RZ 3616 (# 99626) 07/21/05
Computer Science 14 pages

Research Report

Regulations Expressed As Logical Models (REALM)

Christopher Giblin[1], Alice Y. Liu[2], Samuel Müller[1], Birgit Pfitzmann[1] and Xin Zhou[2]

[1]IBM Research GmbH
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

[2]IBM China Research Lab
Haohai Building
Beijing 100085
China

Regulations Expressed As Logical Models (REALM)

Christopher Giblin[1], Alice Y. Liu[2], Samuel Müller[1],
Birgit Pfitzmann[1], Xin Zhou[2]

[1] IBM Zurich Research Lab, Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland
[2] IBM China Research Lab, Haohai Building, Beijing 100085, China

Abstract. Recent years have seen a number of high-profile incidents of corporate accounting fraud, security violations, terrorist acts, and disruptions of major financial markets. This has lead to a proliferation of new regulations that directly impact businesses. As a result, businesses, in particular publicly traded companies, face the daunting task of complying with an increasing number of intricate and constantly evolving regulations. Together with the growing complexity of today's enterprises this requires a holistic compliance management approach with the goal of continually increasing automation.
We introduce REALM (Regulations Expressed as Logical Models), a metamodel and method for modeling regulations and managing them in a systematic lifecycle in an enterprise. We formalize regulatory requirements as sets of compliance rules in a novel real-time temporal object logic over concept models in UML, together with metadata for traceability. REALM provides the basis for subsequent model transformations, deployment, and continuous monitoring and enforcement of compliance in real business processes and IT systems.

1 Introduction

Recent years have witnessed a growing amount of regulatory requirements directed towards businesses, particularly publicly traded companies. Prominent examples of such regulations include the Sarbanes-Oxley Act, the U.S. Patriot Act, Basel II, anti-money laundering regulations and various de facto standards such as the International Financial Reporting Standards (IFRS). Not only the sheer number of relevant laws and standards but also the complexity of individual regulations is drastically increasing. Consequently, affected businesses are confronted with the task of adapting to new and evolving regulatory requirements. While this process is initially driven by regulators, companies increasingly recognize this challenge as an opportunity to improve operational transparency, traceability and reporting.

We propose a systematic compliance management approach for addressing the increasing breadth and complexity of regulatory requirements on businesses. Our approach is based on a lifecycle view of compliance management, which

begins with the formalization of regulations by means of REALM, a dedicated metamodel for the specification of regulatory requirements.

The use of a common metamodel such as REALM has the advantage that different regulations, to a large extent, can be captured and formalized using a shared language and semantics. Figure 1 provides a visual representation of the metamodeling approach as compared with addressing each regulation as a single case. With the latter method, a worst case $M \times N$ mappings are required to implement M regulatory requirements over a set of N target systems. Systems include business processes, IT applications and systems, and people issues such as education material. In contrast, the metamodeling approach theoretically involves only $M + N$ transformations.

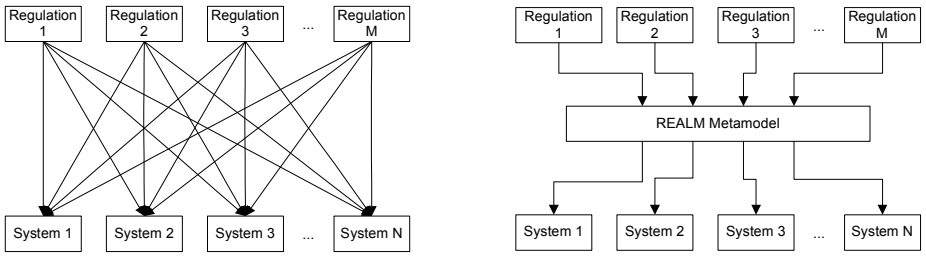

Fig. 1. Advantage of using the REALM metamodel for business compliance

However, the primary benefit can be perceived in the reuse and sharing of common terms, semantics, models and tools in understanding and enforcing regulatory requirements. An additional advantage of systematic compliance management is that it offers traceability from the regulations to the systems; this enables businesses to demonstrate to auditors or regulators how they achieve compliance, which is increasingly required by new regulations.

A REALM model consists of the following parts:

- *Concept model.* We represent the domain of discourse of a regulation by a concept model expressed as UML class diagrams using a UML profile developed for REALM.
- *Compliance rule set.* We provide a set of logical formulae expressed in a novel real-time temporal object logic that represent the regulatory requirements, e.g., sequencing and time constraints on actions, over the elements from the concept model.
- *Metadata.* We capture meta-information about the structure of the legal source as well as lifecycle data such as the enactment date.

The main novelty of our work is our use of a temporal object logic for regulation modeling. Our need for a temporal logic arose from our focus on proactively enforcing regulations at the level of business processes. In contrast, prior work focusses on comparing laws, applying them to finished cases, and on legal discourse, so that temporal aspects are not of specific importance. Moreover, the logic itself is new.

A real-time logic is required since many regulations contain concrete time constraints, such as data retention duration, the obligation to respond to requests within a certain time, or the requirement to perform certain actions contingent on other actions occurring or not within a specific past period.

