ExPDT: A Policy-based Approach for Automating Compliance

The majority of current approaches for achieving compliance rely on controlling access to data and processes as well as documenting their actual use and execution. Automating compliance often conflicts with business requirements since too rigid compliance rules oppose the need for flexible adaptation of business processes to situational context. As solution, the formal policy language ExPDT is presented allowing both. It is discussed how ExPDT can be used to bridge non-technical compliance requirements and technical IT systems by adequate expressiveness, calculability, and modularity, and to maintain flexibility of business processes by enabling optional control decisions.

DOI 10.1007/s11576-008-0078-1

The Authors

Dr. Stefan Sackmann
Dipl.-Inf. Martin Kähmer

Albert-Ludwig University of Freiburg
Department of Telematics, Institute of Computer Science and Social Studies
Friedrichstraße 50
79098 Freiburg
Deutschland
{sackmann | kaehmer}@iig.uni-freiburg.de

Submitted 2007-12-01, after two revisions accepted 2008-06-27 by the editors of the special focus.

1 Automating compliance of business processes

Compliance is about ensuring that business processes are executed as expected and that operations as well as practices are in accordance with prescribed laws (e.g., Sarbanes Oxley Act SOX, HIPPA), regulations (e.g., Basel II, Solvency II), agreed on standards (e.g., ISO/IEC 27000-series), commercial contracts (e.g., Service Level Agreement, Non-disclosure Agreement), or a company's governance. Together, the resulting compliance requirements constitute a set of rules to which a company has to validate its adherence for being compliant.

Current studies show that the frequency of audits, monitoring, and reporting correlate with the success of compliance management (Liebenau and Kärrberg 2006). As compliance validation is still mainly a manual task (Bace and Rozwell 2006; Agrawal et al. 2006), automation offers a significant economic opportunity. Most current tools start at making controls more efficient, e.g. by integrating hard-coded checks into standard software for validating a compliant segregation of duty as prescribed by SOX or by deploying compliance repositories as in the GRC Repository of SAP (Sadiq et al. 2007) or workflow repositories described in (Agrawal et al. 2006). Due to the inflexibility of the tools used, there is a risk of becoming trapped in routine tasks and reducing the ability to adapt business processes flexibly to market needs and context-specific requirements. At worst, the use of IT for automating compliance without a general approach may even be harmful (Cannon and Byers 2006).

Automating the validation of compliance requires a method-based approach integrating compliance management, risk management, and business process management as depicted in **Fig. 1**. The result of compliance and risk management defines the relevant compliance requirements that have to be translated into control objectives. The control objectives that are to be achieved automatically have to be mapped to a compliance policy. Then, these compliance policies serve as input for control processes forming the basis for monitoring and enforcing given rules within the business processes. The link between business processes and control processes can be realized, e.g., by annotations (Muehlen and Rosemann 2005) or by superordinated and independent modeling (Sadiq et al. 2007; Karagiannis 2008; Namiri and Stojanovic 2008). This paper focuses on a language to formulate compliance policies called ExPDT (Extended Privacy Definition Tool) which is based upon the work of Raub and Steinwandt (2006). ExPDT was originally defined for enforcing privacy (Kähmer and Gilliot 2008) and will be extended in this paper to show its usability for validating compliance adherence of business processes.

2 A framework for automating compliance

Since adherence to law cannot be directly encoded in machine-readable code, a framework outlined in **Fig. 2** is proposed. In a first step, laws and regulations that are described in a textual form have to be interpreted for a business domain and transformed into compliance requirements. This transformation is the focus of, e.g., best practice frameworks for IT governance COBIT (ITGI 2007) or ITIL (OCG 2007) as well as methodologies presented in Klempt et al. (2007) or Raghupathi (2007) and is hard to automate. Breaux et al. (2005) developed a method for getting a set of requirements from legal texts by using predicates to identify and classify language patterns. In the best event, the compliance requirements at this level match the legal requirements of the

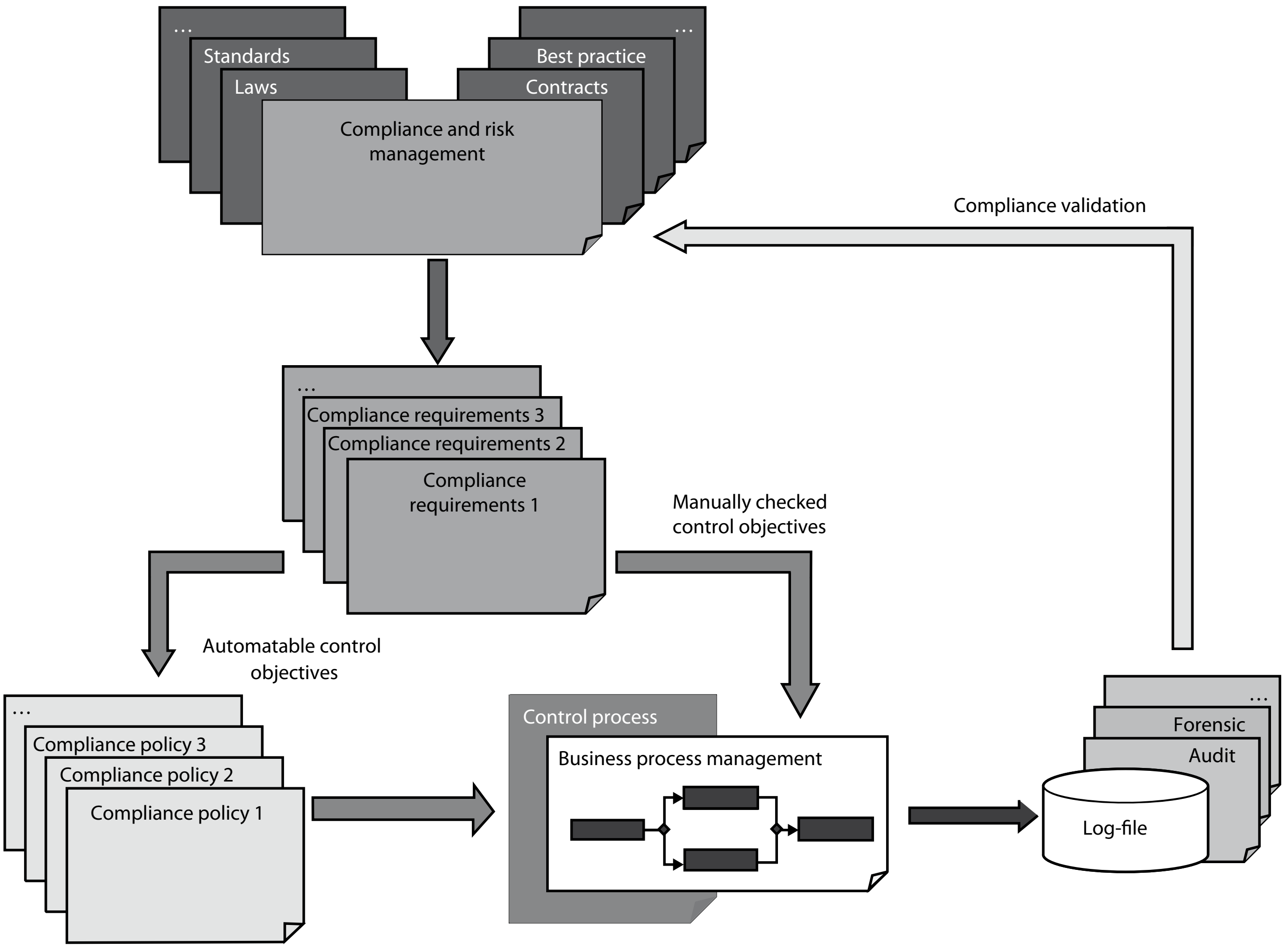
…
Standards
Laws
…
Best practice
Contracts
Compliance and risk management
Compliance validation
…
Compliance requirements 3
Compliance requirements 2
Compliance requirements 1
Manually checked control objectives
Automatable control objectives
…
Compliance policy 3
Compliance policy 2
Compliance policy 1
Control process
Business process management
…
Forensic
Audit
Log-file

