Towards Systematic Achievement of Compliance in Service-Oriented Architectures: The MASTER Approach

Service-oriented architectures (SOA) provide the flexible IT support required by agile businesses. To simultaneously meet their compliance requirements, continuous assessment and adaptation of the IT controls embedded in SOA is mandatory. The paper outlines the MASTER methodology and architecture for systematic achievement of compliance in SOA. MASTER features automated support of the full control lifecycle, definition of key indicators that can be interpreted in the business context and scale to outsourcing scenarios, and a model-based and policy-driven approach that allows to capture business and technical context and to adapt metrics and controls to it.

DOI 10.1007/s11576-008-0086-1

The Authors

Dipl.-Inform. Volkmar Lotz
Dipl.-Ing. Emmanuel Pigout
SAP Labs France, SAP Research
BP1216
06254 Mougins Cedex, France
{volkmar.lotz | emmanuel.pigout}@sap.com

Dr. Peter M. Fischer
Prof. Dr. Donald Kossmann
ETH Zürich, Systems Group
Universitätsstrasse 6
8092 Zürich, Switzerland
{peter.fischer | donald.kossmann}@inf.ethz.ch

Prof. Dr. Fabio Massacci
Universita di Trento
Via Sommarive 14
38050 Povo (Trento), Italy
Fabio.Massacci@unitn.it

Dr. Alexander Pretschner
ETH Zürich
Information Security
Haldeneggsteig 4
8092 Zürich, Switzerland
alexander.pretschner@inf.ethz.ch

Submitted 2008-12-01, after two revisions accepted 2008-07-01 by the editors of the special focus.

1 Introduction

As a result of increased business complexity, growing public scrutiny and more stringent regulations emerging, regulatory compliance has received major attention and become a top priority target for businesses. Regulations typically address different business domains and assets, for instance,

- the Sarbanes-Oxley Act (SOX) (Congress of the United States of America 2002) focusing on protecting investors' interests by allowing them to make informed decisions based on the correctness of the financial reporting; or
- the Basel II Accord (The Basel Committee on Banking Supervision 2006) aiming at mitigating the risk caused by credits and loans issued by banks through requiring a given percentage of equity depending on the risk associated to a loan, and means to prove that appropriate risk assessment, management and mitigation controls are in place.

Regulations of this kind share the requirement of putting a set of controls in place that govern the organizational structure or constrain the behavior of business processes in order to facilitate the achievement of the respective control goals. Although some regulations contain the definition of the nature of a control (for instance, SOX section 802 demands a data retention period of 5 years), the specification and assessment of a concrete control architecture – the set of controls and their interaction with the system and among each other – is, in general, left as a task to the business being subject to a regulation, and their auditors. This reflects the view that a control architecture is the result of a risk analysis specific to the business at hand, and the way how a particular company is organized and operates to achieve its goals.

Risk assessment and control frameworks have been introduced to support businesses in setting up adequate control infrastructures. Risk assessment frameworks like COSO (COSO, n. d.) address the identification of a business' potential material weaknesses and the derivation of control goals from them. In addition, control frameworks introduce abstract controls as well as methods and instruments to manage them. COBIT (The IT Governance Institute 2007), for instance, introduces IT-related controls according to control domains and goals, along with indicators that allow assessing their effectiveness. The latter facilitates a major aspect of compliance achievement: since they are the result of a risk analysis activity, the set of controls cannot be claimed to be "complete", nor is their adequacy guaranteed. Thus, the controls need to be continuously observed and assessed. This assessment is a typical task of auditing. In sum, to achieve compliance, a business needs to:

- map abstract controls to concrete control structures and processes;

- enforce the controls in business operations; and
- evaluate the effectiveness of the controls.

In this paper, we focus our attention on IT controls for business systems built on top of Service-oriented Architectures (SOA) (Alonso et al. 2004). "IT controls" denote IT functionalities, components, and architectural concepts that are designed to achieve a set of control goals. Control goals can be expressed in terms of business structures and processes implemented in IT. An IT control is, thus, a software artifact that can be executed and that can interact with business processes running in software. Furthermore, we focus on security related controls, i. e., those controls that are designed and used in order to protect assets. This includes security processes and mechanisms like those introduced by ISO 27001 (ISO 2005) or Part 2 of the Common Criteria (n. a. 2006), but also controls set up to protect particular business assets, such as limiting the volume of trades for individual traders in a banking environment.

SOA provides a common platform that allows integrating services and components across organizational domains, reusing them in different business settings, and building applications through composing services. Thus, SOA provides an IT infrastructure that supports dynamic outsourcing and the maintenance of business ecosystems. SOA promises to adapt business processes, applications and their interactions to changing requirements and contexts. Because it enables flexibility and agility, SOA has been quickly adopted by software vendors, service providers, and businesses. However, the achievement of the compliance tasks mentioned above faces additional challenges:

- Abstraction: A crucial feature of SOA is that services can be accessed through an abstract interface. The abstraction levels of the control goals and these interfaces need not necessarily be the same. There is a need for an explicit mapping when control objectives are imposed on a service.
- Dynamics: SOA supports the continuous change of business relations (i. e., the services provided and consumed) and business processes (the orchestration of the services). Each change is potentially violating control goals or influencing the effectiveness of controls. Their evaluation hence is an ongoing task.
- Distributed control: A fundamental principle of SOA is that it is open to services of different providers being discovered and integrated at runtime. From the consumer point of view this means that controls may not be directly imposed on alien services, though some properties of them might be necessary for the achievement of control goals.
- Multiple trust domains: Third-party services are not necessarily trusted by their consumers. This increases the need for assuring that control goals are met and compliance properties are satisfied.

