted message flow and hands it to the *reaction and control* mechanism. In turn, this component performs either preventive or reactive actions onto the subject message flow so that the system trace becomes policy compliant (likely checked by different mechanisms). ESB policies, from this point of view, pertain to service endpoints (e.g., what services a service is allowed to communicate with) and service usage (e.g., threshold on number of requests launched to a service).

8. RELATED WORKS AND CONCLUSIONS

There have been substantial efforts in evaluating and facilitating the control over IT and financial processes. In this area, models like CobiT [10] and CoSo [19], SSE-CMM [9] and BSIMM [17] are most relevant to our work. These models and standards aggregate requirements to make a business process secure and also to assess its *maturity*, security-wise. Apart from evaluating information security, however, they fail to provide a measure of how efficient or credible this valuation can be, or how to define what it means for a business process to be on one security level or another. For example, BSIMM does not account for any path between the levels it describes, nor does it give a ration for security metrics or the purpose of these metrics.

Another way to assess systems is with the use of *security metrics* which use measurements of actual systems instead of "best practices" or "security standards" such as ISO 27001. This approach has gained prominence in the last few years [23]. The idea

[1]`http://jcp.org/en/jsr/detail?id=208`

behind using actual measurements instead of—or in addition to—best practices and standards is that measurements offer a dynamic, rather than static, way to observe what is actually happening in a system. However, in order to be useful, metrics need to be extensively contextualised, i.e., be made process-specific. This context specificity is especially important when research (e.g., by Nagappan et al. [18]) suggests that there will be no metrics that correlate with interesting numbers such as defect density across all possible software projects.

This point is also made by Jaquith [12], who claims that a good metric needs to be consistently measured, cheap to gather, expressed as a number or percentage, expressed using at least one unit of measure, and *context-specific*. The referenced work contains a collection of metrics, but these metrics in fact lack the contextual specificity. Evidence that these metrics actually correlate with interesting security data such as incident or vulnerability counts is not provided. This exemplifies one of the major research problems with most ongoing research on security metrics.

Other researchers, including Bellovin [4], see a problem with security metrics in general because of the inherent assumption in using metrics that software has a continuous failure mode. In other words, it makes sense to talk of software as "72% secure." Bellovin argues on the contrary that these failure modes are discontinuous (what he calls "brittleness") and therefore continuous metrics are not applicable. In this work, we are concerned with demonstrating that we are *in control*. In this context, we feel it makes sense to speak about being in control "72% of the time."

In this paper, we have introduced a maturity model and an enforcement infrastructure for measuring and ensuring compliance. Moreover, we have shown how to derive measurable indicators, or metrics, that are needed for reaching higher maturity levels. Our starting point is the business process and related control objectives. This is in contrast to standard compendia of metrics [12] that may be easy to compute but for which it is not always clear what they mean, how they relate to business or control objectives, or which may even harm an organization.

Similar to other models, our maturity model starts with chaos; then requires the existence of controls; then requires that of a correlation between controls and objectives; then requires the existence of automatic controls; and finally performs compliance measurements automatically.

9. REFERENCES

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *VLDB*, pages 143–154, 2002.

[2] Y. Asnar, P. Giorgini, F. Massacci, and N. Zannone. From trust to dependability through risk analysis. In *Proc. of ARES'07*. IEEE Press, 2007.

[3] Basel Committee on Banking Supervision. International convergence of capital measurement and capital standards, June 2006.

[4] S. M. Bellovin. On the brittleness of software and the infeasibility of security metrics. *IEEE Security & Privacy*, 4(4):96, July–August 2006.

[5] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Requirements engineering for trust management: Model, methodology, and reasoning. *IJIS*, 5(4):257–274, 2006.

[6] M. Hafner, M. Breur, R. Breu, and A. Nowak. Modelling inter-organizational workflow security in a peer-to-peer environment. In *Proceedings of the IEEE International Conference on Web Services*, pages 533–540, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[7] K. W. Hamlen, G. Morrisett, and F. B. Schneider. Computability classes for enforcement mechanisms. *TOPLAS*, 28(1):175–205, 2006.

[8] M. Hilty, A. Pretschner, D. Basin, C. Schaefer, and T. Walter. A Policy Language for Distributed Usage Control. In *Proc. ESORICS*, pages 531–546, 2007.

[9] International Systems Security Engineering Association. Systems Security Engineering - Capability Maturity Model, 2009. `http://www.sse-cmm.org/index.html`. Last document upload 2002.

[10] ISACA. Cobit. `www.isaca.org/cobit/`, 2008.

[11] IT Governance Institute. *IT Control Objectives for BASEL II. The important of Goverance and Risk Management for Complience*, 2007.

[12] A. Jaquith. *Security Metrics: Replacing Fear, Uncertainty, and Doubt*. Addison-Wesley Professional, 2007.

[13] Y. Karabulut, F. Kerschbaum, F. Massacci, P. Robinson, and A. Yautsiukhin. Security and trust in it business outsourcing: a manifesto. In *Pro, of STM'06*, ENTCS. Elsevier, 2006.

[14] G. Karjoth, B. Pfitzmann, M. Schunter, and M. Waidner. Service-oriented Assurance - Comprehensive Security by Explicit Assurances. In *Proc. of QoP'05*, 2005.

[15] K. Krukow, M. Nielsen, and V. Sassone. A framework for concrete reputation systems with applications to history based access control. In *Proc. of CCS'05*, 2005.

[16] J. Ligatti, L. Bauer, and D. Walker. Edit Automata: Enforcement Mechanisms for Run-time Security Policies. *Int. J. of Inform. Sec.*, 4(1-2):2–16, February 2005.

[17] G. McGraw, B. Chess, and S. Migues. The building security in maturity model. `http://www.bsi-mm.com/`, June 2009.

[18] N. Nagappan, T. Ball, and A. Zeller. Mining metrics to predict component failures. In *Proceedings of the 27th International Conference on Software Engineering*, New York, New York, USA, May 2005. ACM Press.

[19] C. of Sponsoring Organizations. Coso internal control-integrated framework. `www.sox-online.com/coso_2004_coso_framework.html`, 2004.

[20] J. Park and R. Sandhu. The UCON ABC Usage Control Model. *ACM Transactions on Information and Systems Security*, 7:128–174, 2004.

[21] P. Samarati and S. D. C. di Vimercati. Access Control: Policies, Models, and Mechanisms. In *FOSAD 2001/2002*, volume 2946 of *LNCS*, pages 137–196. Springer-Verlag, 2001.

[22] A. Schaad and J. Moffett. Delegation of Obligations. In *Proc. of POLICY'02*, pages 25–35. IEEE Press, 2002.

[23] A. Shostack and A. Stewart. *The New School of Information Security*. Addison-Wesley Professional, 2008.

[24] Software Engineering Institute. Capability Maturity Model Integration, 2009. `http://www.sei.cmu.edu/cmmi/`.

[25] United States Code. Sarbanes-oxley act of 2002, pl 107-204, 116 stat 745. Codified in Sections 11, 15, 18, 28, and 29 USC, July 2002.

[26] J. Wainer, P. Barthelmess, and A. Kumar. W-rbac: A workflow security model incorporating controlled overriding of constraints. *International Journal of Cooperative Information Systems*, 12(4):455–485, 2003.