Fig. 1 Process of compliance validation

superordinated level correctly and completely so that their fulfillment is identical with the fulfillment of the respective laws. Incorrect or incomplete requirements at this level always give cause for an ex post control and non-automatable validation of actual operations.

In a second step, the compliance requirements have to be transformed into a policy. Policies are a substantial element of security and privacy management since they describe what is allowed or prohibited and what is mandatory (Schneider et al. 2001). Converting compliance requirements into a set of machine-readable policy rules is a challenging task (Sadiq et al. 2007) that requires a domain ontology and, where policies are derived from multiple regulations, a solution to inconsistency between them (Delbaere and Ferreira 2007). As we consider the policy language as key to automating compliance, the following sections focus on this level.

The adherence to such policies is observed by monitors. They are usually based on the principal of Schneider's execution monitor (Schneider 2006) that intercepts all those commands at processor-level that would violate a policy and inhibits their occurrence by stopping the current execution. Approaches hosting this principle on middleware are, for example, IBM REALM that automatically refines regulatory policies to standardized low-level process events allowing a monitor to control the flow of business processes (Giblin et al. 2006), or the Hippocratic database that enforces security and privacy policies at data item-level (Johnson and Grandison 2007). Where the IT system runs correctly and safely, enforceable rules cannot be violated and an ex post validation thus becomes pointless.

For non-enforceable rules, such as obligations (Hilty et al. 2005), a violation can only be detected after the fact by an audit. For example, if an obligation such as "delete stored data after two years" has been fulfilled or not, can be validated at the end of the period of time by analyzing a corresponding log-file. For such validations, e.g., the Hippocratic database offers an interface to its query logs, and Delbaere and Ferreira (2007) developed an audit service to collect and evaluate logs at an enterprise-level. However, to be used for compliance validation, the required logs have to be complete and correct (Sackmann et al. 2006; Sadiq et al. 2007; Accorsi 2008). If, however, the fulfillment of the obligation occurs outside a company's own security domain, e.g., in the case of outsourced parts of the business process, an enforcement of the obligation cannot be guaranteed. External auditing has to be done that can only be automated by expanding the security domain, e.g. by deploying trusted computing (Iliev and Smith 2005). Automation of audits and compliance validation can then reduce the assessment time and, correspondingly, the time of remediation or mitigation of control deficiencies and thus reduce the economic risk of noncompliance.