SOA thus asks for increased automation of the assessment of the controls' effectiveness, the support of their adaptation, and the enforcement of controls on services. Fortunately, we can use the SOA paradigm itself to facilitate this automation. In the remainder of this paper, we propose an architecture that extends a SOA by components for signaling and monitoring events caused by services, for their aggregation and analysis with respect to control objectives, for decision support for a business-level user when it comes to the specification of appropriate reactions to the observations made, and for the automated enforcement of a subset of controls. Thus, the architecture supports the full control lifecycle (observation, analysis, and reaction). The focus of the architecture is on security-related IT controls as introduced above, and we consider this restriction to be justified by the dominant role that the protection of assets has in control environments.

The remainder of this article is organized as follows. We illustrate the above principles and challenges by means of an example control in Section 2. Section 3 conveys the proposed architecture in detail, starting with an overview of the components and the major artifacts (models, policies, and indicators) that facilitate automation, before describing the components in more detail. Section 4 investigates into the role of model transformations to bridge abstraction levels. Section 5 concludes the paper, with an emphasis on the future work necessary to support dynamic outsourcing scenarios.

The work reported here has been performed in the context of the MASTER (Managing Assurance, Security and Trust for Services) project, an EU FP7 integrated project started in February 2008.

2 An example

We illustrate the challenges related to controls in a SOA by means of an example security control. Let us assume an implementation of a business process where each task is implemented by a web service. To distinguish them from services implementing controls, we call them "business services". The business services are orchestrated through a business process specification, for instance, using BPEL or BPMN. Let us further assume that one of the services being called includes interaction with a human user, requiring the user to log in to the service and perform an operation, where authentication of the user is considered to be critical. The user interaction, including both the fact that the interaction occurs as well as the login credentials or properties of them, is concealed by the service interface specification. For the sake of the example, we assume that the login mechanism is based on username/password authentication, and that this fact is publicly known.

In such an environment, the IT Control Objectives for SOX (The IT Governance Institute 2006) suggest "that procedures exist and are followed to maintain the effectiveness of authentication and access mechanisms". Part of such a procedure in case of a password-based authentication mechanism is to implement a control that requires "regular password changes". There are different ways to implement such a control, including a written guideline for users, sending e-mail reminders, the presentation of a "change password" GUI upon expiration of the current password, and blocking the user after expiration and unlocking only after the user has received a new password from the administrator. While a written guideline is a non-IT control, the remaining three controls are IT controls according to the definition of Section 1. The policies guiding the control may vary from simple ones ("every other month") to complex ones taking organizational or business process contexts into account ("if the user is involved in a critical process instance, his password must be changed every week, otherwise every month").

None of the example controls for the enforcement of regular password change provide a guarantee for achieving the control goal per se, and they need to be subject to monitoring and assessment. For instance, a user might enter the same pass-

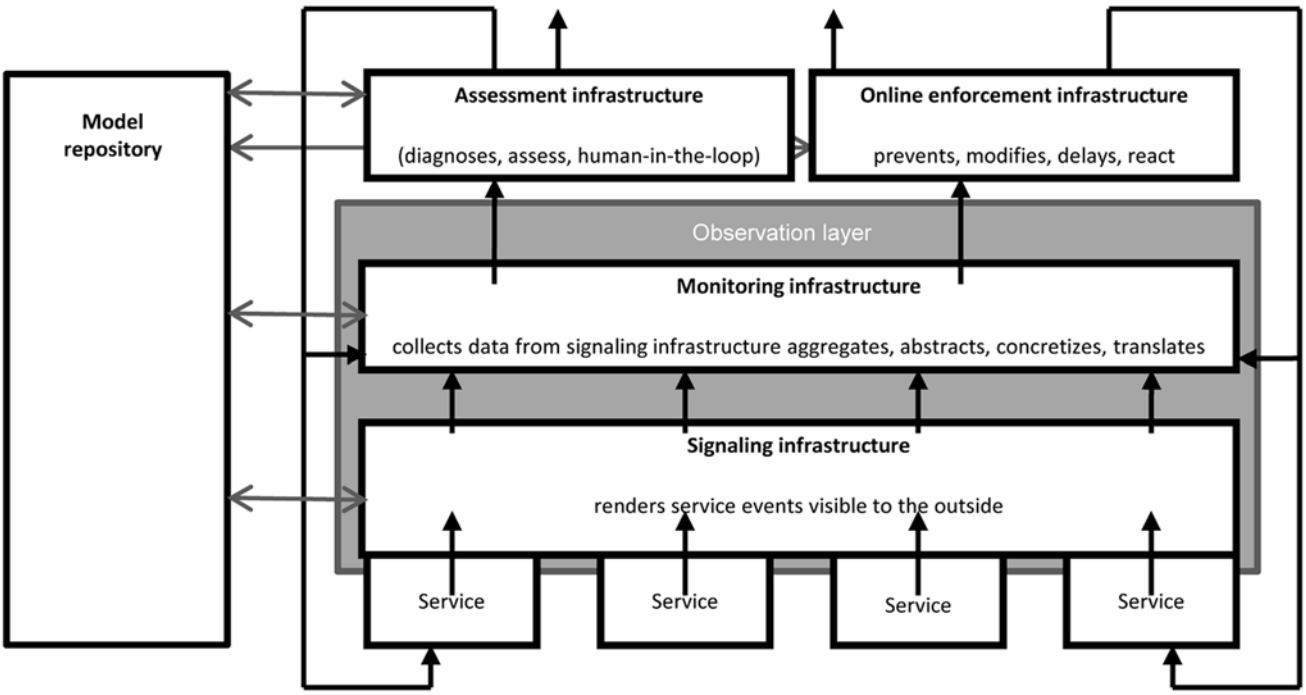

Fig. 1 Compliance architecture as proposed by the MASTER project

word again as his new one, or, if this is prevented by the mechanism, renew the password five times in a row immediately to arrive back at the original one. The assessment of the control mechanism's effectiveness depends on the events and resources that can be observed, and this is where SOA imposes additional challenges, but also offers new opportunities.