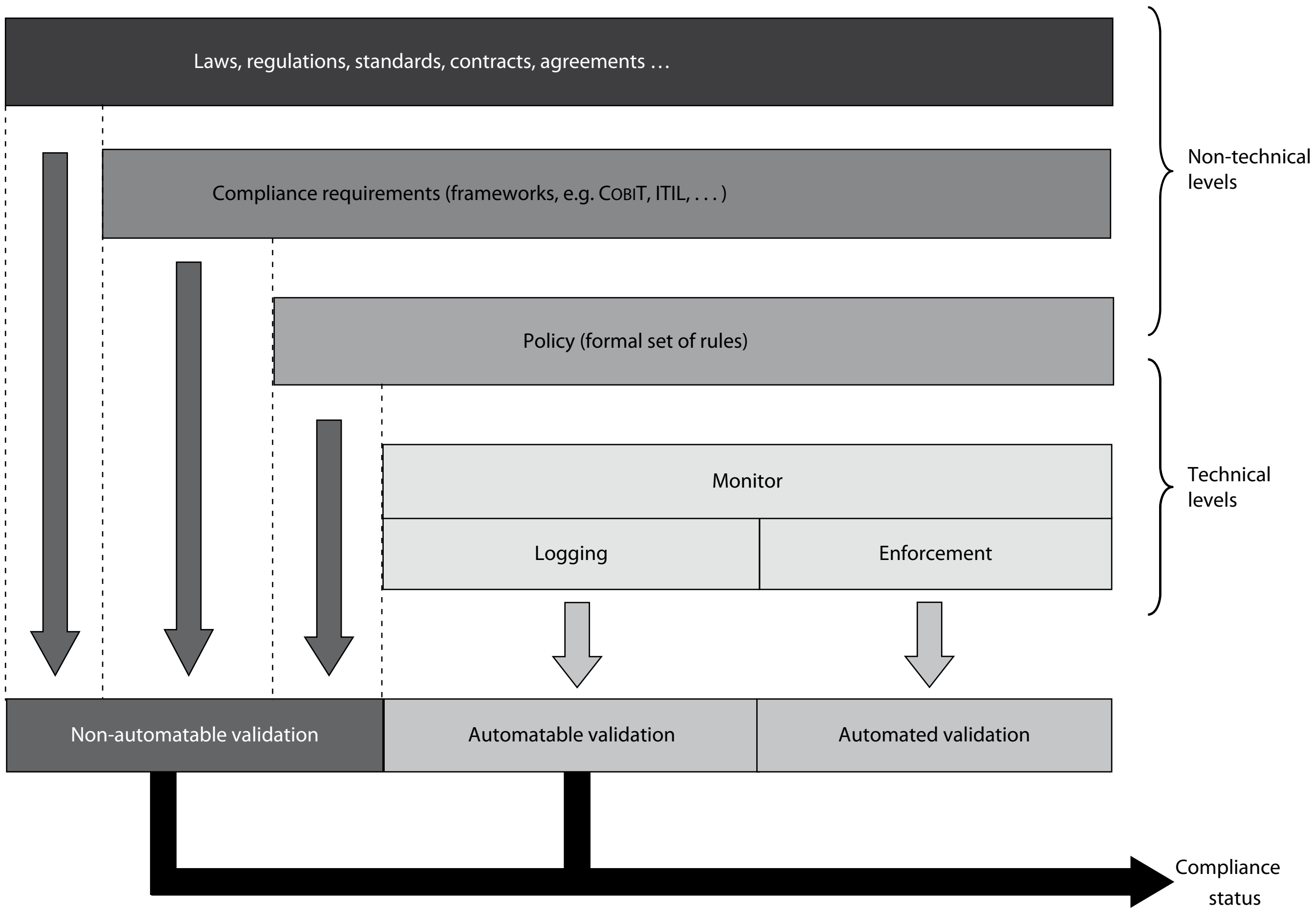
Laws, regulations, standards, contracts, agreements …
Compliance requirements (frameworks, e.g. COBIT, ITIL, . . .)
Policy (formal set of rules)
Monitor
Logging
Enforcement
Non-technical levels
Technical levels
Non-automatable validation
Automatable validation
Automated validation
Compliance status

Fig. 2 Framework for automating compliance

3 Compliance policy languages

The key to automating compliance is the policy language since it determines the set of compliance requirements that can be expressed in a formal way. For automating compliance, a policy language must satisfy at least the following four criteria (Sackmann et al. 2008):

1. **Expressiveness:** Current efforts to secure a system concentrate on the specification of undesired system behavior and preventing bad things from happening by controlling access to the system objects. However, just preventing undesired behavior is not sufficient for compliance. To ensure that a system reaches a desirable state, good things need to happen as well. In a policy, this can be achieved by controlling the usage of data objects by additionally imposing obligations on granted access rights and ordering the execution of actions that conduce to the process result (Breaux et al. 2005). Thus, not only permissions and prohibitions describing what may or may not happen within a system, but also obligations and orders triggering what must happen need to be specifiable in a policy (Müller et al. 2008).
2. **Flexibility with sanctionability:** A policy language should be able to provide optional decisions. This is required for economic reasons, since not all possible situations and especially their impact on a particular business process instance can be considered ex ante. If the economic impact of interrupting a process outweighs the risk of non-compliance, the policy should be able to allow the escalation within a business process, e.g., the skipping of a required control. For escalation not becoming the default behavior, sanctions have to be specifiable as well.
3. **Calculability:** Since compliance policies can be derived from different compliance requirements, it is likely that inconsistencies or even contradictions of the requirements are to be taken into consideration. Conflicting rules avert an automated decision and thus mean an operational risk. Therefore, the calculability of the policies should be feasible for detecting such conflicts and resolving them as far as possible.
4. **Modularity:** It is necessary to allow for modular specification of the policy rules so that every single compliance requirement can be addressed and combined with valid policies for deployment.

There are already quite a few policy languages. The World Wide Web Consortium (W3C) developed the Platform for Privacy Preferences (P3P) to express privacy policies in a machine-readable form and its counterpart, the P3P exchange language APPEL, to express customers' preferences (Cranor et al. 2005; Cranor et al. 2006). Reduced in their capability to control usage of data and execution of processes, both lack conditions, obligations, and any kind of enforceability. IBM's Enterprise Privacy Authorization Language (EPAL) accounts for the further usage of accessed data objects by supporting obligation elements in its policy rules and exhibits a more fine grained vocabulary as well as monitor integration (Ashley et al. 2003).

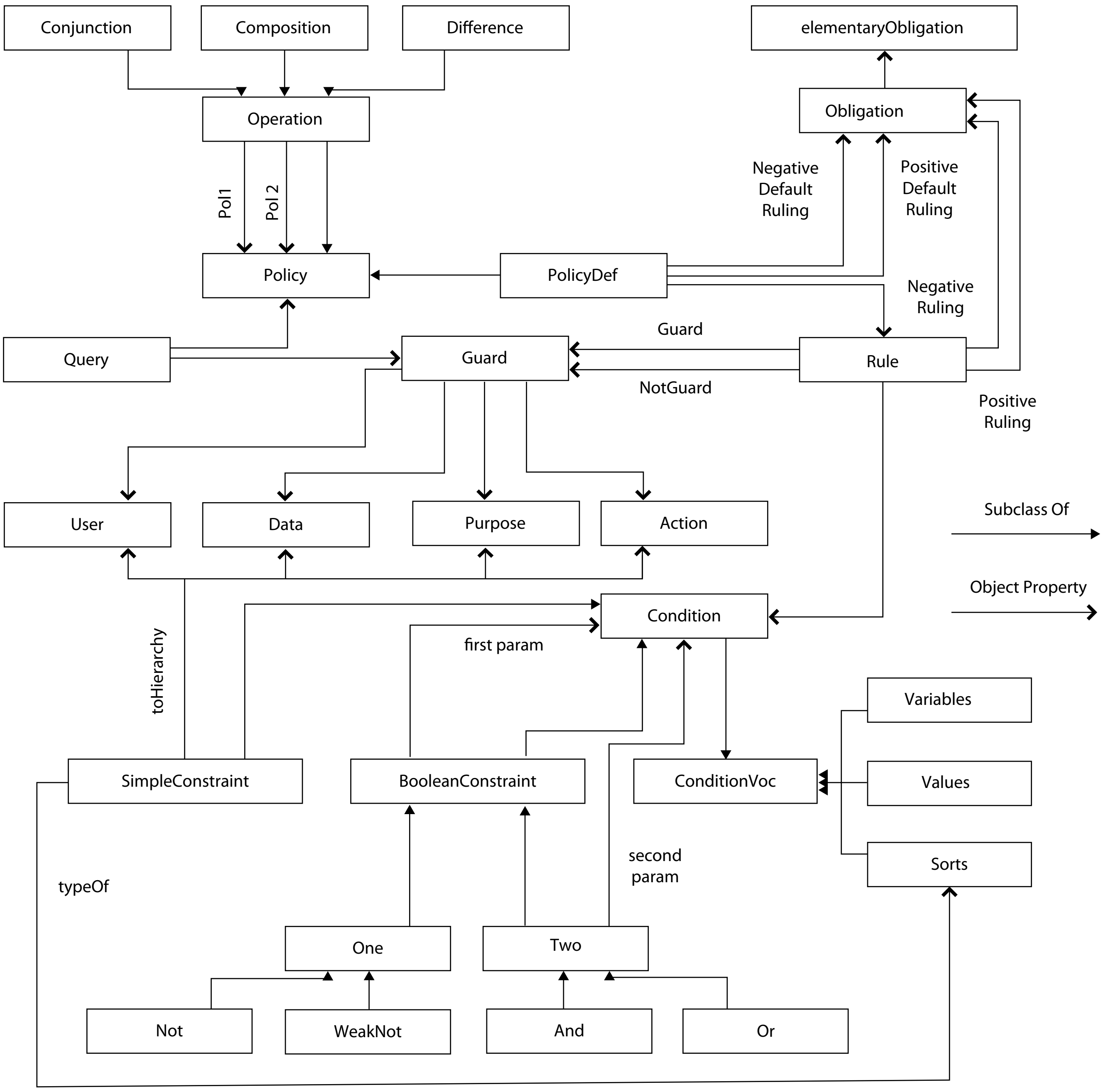

Fig. 3 Class diagram of ExPDT language for complicance policies

The Novel Algebraic Privacy Specification (NAPS) framework enhances EPAL on a logical level to an algebra additionally allowing for modular specification of policies (Raub and Steinwandt 2006) and adds a concept of sanctions to allow for flexible rule adherence. The eXtensible Access Control Markup Language XACML (Moses 2005) was designed by the OASIS consortium as open standard to specify expressive policies covering both usage control and commands. A data flow model describes how XACML policies are to be interpreted and enforced. In contrast to EPAL, XACML provides policy combination tools to support distributed policies, although it is not suitable for comparing or negotiating policies, because the intersection of two general policies is not defined. The WS-Policy framework (Bajaj et al. 2006) for web services provides a general purpose model and syntax to describe and communicate policies of web services, which consists of sets of different kinds of assertions, e.g., for security, privacy or reliability. Although allowing for optional assertions, this flexibility cannot be guided by sanctions or penalties. Optimized for queries on activity relations, XQuery can be used to hard-code simple policy rules and check them within continuous XML streams, such as system logs (Botan et al. 2007). Common to all of these policy languages is the lack of adequate operators for comparing and analyzing policies. Focusing on this calculability, the Formal Contract Logic FCL (Sadiq et al. 2007) expresses semantics of only single contracts in the former mentioned modalities and allows for reasoning not only about contract consistency but also about violations and possible sanctions. Exhibiting similar calculability and expressiveness, PENELOPE (Goedertier and Vanthienen 2006) deals with compliance requirements from different sources but lacks flexibility and sanctionability.

Some of the languages exhibit sufficient expressiveness covering usage control and orders, others bring along the concept of sanctions. But none of them satisfy all four criteria, usually missing calculability or modularity.

4 ExPDT – A formal language for compliance validation

The Extended Privacy Definition Tool (ExPDT) language allows users to specify declarative policies over specific domain knowledge using OWL-DL (McGuinness and van Harmelen 2004), a computational complete and decidable subset of the Web Ontology Language (OWL) corresponding to Description Logic. Although originally developed to formalize privacy preferences, ExPDT can also be used for compliance validation in processes. Its expressiveness allows the describing of permissions, prohibitions, and orders that have to be adhered to in case of certain contextual provisions or obligations. Sanctions, whose use is discussed in section 5 by means of a detailed example, can be specified on rule level. Based on the algebraic framework NAPS (Raub and Steinwandt 2006), ExPDT inherits semantics and provides difference as well as combination operators, allowing calculability as well as a modular specification of policies (Kähmer 2008).

4.1 Syntax

The syntax of ExPDT language is presented on the basis of the simplified OWL-DL class diagram with the inheritance and selected properties of the OWL-DL classes (see **Fig. 3**).

An ExPDT *Policy* is defined either by a prioritized list of rules and a *Default Ruling* in the case where no rule applies or by a result of a policy *Operation*. A *Rule* is comprised of one or more possibly negated guards constraining the scope of this rule from users, actions, data and purpose, a number of conditions and the ruling that subsequently delivers the decision of this rule. Hence, a generic rule has the following form:

([¬](*User, Action, Data, Purpose*))+, *Conditions,* (*Ruling*)

The element instances of a *Guard* are partially ordered in hierarchical structures allowing for grouping of instances and the formulation of policies rules applying to entire sub-hierarchies, e.g., to all users of a particular department or all the data belonging to a particular purchase order. Thereby, each of them has his own structure: customers, employees and system services are combined in the *User* structure, system objects and data items are described in *Data*, possible actions on these are given in *Action* and the possible intentions of actions in question are structured in *Purpose*. It is not required that hierarchies have unique predecessors as long as they form a directed acyclic graph.