In our example implementation of the business process, the only directly observable activities are those that result in messages being passed between services. Observation requires subscribing to these messages, and subsequently analyzing them. If the password changing GUI is implemented as a service, the message calling it can be interpreted as the control being executed. However, this is not sufficient to show whether the password has indeed changed. To do so, it is necessary to either subscribe to the response of the password changing GUI service and relate that response to the respective request, or to provide another service that allows to access the password hash table and inspect it for changes for the respective user. If the service is owned by the business process owner, the latter might implement the service in a way that the hash table can be accessed through the service interface. In case of a third-party service, its owner, however, might not want to provide such access. This is because the hash table reveals user identities, an information that is likely to be considered as sensitive. The business process owner either needs to trust the service, bind it to a contract that can be enforced by a trusted third party, require additional evidence from the service (e. g., a description of mechanisms and policies being in place), or replace the service by another one that provides the desired properties. The latter is particularly supported in a SOA.

This simple example already shows some of the key aspects of compliance enforcement in a SOA:

- Indicators for the effectiveness of controls refer to observable events that may be complex, for instance, if they relate several messages of different services or evaluating context information, asking for maintaining a state upon observing, and relating to models describing the relevant parts of the system behavior.
- Controls and observations need to be guided by explicitly defined policies that can be passed through the service layers. In case a relevant service behavior cannot be directly observed (as the contents of the password hash table in the example above), such a policy is used as requirement for the service.
- In case of dynamic binding of services, trust establishment can be facilitated by indicators relating to properties of the service implementation ("data sheet"). These indicators cannot be observed.
- The flexibility of a SOA allows for the replacement of services if they turn out not to ensure effective control. Optionally, their weakness might be compensated by instrumenting additional control services and orchestrating them accordingly. In the example above, the observation of frequent calls of the GUI service can lead to calling a service that sends an alarm to the service administrator.

3 Towards a compliance architecture

In the following, we investigate architectural principles that cover the key aspects that we have introduced with the example. The described components are seen as part of an extended SOA that is suitable to facilitate compliance in a sense of assessing and enforcing security controls.

3.1 An integrated architecture

The example in Section 2 shows the importance of supporting the full control lifecycle, in particular the measurement and analysis of the effectiveness of the controls. Thus, a security control architecture

for SOA does not only need to implement the controls themselves. It also needs to provide means to observe events related to a control objective, analyze them and react to them. **Fig. 1** introduces the respective components.

We distinguish between an observation layer that hooks into the SOA and the services to extract raw events and aggregates them to complex events, and an enforcement layer that provides analysis and reporting facilities and decision support on the events as well as automated enforcement where possible. The observation layer consists of two parts. One part is provided by a signaling component that deals with distribution interfaces. The other part is provided by an aggregation component that performs monitoring along with the generation of complex events. As such, it raises the level of abstraction while taking into account contexts. Context is typically provided in terms of executable models, for instance, business process models. The architecture components can access these models and their execution state through a model repository that is considered an integral part of the architecture. In addition, the model repository contains descriptions of translations between different levels of abstraction as well as other models that will be discussed in Section 4.

The assessment and enforcement components receive input from the monitoring infrastructure and feed back into the service layer. The nature of the feedback of the two components is similar: changes of policies guiding controls, modified service orchestrations, or replacement of services. The difference is that enforcement emphasizes on automation of reaction and the adaptation of controls, while assessment focuses on reporting, knowledge management and data warehousing with respect to the events.

Policies, in the sense of rules restricting the behavior of business processes, play a crucial role in the MASTER architecture. In the example of Section 2, the frequency and conditions applying to password changes are defined in terms of a policy. Policies can be enforced in two fundamentally different ways. The first approach relies on a strategy of observe-and-penalize (Povey 1999). If users do not change their passwords regularly, their trust ratings may get decreased. This kind of enforcement relies on the following basic ideas: a violation of the law can usually not be prevented, but the threat of a fine acts as a deterrent. In the sequel, this kind of enforcement will be called "enforcement by observation". The monitoring infrastructure of our architecture, originally introduced to enable the observation of the controls, can be equally used to implement an "enforcement by observation" control. The second approach is to make sure that policies cannot be violated. In the password changing example, forcing users to change their passwords or assigning them new ones are enforcements of that type. In the sequel, we will refer to this enforcement scheme as "enforcement by control". In the context of our architecture, such a scheme will be implemented in terms of dedicated services as part of the implementation of a business service or a set of business services.