Compliance requirements often depend on context information, e.g. permitting a purchase order only if its value is below a certain limit or a supervisor has given his consent. For the inclusion of such constraints, a many-sorted, 3-valued Łukasiewicz L_3 logic (Gallier 1988) is reverted to. A condition is a formula of this logic defined over the condition vocabulary *ConditionVoc* and its interpretation functions. The condition vocabulary consists of the final set of *Sorts* (i.e. variable types) each with a final set of *Variables*. The set of non-logical symbols of simple constraints *SimpleConstraint* includes relations, the set of logical symbols the operators *And*, *Or*, *Not*, *WeakNot* and *Values* 0, 1 and u as undefined. The undefined value *u* is advantageous to an environment of dynamic character, such as a company with continuously changing transaction partners and modified or switched services. If the evaluation of a condition does not return a clear decision 0 or 1 due to lack of available information, e.g., whether a certain threshold is exceeded or not, the evaluation of the policy is continued and the decision conjunctively joined with the final decision, as will be shown later. Formulas and terms of the condition logic are recursively defined as usual as in the predicate logic free of quantors.

A policy rule not only regulates the actions on data items, but can impose *Obligations*, such as "notify auditor" or "complete confirmed double check within one day". In contrast to many other policy languages ExPDT does not consider obligations as pure black box instructions. It has an underlying obligation model of a half lattice above the power set of the *elementary obligations* $\tilde{O}$, subset as relation, conjunction as aggregation, with maximum element top $\top$ as the empty obligation, and the minimal element bottom $\perp$ as the impossible obligation. Imposing the obligation $\top$ means that the action of the guard can be carried out without further undertaking, imposing $\perp$ that an action may not be carried out. Eliminations of contradicting elementary obligation combinations, such as "delete data within a week" and "keep data for a year" at the same time, can be achieved by excluding from the lattice all those obligation sets containing problematic obligations. The ruling of an ExPDT rule and the default ruling of the overall policy are specified by a tuple of obligations (*postiveObligation*, *negativeObligation*), each an element of the power set of $\tilde{O}$.

4.2 Semantics of a rule

The evaluation function of a query (also a tuple of user, action, data, and purpose) resulting in a ruling for a given policy defines the semantics of the policy language ExPDT. The required authorization and order rule modalities can be expressed. Actions cannot only be permitted but also forbidden or explicitly triggered. In addition to provisions, obligations as actions to be performed in future can also be imposed on the user.

The ExPDT language also allows the users the certain degree of freedom in adherence to the rule actions. While this always applies in case of a permit, users can decide whether they adhere to a prohibition or an order. If they do not, specified sanctions in the form of so-called negative obligations take effect. If these sanctions correspond however to the impossible obligation $\top$, adherence becomes necessary for the users, a rule circumvention impossible. The various rule modalities as well as the obligations and sanctions are mapped via the tuple of obligations of the ruling as shown in **Tab. 1** whereby the first obligation specifies the future additional actions and the second possible sanctions. Here are some examples:

- Permission: Employees are allowed to open a purchase order. Ruling: ($\top$, $\top$)
- Permission with obligation: Employees are allowed to open a purchase order but a supervisor is notified. Ruling: (notify, $\top$)

- Prohibition with sanction: Employees are not allowed to open a purchase order. If they disregard this prohibition and, however, open a purchase order, the supervisor will be notified. Ruling: (⊥, notify)
- Compulsory order: The administrator has to make a weekly backup. The sanction according to the impossible obligation makes adherence to this order indispensable. Ruling: (⊤, ⊥)

4.3 Semantics of a policy

The semantics of the policy language is determined through the evaluation function $\mathrm{eval}_\alpha(P,q)$ for a query q regarding a particular policy P and current assignment α of the contextual condition variables. Roughly, the function searches through the list of policy rules until a rule is matched by the query, i.e. all elements of the rule guard are either equal to the user, action, data, and purpose of the query or stand higher up in their corresponding hierarchy. Additionally, the condition of the rule must not evaluate to false using the current variable assignment. The complete evaluation works as follows:

1. Initialize the *result* with (⊤, ⊤) and preset evaluation status *v* to default.
2. Evaluate rules one by one according to their priority. If the rule's guard is matched by the query and...
   a) its condition evaluates to 1, return the conjunction of the rule's ruling and hitherto accumulated *result* as policy ruling and an evaluation status *v* of final.
   b) its condition evaluates to u, add rule's ruling to *result*, set the status *v* to applicable and proceed with the next rule.
3. If the status *v* is applicable, then return *result* as ruling and status *v*.
4. If the status is still default, no rule has matched and the default ruling is returned together with the status *v* default.

The case of incomplete context information resulting in an undefined condition value for a rule is taken into account by accumulating the ruling of such a rule with a possibly previous found ruling, i.e., conjunct both the positive obligations and the negative obligations, and proceeding with the evaluation. Hence, it is ensured that the evaluated ruling is possibly too restrictive due to the additional obligations, but never too weak.

Tab. 1 ExPDT codes the modalities into the ruling

Modality	Obligations	Sanctions	Ruling
Permission			(⊤, ⊤)
	O+		(O+, ⊤)
Prohibition			(⊥, ⊤)
		O-	(⊥, O-)
Order			(⊤, ⊥)
	O+		(O+, ⊥)
		O-	(⊤, O-)
	O+	O-	(O+, O-)
Error			(⊥, ⊥)

4.4 Policy operators

The extensive dragging along of the evaluation status *v* with its distinction of final, applicable or default ruling allows not only the handling of incomplete context information but also the definition of combination operators for modular specification and evaluation of policies despite their stub-behavior. The stub-behavior corresponds to the intention of the default ruling, i.e., to ensure a safe ruling until another rule matches. Therefore the refinement of a default ruling with an applicable or final one should be possible in the case of a policy combination. In ExPDT, two combination operators are defined: the conjunction $P_1 \wedge P_2$ thereby evaluates P_1 and P_2 with equal priority, while the composition $P_1 \,||\, P_2$ gives P_1 higher priority for the evaluation, therefore evaluates it first. For more detailed combination tables of rulings and generating algorithms, see (Raub 2004).

The ability to compare policies is essential for calculability, e.g. if a policy P_1 is to be replaced by a different policy P_2, when compliance requirements change. The starting point for a comparison of two policies usually is equivalence, i.e., both supply the same query results in any case, and the refinement that indicates whether one policy is more restrictive or specific than another (see, e.g. Backes et al. 2004). If this, however, is not the case, there is no indication as to which rules require further decisions. Therefore, in ExPDT the difference operator is defined. Given two policies P_1 and P_2 over compatible vocabulary, the difference $P_2–P_1$ is a mapping from P × P to a list of rules R that covers exactly those queries q and assignments α of conditional variables that result in a less restrictive ruling for P_2, so $(\mathrm{result}_i, v_i) = \mathrm{eval}_\alpha(P_i,q)$ for i ∈ {1, 2} and $\mathrm{ruling}_1 \nsubseteq \mathrm{ruling}_2$. For these, the difference rule list results in the same decisions as P_2. Thus, the difference operator reduces the regulation of the policy to become effective to precisely those rules describing all less restrictive situations that are of particular interest.

The difference rule list is constructed by the following plot: rules of both policies are looped through according to their priority, so that each rule of P_2 is compared with all rules of P_1. If such a comparison detects only equal or more restrictive situations with bigger scope or weaker conditions and obligations, the looping is continued with the remaining rules of P_1. If there are, however, such situations and if they are not captured by a following P_1-rule with lower priority, they are formally captured by a new rule that is appended to the difference result. Then, the looping is discontinued for the current P_2 rule and starts with the next rule anew. If all P_2 rules are examined, the construction of the difference terminates.

This rule list describes the functional difference of both policies, so that they are compared independently of their possible evaluation status *v*; the stub-behavior of the policies is not taken into consideration. For more detailed specification of this algorithm including the handling of conditions refer to Kähmer and Gilliot (2008). By means of the difference, the equivalence and refinement of two policies can be computed: if $P_2–P_1$ results in an empty list, P_2 describes less restrictive situations and P_2 is a functional refinement of P_1. If the difference of switched policies

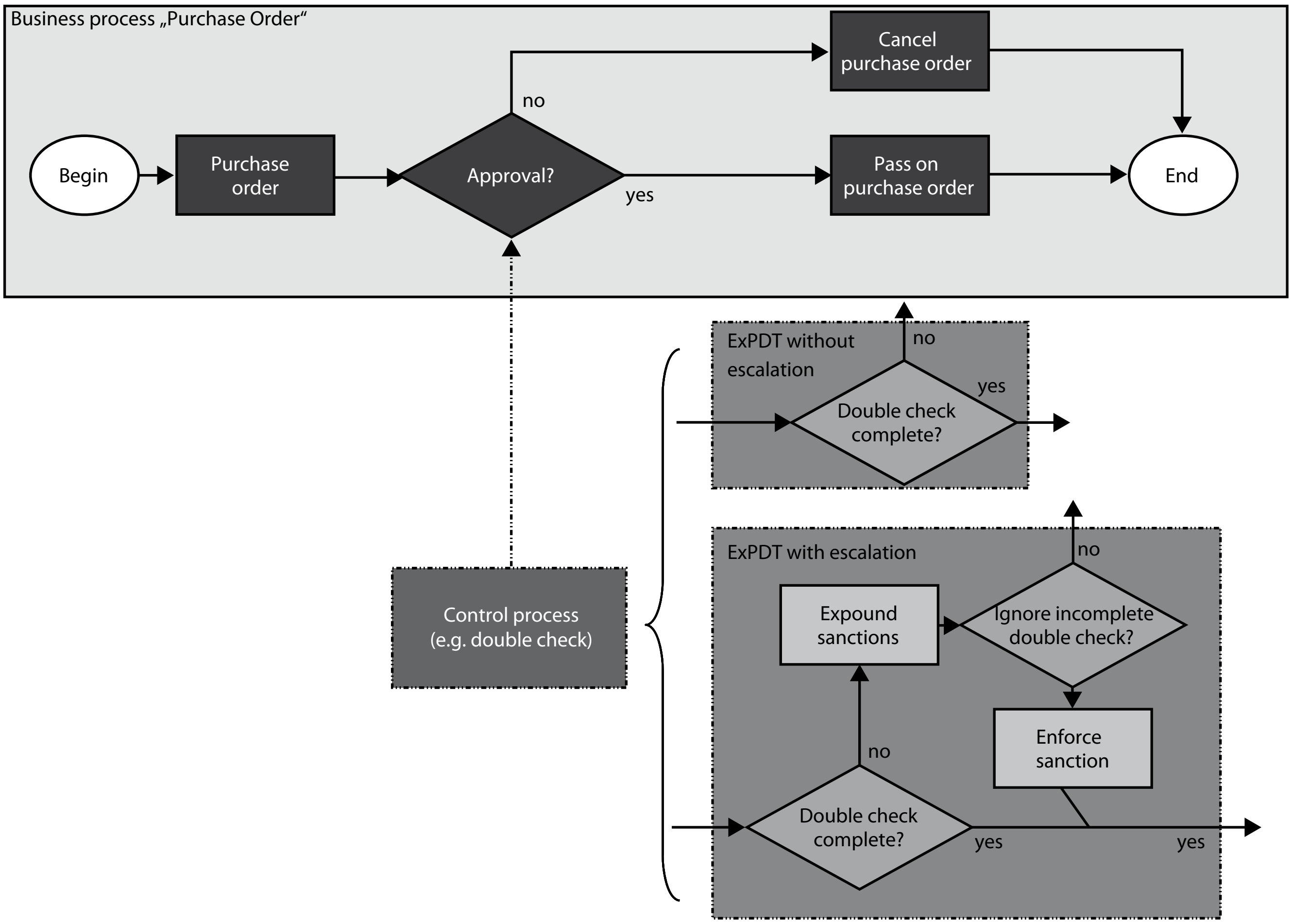

Fig. 4 Exemplified workflow with two alternative control processes

results in an empty rule list as well, P_1 and P_2 are functional equivalent.

5 ExPDT for escalation within business processes

The crucial characteristic of a policy language for compliance is the support of flexibility with sanctionability. In section 4.2 the corresponding formalism of ExPDT for optional decisions and sanctions has been presented. In the following, the practical use is discussed along with a simplified escalation scenario: for compliance reasons the exemplified workflow intends a double check authorization for every purchase order before it is passed on to a supplier (see **Fig. 4**). Rectangles represent single actions, a rhombus a decision, and the rectangle with the dashed line a control process. Currently, the double check control requires that two different officers check the purchase order; otherwise, the purchase in process cannot be completed. Transferring this example to ExPDT results in the following compliance rule:

(E_1, PassOn, Order, forPurchase), not(checkedBy(E_1,E_2) ∧ ($E_1 \neq E_2$)), (⊥,⊤)

This rule prohibits an employee E_1 from passing on a purchase order if it was not checked by a second employee E_2 who is different from E_1 and is visualized in Figure 4 in the upper control process *without escalation*. Formulized in actual but lengthier ExPDT OWL-DL syntax, a full example can be found in Kähmer (2007).