Both enforcement strategies are based on and apply to different trust models (Pretschner et al. 2007). Enforcement by control (which is the primary mechanism in the DRM sector) is based on the suspicion that the client will not adhere to a previously agreed-upon policy and that the objects in question are comparably cheap. In contrast, enforcement by observation is applicable in situations with higher risk associated, e. g., in outsourcing contracts, and where all parties have an interest in continuing their collaborations. However, if machine support can be provided for enforcement by control, e. g., to meet notification or logging requirements, then there is no need to rely on enforcement by observation. In other words, both forms of enforcement are relevant in the area of ensuring compliance.

3.2 Compliance assessment and key indicators

In this section, we discuss the type of indicators that are needed to assess the adequacy of security controls. Indicators are measurements of events and properties resulting in numerical values that support drawing conclusions with respect to the achievement of control goals. Indicators need to be measurable, i. e., derived from system observations including explicitly available specifications and models.

We give a first intuitive understanding of the concept of indicators by refining the example of Section 2 where we established a control goal ("password to be changed every other month"), and where we discussed events that indicate whether or not this goal is achieved. However, we might enter situations in which the relation between events and goal achievement is less straightforward. Consider the following situations:

1. Some users have not changed their password every two months and have performed some sensitive operation.
2. Some users have not changed the password as requested but have not performed any operation after the password renewal date.
3. Some users have performed sensitive operations and changed their password only every two months (recall the requirement that users involved in sensitive transactions need to change their password every week). However, they have not performed any sensitive operation after the first password change.

From the point of view of the password changing policy, all situations violate the policy. Yet, one might argue that (2) can be tolerated, since the users actually never logged in again. The violation (3) is also an example of "almost" meeting the control goal: the user does actually perform a regular password change but does not meet the requirements on its frequency. A human expert rating the situations is likely to conclude that (2) accounts for low risk, (3) for medium risk, and only (1) might qualify for a considerable risk. Compliance is not black or white. In situation (2), a control like the forced password change mechanism would actually completely mitigate the associated risk by first asking the user to change their password upon their next login attempt.

The example shows that mere indication of policy violation needs to be replaced by indicators pointing to potential risk caused by the type of violation.

The idea of such indicators is not new, and a number of so-claimed "security metrics" are indeed indicators of compliance with a well known standard (e. g. ISO 17799/27001) or with internal security policy or security design requirements (Swanson et al. 2003). The usual metrics for such analyses is the percentage of compliance with the set of rules. Several certification and accreditation methodologies (DISTCAP or DIACAP (US Department of Defense 2007), etc.) exist. Among other things, they are used to accredit governmental agencies or governmental tenders. Most of these methodologies also include a risk analysis as one of the best practices

but do not say how the analysis must be performed.

Vulnerability analyses are probably the best known security assessment approaches that produce indicators. During a test phase a team tries to penetrate the system boundaries, documents uncovered vulnerabilities and suggests suitable protection improvements. Such an analysis is quite subjective since the results of testing depend on the experience of the testing team but can be used by other methods as a starting point for more elaborate analyses.

A key observation of the MASTER approach is that all of the above methods, including those that have a risk analysis component (Butler 2006; Stoneburner et al. 2001; Gordon und Loeb 2003), suffer from the same drawback: they mostly focus on the IT aspects of system security rather than on the business impact that security decisions might have (Karabulut et al. 2007). In order to better assess the notion of compliance it is useful to make a distinction between measuring the satisfaction of a control goal in terms of observable events and measuring the means used to achieve that goal (a system description or "data sheet" that allows to draw conclusions related to the satisfaction of the goal). The former measure is by necessity a lag-indicator that is only available ex-post. The latter measure is instead a driving-indicator that can be measured ex-ante.

This leads to the definition of two types of indicators:

- **Key Assurance Indicators (KAI)** are measurable indicators negotiated by a client and a contractor to show that the client's control goals are addressed.
- **Key Security Indicators (KSI)** measure technical security features (technical, organizational, and process-oriented means) used by contractors to achieve a control goal.

One possible KAI in the context of our example determines the number of users that have performed some actions in the past month and have not changed their password for the past 3 months, divided by the total number of users that have performed some actions in the past month. A KAI close to zero is a good sign of compliance. Clearly, this indicator can only be established through observation. If, in our example, the measurement requires access to the password hash table, but the table is not disclosed by the outsourced party, the KAI cannot be evaluated. In this case, a KSI can compensate for this lack of accessibility.

The KSI establish the existence of technical measures in order to meet a KAI. Given the example mechanisms of Section 2, we have the following partial order:

(A) Guideline only, no technical security measure;

(B) User notifications via email, prompting them to change the password;

(C) System forcing the users to change the password at their first login after the expiry period;

(D) System that changes the password after the expiry period and notifies the user to meet the administrator to receive the new password.

Using the terminology of (Karabulut et al. 2007), KSI are "internal" metrics needed by contractors to assess the characteristics of their system. In contrast, KAI are "external" metrics that clients of an outsourced service might want to fix because they do not (nor wish to) control the inner status of their contractor's security measures. In our example, an applicable KAI is the percentage of employees that have a different password after each password expiry date. A sample KSI is the percentage of employees to which the forced password change activity (B) has been applied.

Notice that one does not imply the other. The password change activity (B) might be memory-less so users can simply reconfirm their old password or rotate among two passwords. For sake of exposition let's consider also a non-technical measure to meet the controls: assume that people from a particular cultural background would normally obey the rules. So a possible control measure is to make sure that new employees belong to that cultural background. In this setting the KAI would remain the same but the KSI would be the percentage of employees that passes the background screening.

When two companies negotiate an outsourcing contract besides usual SLAs they could now also negotiate KAI or KSI components.