However, there might be exceptional situations that require fast adaptation, e.g., an opportunity to get discount for fast ordering or the time-critical replacement of expected shipping lost by accident to maintain the production process. If there is no second employee available for confirming the order, the termination of the order process according to the compliance policy might bring enormous loss. To consider such opportunities already within the control process, ExPDT offers the ability to integrate exceptions:

(E_1, PassOn, Order, forPurchase), not(checkedBy(E_1,E_2) ∧ ($E_1 \neq E_2$)), (⊥, notify)

This rule is visualized in the control process *with escalation* and differs from the former one by allowing employee E_1 to disobey the double check and to pass on the order. In this case, the underlying monitor has to ensure that the sanction is enforced and a supervisor is notified. Of course, "notifying" is only a first step to integrating consequences to actions into a compliance policy since the actual sanction is outsourced to the supervisor opposing automation of compliance. Sanctions such as ignoring the violation, halting the business process, or rolling back activities could also be integrated. The next outstanding step is the improvement of decision support for the employee by integrating information about the economic impact of interrupting a process and the risk of non-compliance into the control process. This requires the combination of risk management and compliance policies as well as their integration into business process models, e.g., by annotations that are expanded with risk types (Muehlen and Rosemann 2005).

6 Conclusion and outlook

Policies are the key to automating compliance: they bridge the gap between the compliance requirements and their realization within the IT systems. Policies determine to which degree these requirements can be formalized and automated. The ExPDT policy language presented, originally developed for specifying privacy policies, can also be used for compliance policies. In contrast to current policy languages, it is the first language exhibiting suitable expressiveness, flexibility with sanctionability, calculability, and modularity.

The integration of decision options with sanctions into policy rules, e.g. for escalating unexpected events in business processes, is technically no real challenge. However, this feature is seen as vital since it is the starting point for both preserving the situation specific adaptation of business processes and at the same time achieving automation of compliance. By enabling a "compliance by detection" approach the enormous effort of a pure "compliance by design" approach can be avoided since not all compliant situations have to be defined in advance. Beginning with the enforcement of the most important rules a stepwise enlargement of automated validation becomes possible.

A prototype of ExPDT has been implemented as proof of concept; however, an evaluation is the next outstanding step. Moreover, two main issues remain: firstly, obedience to the policy has to be realized on the monitor and IT system levels. This is a prerequisite for enlarging the part automatable and automated validation of compliance. Secondly, to take full advantage of the flexibility, sanctions have to be specified according to the context of the actual business process. For this, components for an accurate assessment of compliance risk in real-time have to be provided. Both issues, enforcement and assessment, are still open research fields.

References

Accorsi, R. (2008): Automated Privacy Audits to Complement the Notion of Control for Identity Management. In: Proceedings of the IFIP Con-

Zusammenfassung / Abstract

Stefan Sackmann, Martin Kähmer

ExPDT: Ein Policy-basierter Ansatz zur Automatisierung von Compliance

Unternehmen sehen sich steigenden Anforderungen aus neuen Gesetzen, regulatorischen Vorschriften, Standards, Governance und auch Verträgen gegenüber. Durch den Einsatz von Informationstechnologie kann die Validierung der Einhaltung solcher Regeln (Compliance) automatisiert und effizienter erreicht werden. Aktuelle Ansätze basieren im Wesentlichen auf Zugangskontrolle und der Dokumentation der tatsächlichen Nutzung von Daten sowie Durchführung von Prozessen. Damit können zwar einzelne Compliance-Anforderungen adressiert werden, ein effizienter IT-Einsatz erfordert jedoch einen allgemeinen Ansatz. Hierfür wird ein Rahmenwerk zur Automatisierung von Compliance vorgestellt. „Policies", wie sie aus der IT-Sicherheit bekannt sind, werden als Schlüssel zur Automatisierung von Compliance identifiziert, da sie eine Brücke zwischen nicht-technischen Compliance-Anforderungen und deren Umsetzung in IT-Systemen bieten. Es wird die Policy-Sprache ExPDT präsentiert und gezeigt, inwieweit diese zur automatisierten Einhaltung von Compliance-Anforderungen eingesetzt werden kann, ohne die situationsspezifisch erforderliche Adaptivität von Geschäftsprozessen zu gefährden.
Stichworte: Automatisierung von Compliance, Policy-Sprache, Risikomanagement, flexibles Geschäftsprozessmanagement

ExPDT: A Policy-based Approach for Automating Compliance

Remaining in compliance with growing requirements from new laws, regulations, standards, or contracts demands increasing IT support beyond simple reporting tools or archiving solutions. However, an efficient IT support of compliance management requires a more general approach. In this contribution, a framework for automating compliance is introduced. Policies are seen as the key to aligning non-technical compliance requirements to a technical IT system. The policy language ExPDT is presented and evaluated with regard to maintaining flexibility of business processes and validating compliance.
Keywords: automating compliance, policy language, risk management, flexible business process management

ference on Policies and Research in Identity Management, Springer, Berlin, pp. 39–48.

Agrawal, R.; Johnson, C.; Kiernan, J.; Leymann, F. (2006): Taming Compliance with Sarbanes-Oxley Internal Controls Using Database Technology. In: Proceedings of the 22nd International Confeence on Data Engineering (ICDE'06). IEEE Computer Society, Washington, DC.

Ashley, P.; Hada, S.; Karjoth, G.; Powers, C. et al. (2003): Enterprise Privacy Authorization Language (EPAL 1.2). Submission to W3C.

Bace, J.; Rozwell, C. (2006): Understanding the Components of Compliance. Gartner, Report G00137902.

Backes, M.; Karjoth, G.; Bagga, W.; Schunter, M. (2004): Efficient comparison of enterprise privacy policies. In: Proceedings of ACM Symposium on Applied Computing (SAC'04), Nicosia, pp. 375–382.

Bajaj, S.; Box, D. et al. (2006): Web Services Policy 1.2 – Framework (WS-Policy). http://www.w3.org/Submission/WS-Policy/, last access 2008-06-27.

Botan, I.; Kossmann, D. et al. (2007): Extending XQuery with Window Functions. In: Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB Endowment, Vienna, pp. 75–86.