The next step for the concrete deployment of the MASTER solution is to provide algorithms and methods so that one can transform KAIs and KSIs at a given level of abstraction into lower levels (e. g. when iteratively outsourcing a task to subcontractors), and vice versa. Moreover, the relationship between KAIs and KSIs must be defined. Such tasks need to be done dynamically and efficiently as processes in a SOA are dynamically changing and adapting to business needs (Massacci und Yautsiukhin 2007). Models describing the systems and services on the respective abstraction layer and points of view together with formal relations between them enable the provision of these algorithms by relating indicators to model elements and applying the respective relations.

3.3 Monitoring events in SOA

Monitoring provides the measurements for the computation of indicator values. In a SOA with abstract interfaces and distributed ownership of services ("outsourcing"), performing such measurements of course faces specific limitations. In general, direct control over services and processes of the outsourcing providers cannot be established. Particular events that are needed to determine the indicators might not be propagated outside the control of an outsourcing provider; this may be for privacy or business reasons. Even in scenarios where a single domain of trust and control exists, getting information from the individual involved components might be difficult, since the underlying software implementation might not provide the facilities to generate the necessary events at a sufficient level of detail.

Existing approaches for monitoring business applications do not provide solutions to these problems, since they focus on the even more restricted goal of monitoring individual applications or services within a domain where full control exists.

To overcome these shortcomings, the approach taken in MASTER is to establish a language for declarative, high-level descriptions of the monitoring policies as well as the interfaces and formats needed to implement them. Business- or regulation-level as well as service-level requirements can be expressed in this language. The language refers both KAIs and KSIs. The monitoring infrastructure interprets higher level polices in the given context and transforms them into lower-level policies, which are propagated across the services and domains involved. Services are instrumented based on these policies. In order to evaluate the indicators, the necessary transformations, correlations and computations are performed on the events.

Alternatively, the monitoring specifications (at any level of transformation) can be given to an outsourcing provider, which will provide the necessary interfaces to understand them and provide the resulting events and indicators. In such cases, the provider is free to implement the technical part completely under its control, but needs to give the corresponding legal and contractual assurances that the events and indicators are truthfully generated.

The password change example outlined in Section 2 highlights the issues faced in monitoring and how the MASTER approach helps to address them. The general requirement of "password changes" is transformed into a set of more detailed policies based on observable events and the actual system infrastructure to monitor, such as notification of actual password change operations within a certain timeframe or the lack thereof. However, the example shows that this requires access to information (the password hash table) that is likely to be restricted in the outsourcing scenario, thus requiring different strategies to determine the events needed for computing the "password change" indicator. A policy that describes which type of events to monitor can be supported by the outsourcing provider in order to convey the information about password changes, either at the level of individual changes of involved accounts or at a general level.

The proposed language must be powerful enough to specify any relevant event of a security policy and cover several layers of abstraction. This includes the support of transitioning between the abstraction layers, taking into account system context and indicator specifications. To achieve these goals, this language must be amenable to the model and indicator transformations outlined below. A starting point for such a language specification are declarative languages for events processing such as XQuery (Boag et al. 2007).

On a more technical level, MASTER will provide an infrastructure where monitoring is performed by a component separated from any kind of application logic or from other system functionality (e. g., data management or web services). This separate monitoring component can then be implemented in a secure and scalable way. Parts of such an infrastructure can run under different control, and can be leveraged to provide the necessary means to instrument services and perform the individual steps to determine (composite) events and compute indicators. As outlined in the architecture above, the monitoring infrastructure can be split into two layers: signaling and monitoring

Signaling components receive a monitoring policy, map them to observable events, identify the events as they occur, and transmit them to the monitoring infrastructure. Thus, a signaling component is service-specific: It needs to include knowledge about the logic provided by a service in order to refine abstract events referred to in a policy statement and to record these events. It also performs a service-specific pre-processing of observed events in order to identify an abstract event referred to in the local monitoring policy.

3.4 Enforcement and decision support

The enforcement infrastructure of the MASTER architecture is designed to support both types of controls introduced in Section 3.1. Enforcement by observation is done on the grounds of the signals received by the signaling and monitoring infrastructure. Depending on the granularity of the applicable policies, the monitoring infrastructure can either directly report a policy violation, or provide the signals that enforcement needs to decide itself if a policy was violated. In case a violation is detected, the enforcement infrastructure takes steps to penalize the violator, to undo relevant transactions, or to perform compensating actions. In some cases, this can be done automatically. In most cases, however, human interaction will be necessary to react.

Similar to observation, enforcement by control also makes use of the signals that it receives from the monitoring infrastructure. Respective components of the enforcement infrastructure will usually keep a local and partial copy of the system's state to prevent policy violations – the monitoring infrastructure can only report violations, yet not prevent them. Furthermore, the prevention of policy violations may require knowledge of a service's state that is not available at the level of the monitoring infrastructure. Depending on the requirements and applicable policies, enforcement by control can essentially be done in the following ways: by execution, by inhibition, by modification, and by delay (Pretschner et al. 2008). Executing an action, such as sending out a notification, adding an entry to a log, or deleting a file caters to requirements like "notify data owner upon each access", "log every access", or "delete file after 60 days". Inhibitors make sure that certain actions cannot be performed under specific conditions. For instance, a data item cannot be transferred to an entity that is not allowed to receive the item according to a Chinese Wall policy. Enforcement by modification autonomously changes a request or a response. For instance, rather than simply denying access to a specific resource, it can recommend access to a less critical resource. Finally, enforcement by delay simply waits a little time and then re-executes a request, in the hope that specific environment conditions have changed.

Conceptually, components that enforce by control consist of two parts. They implement a condition that specifies when a mechanism is applicable, e. g., when thirty days have passed, or if an undesired request for transmission is executed. These examples show that the condition may or may not include a triggering event. The second part of enforcement components implements an effect – an action to be executed, a denial of a request, a modified request, or the re-execution of a request after some time.

Technically, then, components that enforce by control consist of a monitor that checks the condition, and a piece of business logics that executes the effect. The implementation can be done in a variety of ways: as explicit wrapper components that act as filters for the input and output of a service – which is particularly interesting in the context of services that cannot be altered, as code that is woven into the source code of the business logic of a service – which obviously can only be done if this modified code can be deployed, at the operating system level by components that intercept system calls, or at the level of runtime systems where, for instance, virtual machines can be modified.

In addition to the problem of making sure that the signaling and monitoring infrastructures are sound and complete in the sense that they report all and only the events that are relevant for a specific policy, intentional and inadvertent bypasses of the control mechanisms must be avoided. Depending on the threat model, relevant challenges here include making sure that the service itself does not execute actions it is not supposed to execute without passing via the controlled

interface of the service; that code is not altered; and that there are no uncontrolled back doors into the service's business logic.

Sometimes, it is not possible or desirable to automatically decide if a policy is violated or about to be violated. One possible reason is given by underspecified requirements which may result in false positives sent by the monitoring infrastructures. In such a case, human feedback seems desirable. Rather than inhibiting an – assumed – policy violation or report a policy violation, the strategy then is to send a message concerning a suspected policy violation. A human user analyzes the available data and decides if actions need to be taken. Similarly, a user or a machine may be able to decide that a specific data stream exhibits anomalies, for instance, if it suddenly includes many banking transactions that exceed a specific amount. A preliminary analysis is then sent to a human who decides on whether or not to take the necessary actions.

4 The role of modeling

In Section 3, we have introduced the general MASTER approach, but still identified a set of challenges that need to be met in order to successfully implement MASTER. These challenges, among others, include the provision of algorithms for indicator transformations (KSI in particular), the derivation of complex events taking business context into account, and the relation of events and indicators to abstract control goals. We believe that a rich set of executable models governing the system's behavior, representing different viewpoints and providing status information to the compliance architecture are the key to meeting these challenges. In this section, we want to introduce our ideas on which models are relevant and how to use them to support compliance achievement.

In order to bridge the abstraction levels involved and to capture the required context that policies might refer to, the models we need should explicitly represent this information and make it accessible for the architecture components. For instance, the monitoring uses status information of an abstract business process model; the decision support assesses complex events with respect to a risk model; or the enforcement infrastructure refers to service interfaces and service choreography models. Already in the simple example of Section 2, a set of models turns out to be helpful: a declarative policy guides the controls, in case of more complex policies a business process model provides the necessary context, and if the password table is not accessible at an outsourced service, a technical model of the service could allow to derive a proper KSI.

The following list provides examples for the different models that we consider relevant for MASTER together with examples of their respective notions and terms in the proposed compliance methodology and architecture. Additionally, we indicate typical description techniques occurring at these levels.

- *Business models: assets, accounts, financial statements, general ledger, business processes, etc.*

Business models are the result of business level requirements engineering. General description techniques for high-level requirements like UML class diagrams can hence be used.

- *Technical system models: service composition, orchestration, etc.*

Service compositions and orchestrations are described using notations like BPEL, BPNM, or WS-CDL. Even though BPEL and BPNM model business processes, we do not consider them to be business models, since they assume properties of a technical infrastructure and refer to components in a SOA (actually, the common basic assumption in their usage is that tasks are represented by services in the technical sense). There are also languages with formal semantics emerging, e. g., Orc (Misra und Cook 2007).

- *Service models: interfaces, operations, resources, data, etc.*

Here we talk about web services specifications, typically expressed in WSDL. For the definition of model transformations, we see a need for extensions providing behavior descriptions.

- *Security models: policies, requirements, measures, etc.*

Typical ways to express security models include policy description languages (e. g., XACML, or SAML) and formal security models like (Brewer und Nash 1989).

It is essential for model usage in MASTER that the different models, in particular those on different abstraction levels, can be formally related. The existence of such relations allows the definition of model transformations. Refinement is one such transformation, but more complex ones will be necessary, for instance, the mapping of KAIs to KSIs and vice versa.

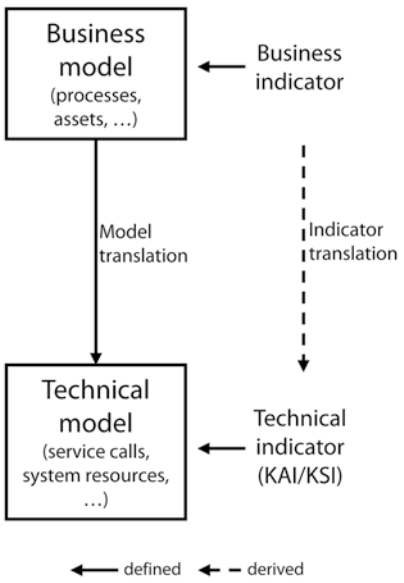

Fig. 2 Indicator translations are derived from model translations

The specification of events and the definition of indicators (both KAI and KSI) occur on all levels, thus supporting tracing and enforcing them across the levels. This requires a systematic mapping between events and indicators on the different levels which can be achieved if model transformations are available.