Breaux, T. D.; Anton, A. I.; Karat, C.-M.; Karat, J. (2005): Enforceability vs. Accountability in Electronic Policies. Report TR-2005–47, North Carolina State University Computer Science.

Cannon, J. C.; Byers, M. (2006): Compliance deconstructed. In: CACM Queue 4 (7), pp. 30–37.

Cranor, L. F.; Dobbs, B. et al. (2006): The Platform for Privacy Preferences 1.1 (P3P1.1). W3C specification. http://www.w3.org/TR/P3P11/, last access 2008-06-27.

Cranor, L. F.; Langheinrich, M.; Marchiori, M. (2005): A P3P Preference Exchange Language 1.0 (APPEL). W3C Working Draft.

Delbaere, M.; Ferreira, R. (2007): Addressing the data aspects of compliance with industry models. In: IBM Systems Journal 46 (2), pp. 319–334.

Gallier, J. H. (1988): Logic for Computer Science. John Wiley and Sons, New York.

Giblin, C.; Muller, S.; Pfitzmann, B. (2006): From regulatory policies to event monitoring rules: Towards model driven compliance automation. IBM Research Zurich, Report RZ 3662.

Goedertier, S.; Vanthienen, J. (2006): Designing Compliant Business Processes with Obligations and Permissions. In: Proceedings of International Conference on Business Process Management (BPM06) Workshops. LNCS 4103, Springer, Berlin, pp. 5–14.

Hilty, M.; Basin, D.; Pretschner A. (2005): On Obligations. In: Proceedings of 10th European Symposium on Research in Computer Security (ESORICS 2005). LNCS 3679, Springer, Berlin, pp. 98–117.

Iliev, A.; Smith, S. (2005)*:* Protecting Client Privacy with Trusted Computing at the Server. Proceedings of IEEE Security & Privacy 3 (2), pp. 20–28.

ITGI (2007): COBIT 4.1, Framework, Control Objectives, Management Guidelines, Maturity Models. http://www.isaca.org/AMTemplate.cfm?Section=Downloads&Template=/MembersOnly.cfm&ContentFileID=14002, last access 2007-12-01 (free registration required).

Johnson, C. M.; Grandison, T. W. A. (2007): Compliance with data protection laws using Hippocratic Database active enforcement and auditing. IBM Systems Journal 46 (2), pp. 255–264.

Kähmer, M. (2007): ExPDT Ontologies and Examples. http://www.telematik.uni-freiburg.de/mitarbeiter/kaehmer/expdt/, last access 2008-06-27.

Kähmer, M. (2008): Extended Privacy Definition Tool – A Formalism for Specification and Comparison of Privacy Policies. PhD Thesis, University of Freiburg, to appear.

Kähmer, M.; Gilliot, M. (2008): Extended Privacy Definition Tool. In: Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI 2008), LNI, Springer, Berlin.

Karagiannis, D. (2008): A Business Process-Based Modelling Extension for Regulatory Compliance. In: Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI 2008), LNI, Springer, Berlin.

Klempt, P.; Schmidpeter, H.; Sowa, S.; Tsinas, L. (2007): Business Oriented Information Security Management – A Layered Approach. In: Proceedings of the 2nd International Symposium on Information Security (IS'07), Vilamoura, pp. 1835–1852.

Liebenau, J.; Kärrberg, P. (2006): International Perspectives on Information Security Practices. London School of Economics and Political Science, McAfee.

McGuinness, D. L.; van Harmelen, F. (2004): OWL Web Ontology Language – Overview. W3C recommendation. http://www.w3.org/TR/2004/REC-owl-features-20040210/, last access 2008.06.27.

Moses, T. (2005): eXtensible Access Control Markup Language (XACML), version 2.0, Oasis Standard. http://xml.coverpages.org/xacml.html, last access 2008-06-27.

Muehlen, M. zur; Rosemann, M. (2005): Integrating Risks in Business Process Models. In: Proceedings of the 16th Australasian Conference on Information Systems (ACIS 2005), Sydney.

Müller, G.; Sackmann, S.; Prokein, O. (2008): IT Security: New Requirements, Regulations and Approaches. In: *Frank-Schlottmann, F.* et al. (Eds.): Handbook on Information Technology in Finance, Springer, Berlin, pp. 711–730.

Namiri, D.; Stojanovic, N. (2008): Towards a Formal Framework for Business Process Compliance. In: Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI 2008), LNI, Springer, Berlin.

OCG (2007): ITIL V3 – Service Life Cycle, Office of Governance Commerce, http://www.itil.org/en/itilv3-servicelifecycle/index.php, last access 2008-06-27.

Raghupathi, W. R. P. (2007): Corporate governance of IT: a framework for development. In: Communications of the ACM 50 (8), pp. 94–99.

Raub, D. (2004): Algebraische Spezifikation von Privacy Policies. Master's thesis, Uni. Karlsruhe (in German).

Raub, D.; Steinwandt, R. (2006): An Algebra for Enterprise Privacy Policies Closed Under Composition and Conjunction. In: Proceedings of International Conference on Emerging Trends in Information and Communication Security (ETRICS), LNCS 3995, Springer, Berlin, pp. 130–144.

Sackmann, S.; Kähmer, M.; Gilliot, M.; Lowis, L. (2008): A Classification Model for Automating Compliance. In: Proceedings of the IEEE Conference on E-Commerce Technology (CEC08), to appear.

Sackmann, S.; Strücker, J.; Accorsi, R. (2006): Personalization in Privacy-Aware Highly Dynamic Systems. In: Communications of the ACM 49 (9), pp. 32–38.

Sadiq, S. W.; Governatori, G.; Namiri, K. (2007): Modeling Control Objectives for Business Process Compliance. In: Proceedings of the 5th International Conference Business Process Management (BPM 2007). LNCS 4714, Springer, Berlin, pp. 149–164.

Schneider, F. B.; Morrisett, G.; Harper, R. (2001): A Language-Based Approach to Security. In: Informatics: 10 Years Back, 10 Years Ahead. LNCS 2000, Springer, Berlin, pp. 86–101.

Schneider, F. B. (2006): Computability classes for enforcement mechanisms. In: ACM Transactions on Programming Languages and Systems 28 (1), pp. 175–205.