Fig. 2 illustrates the idea. An indicator is defined in terms of a model and its elements. For instance, if a business model contains assets, a KAI on the business level may refer to the number of assets. If a technical model refers to service calls, observation of a call can be used for KAI definition on the technical level. If indicators are strictly expressed in terms of model elements, the model transformations can be used to derive indicator transformations.

MASTER will ensure that a rich set of model transformations ensures the applicability of these concepts. For instance, at design time, the MASTER methodology will provide specification means (languages, language extensions, and tools) for the translations, refinement operators, and context transformations. These means will be machine accessible, so that they can be used at run-time for the real-time execution of translations, for instance, to enable compliance-aware matchmaking of services.

Zusammenfassung / Abstract

Volkmar Lotz, Emmanuel Pigout, Peter M. Fischer, Donald Kossmann, Fabio Massacci, Alexander Pretschner

Der MASTER-Ansatz zur systematischen Umsetzung von Compliance-Anforderungen in dienstorientierten Architekturen

Dienstorientierte Architekturen (Service Oriented Architectures – SOA) sind dank der Flexibilität der Dienste und der aus ihnen komponierten Anwendungen die heutige Referenz für die IT-Unterstützung agiler Unternehmen, die in einem dichten Netz mit Partnerunternehmen agieren und dynamisch Geschäftsprozesse ausgliedern. Die mit SOA einhergehenden Möglichkeiten haben aber gleichzeitig Konsequenzen hinsichtlich der Einhaltung von Regularien und Vorschriften: abstrakte Dienstschnittstellen, verteilte Verantwortlichkeiten und Interaktion über Unternehmensgrenzen hinweg stellen verschärfte Anforderungen an die Implementierung organisatorischer Kontrollprinzipien in einer SOA, insbesondere hinsichtlich der Bewertung der Effektivität der umgesetzten Maßnahmen.

Automatisierung bei der Umsetzung und kontinuierlichen Bewertung von Kontrollmaßnahmen ist essentiell. Der vorliegende Beitrag beschreibt die damit einhergehenden Herausforderungen anhand eines eingängigen Beispiels, um dann eine auf IT-Sicherheitsmaßnahmen fokussierte Methodik und Architektur einzuführen, die die Unterstützung aller Phasen im Lebenszyklus von Kontrollmaßnahmen ermöglichen. Die Architektur beinhaltet in eine SOA eingebettete Komponenten zur Beobachtung, Bewertung, Entscheidungsunterstützung und automatisierten Umsetzung von Maßnahmen. Der Ansatz basiert auf ausführbaren Modellen zur Bereitstellung von Kontextinformation auf unterschiedlichen Abstraktionsebenen sowie deklarativen Policies, die eine Steuerung der Maßnahmen erlauben. Modellen und Policies werden Indikatoren gegenübergestellt, die die Bewertung der umgesetzten Maßnahmen ermöglichen und Anhaltspunkte für kritische Situationen und notwendige Entscheidungen liefern.

Die vorgestellten Konzepte werden im Rahmen des EU-Förderprojektes MASTER realisiert, das damit den ersten automatisierten Ansatz zur systematischen Umsetzung von Compliance-Anforderungen in SOA liefern wird.

Stichworte: Compliance, serviceorientierte Architekturen, IT-Sicherheitsmaßnahmen, Sicherheitsmetriken, Überwachung, modellbasierte Ansätze

Towards Systematic Achievement of Compliance in Service-Oriented Architectures: The MASTER Approach

Service-oriented architectures (SOA) have been successfully adapted by agile businesses to support dynamic outsourcing of business processes and the maintenance of business ecosystems. Still, businesses need to comply with applicable laws and regulations. Abstract service interfaces, distributed ownership and cross-domain operations introduce new challenges for the implementation of compliance controls and the assessment of their effectiveness.

In this paper, we analyze the challenges for automated support of the enforcement and evaluation of IT security controls in a SOA. We introduce these challenges by means of an example control, and outline a methodology and a high-level architecture that supports the phases of the control lifecycle through dedicated components for observation, evaluation, decision support and reaction. The approach is model-based and features policy-driven controls. A monitoring infrastructure assesses observations in terms of key indicators and interprets them in business terms. Reaction is supported through components that implement both automated enforcement and the provision of feedback by a human user. The resulting architecture essentially is a decoupled security architecture for SOA with enhanced analysis capabilities and will be detailed and implemented in the MASTER project.

Keywords: Compliance, service-oriented architecture, IT security controls, security metrics, run-time monitoring, model-based architecture

5 Conclusions

The key to the systematic and effective achievement of compliance of participants in agile business ecosystems based on service-oriented architectures (SOA) lies in assessing and enforcing compliance through technical means. IT security controls provide a significant part of these means. However, they need to be embedded in a compliance architecture that allow their continuous assessment and support reactions to observed weaknesses. We have investigated the particular challenges that SOA features like abstraction and outsourcing support impose on such an architecture, and sketched a solution that is likely to meet these challenges: the MASTER architecture.

In contrast to existing control frameworks that provide written guidelines the effectiveness of which depends on human interpretation and expertise in implementing them in IT and support their lifecycle, this architecture embeds IT controls in an infrastructure that monitors their behavior, derives security and compliance related indicators from these observations, and supports automated reaction to identified weaknesses. Though human expertise is still required, MASTER makes it available for continuous evaluation, reuse and adaptation of the controls, once an initial set of controls is put in place and indicators are defined. While the example shows that the concept of KAIs and KSIs is adequate, the future provision of a rich set of such indicators is paramount for success.

The architecture components are themselves provided in terms of services and guided by declarative polices, which allows adapting them according to the requirements of specific regulations and controls in a better way as if they were embedded in the infrastructure. This is an essential feature for coping with different and evolving regulations and controls. However, different regulations applying at the same time might lead to conflicting control goals. The definition and implementation of resolution strategies in such cases is an important target of future work.

The events that need to be observed and reacted to can be complex results of aggregating and interpreting a set of direct observations (i. e., messages exchanged between services) and by taking organizational, business process, physical and IT context into account. We propose a model-

driven approach to manage this complexity: explicit representations of business processes, organizational structures, IT systems and services across several abstraction layers enable the maintenance of the necessary information at run-time and make them accessible to the architecture components. The security and assurance indicators are defined in terms of these models.

The proposed architecture will be implemented in the course of the MASTER project. Though the principal directions have been set, they will be constantly evaluated by means of case studies from different domains subject to differing, rich set of regulations: credit insurance, healthcare, and e-government.

Literatur

Alonso, Gustavo; Casati, Fabio; Kuno, Harumi; Machiraju, Vijay (2004): Web Services. Concepts, Architectures and Applications. Springer.

Boag, Scott; Chamberlin, Don; Fernández, Mary F.; Florescu, Daniela; Robie, Jonathan; Siméon, Jérôme (2007): XQuery 1.0: An XML Query Language. W3C Recommendation.

Brewer, David F. C.; Nash, Michael J. (1989): The Chinese Wall Security Policy. In: Proceedings of the 1989 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, pp. 206–214.

Butler, Shawn A. (2006): Security attribute evaluation method: a cost-benefit approach. In: Proceedings of ICSE-02, ACM press, pp.232–240.

Committee of Sponsoring Organizations of the Treadway Commission (n. d.): The COSO framework. http://www.coso.org/guidance.htm, retrieved 2008-06-30.

Congress of the United States of America (2002): The Sarbanes-Oxley Act, (Pub. L. No. 107–204, 116 Stat. 745). Available at http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=107_cong_bills&docid=f:h3763enr.tst.pdf, retrieved 2008-06-30.

Gordon, L.; Loeb, M. (2003): The economics of information security investment. TISSEC, 5(4), pp. 438–457.

ISO (2005): ISO/IEC 27001:2005 – Information technology – Security techniques – Information security management systems – Requirements.

Karabulut, Yuecel; Kerschbaum, Florian; Massacci, Fabio; Robinson, Phillip; Yautsiukhin, Artsiom (2007): Security and Trust in IT Business Outsourcing: a Manifesto. Electronic Notes in Theoretical Computer Science, Vol. 179, Elsevier, pp. 47–58.

Massacci, Fabio; Yautsiukhin, Artsiom (2007): An Algorithm for the Appraisal of Assurance Indicators for Complex Business Processes. In: Proceedings of the 3rd ACM Workshop on Quality of Protection. ACM Press.

Misra, Jayadev; Cook, William R. (2007): Computation Orchestration: A Basis for Wide-Area Computing. In: Journal of Software and Systems Modeling 6 (1), pp. 83–110.

n. a. (2006): Common Criteria for Information Technology Security Evaluation, Version 3.1, http://www.commoncriteriaportal.org/thecc.html, retrieved 2008-06-30.

Povey, D. (1999): Optimistic Security: a New Access Control Paradigm. In: Proc. Workshop on New Security Paradigms.

Pretschner, Alexander; Massacci, Fabio; Hilty, Manuel (2007): Usage Control in Service-Oriented Architectures. In: Proceedings of the 3rd International Conference on Trust, Privacy & Security in Digital Business, Springer, LNCS 4657.

Pretschner, A.; Hilty, M.; Basin, D., Schaefer, C.; Walter, T. (2008): Mechanisms for Usage Control. In: Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS), Tokio, pp. 240–245.

Stoneburner, Gary; Goguen, Alice; Feringa, Alexis (2001): Risk Management Guide for Information Technology Systems. NIST report 800–30, http://csrc.nist.gov/publications/nistpubs/800–30/sp800–30.pdf, retrieved 2008-06-30.

Swanson, Marianne; Bartol, Nadya; Sabato, John; Hash, Joan; Graffo, Laurie (2003): Security Metrics Guide for Information Technology Systems. NIST, Report 800–55 2003, http://csrc.nist.gov/publications/nistpubs/800–55/sp800–55.pdf, retrieved 2008-06-30.

The Basel Committee on Banking Supervision (2006): The Basel 2 Account. http://www.bis.org/publ/bcbs128.htm, retrieved 2008-06-30.

The IT Governance Institute (2006): IT Control Objectives for Sarbanes-Oxley. http://www.isaca.org/Template.cfm?Section=Home&Contentid=17003&Template=/ContentManagement/ContentDisplay.cfm, retrieved 2008-06-30.

The IT Governance Institute (2007): Control Objectives for Information and related Technology (COBIT), Version 4.1. http://www.isaca.org/Content/NavigationMenu/Members_and_Leaders/COBIT6/Obtain_COBIT/Obtain_COBIT.htm, retrieved 2008-06-30.

US Department of Defense (2007): DoD Information Assurance Certification and Accreditation Process (DIACAP). Instructions Num. 8510.01,. http://iase.disa.mil/ditscap/, retrieved 2008-06-30.