REALM models of regulatory requirements are part of and support a compliacne management lifecycle. Naturally, a regulation of interest must be initially stored, analyzed and understood. REALM models consisting of a concept model, a compliance rule set, and metadata can then be derived. These abstract models are ultimately transformed to implementation artifacts such as business process definitions, data retention policies, access control lists, or correlation rules, while preserving traceability to relevant passages in the respective regulation sources. The artifacts are deployed into the business and IT infrastructure of the enterprise. Compliance with the regulatory requirements can be monitored and enforced leveraging the capabilities of the involved target technologies and the common models shared by all stages of the compliance management lifecyle.

2 Related Work

Regulatory ontologies and logics have been commonly used in the development of expert and formal dialogue systems and more recently in compliance-assistance solutions [1–4]. Exemplary models of concrete regulatory requirements include the formalization of the British Nationality Act, Dutch tax law and the Ontario Freedom of Information and Protection of Privacy Act, and the representation of environmental regulations [5–8]. While we have found that regulations usually contain many timing requirements with references to real time, we are not aware of any efforts to formalize regulations concentrating on real-time temporal logics as a central building block.

UML, the Unified Modeling Language, which we use for the concept models, is a widely used standard for modeling object-oriented systems and has also been used for ontology specification [9]. UML profiles are an extension mechanism for defining domain-specific modeling elements by specializing UML metaclasses. Using the Object Constraint Language (OCL), UML models can be complemented by precise constraints. The official UML and OCL standards, however, provide no inherent support for expressing temporal predicates. Research into temporal extensions to the OCL includes [10–12]. However, these extensions seem not capable of specifying absolute real-time references as often needed in a legal context and may be impractical when global constraints over independent objects or over objects with non-intersecting lifetimes are needed. Furthermore, [12] does not support real time at all.

Propositional temporal logics (PTL) have mainly been proposed for the specification of dynamic systems such as programs [13]. For real-time aspects, we build upon the Timed Propositional Temporal Logic (TPTL), which employs a novel restricted use of temporal quantification, by introducing a so-called freeze quantifier that binds a variable to the time of a particular formula evaluation [14]. Approaches in logic to combine temporal and object aspects [15, 16] do not allow real-time features as we need them.

As to the intended deployment and transformation of regulation models, the Model Driven Architecture (MDA) [17] is a framework in which software implementations are generated primarily from models.

3 REALM-based Compliance Management Lifecycle

In order to enhance an enterprise's ability to adapt to new regulations and standards, we propose a compliance management lifecycle process based on REALM models. The formalization of regulatory requirements is one part of this larger process. Other activities include the initial selection of relevant regulations and the eventual deployment, monitoring, and enforcement of their respective compliance. We now describe this lifecycle for one enterprise and one new regulation. Abbreviations of this process are possible if another party, e.g., a standards body, has already formalized the regulation.

1. *Determination of scope.* First, the scope of the given regulation is determined and its relevance and potential impact on the enterprise are evaluated. This may restrict which parts of a given regulation must be modeled formally. It may also suggest which parts of the enterprise need to be analyzed in greater detail for compliance with this regulation.
2. *Formalization of regulatory requirements.* Next, those portions of the regulation in scope for the enterprise are formalized into a REALM model, i.e., a concept model, a compliance rule set, and metadata as described in the introduction. This process will typically produce two models:
   (a) *Immediate model.* First one creates a model that stays close to the terms and requirements of the regulation; we call this the immediate model.
   (b) *Refined model.* Many terms in a regulation are vague or coarse-textured, obliging the enterprise to choose an instantiation in line with current best practices and its own strategic goals, practices and compliance management objectives. We call the result of this phase, in which specific deployment values are selected, the refined model.

   Both immediate and refined models are REALM models, based on the same metamodel.
3. *As-is analysis of the enterprise.* In order to assess the impact of the selected regulatory requirements in detail, a thorough as-is analysis is needed. The results are models of the parts or the enterprise in scope of the regulation, e.g., business processes, applications, data models, and IT resources.
4. *Gap analysis.* Given rigorous and concise representations of the regulation and the potentially affected parts of the enterprise, the impact on the existing processes, data and resources is assessed by comparing the new REALM models with the as-is situation. In other words, a gap analysis is performed between the current environment and its potential implementation of the regulation.
5. *Deployment.* Informed by the gap analysis the REALM models are deployed into the identified target systems. Even where no gaps currently exist, formal deployment may prevent new gaps from arising when processes or IT

systems change in the enterprise. Depending on the type of requirements captured, target systems can be process definitions, access control lists, privacy policies, storage policies or correlation rules for event monitoring. The models are mapped onto these artifacts by means of model transformations. These diverse deployment procedures are represented in Figure 2. In terms of MDA, one can regard the REALM models as platform-independent models (PIM) and the target models as platform-specific models (PSM). An example of industrial business process modeling, a key application area, is given in [18].

6. *Compliance monitoring and enforcement.* After a REALM model has been deployed into a target system, compliance is sometimes ensured automatically. For instance, a business process execution engine ensures that activities are executed in the order specified in the process definition; hence ordering constraints deployed by adapting that process definition are automatically enforced. However, due to the inherent possibility of human or system error, compliance must almost always also be monitored and enforced in real time. For instance, a business process definition may state that an activity only takes two days, but unexpected circumstances may hold it up. Hence a rule with a time constraint would also be deployed into a correlation engine for monitoring. Furthermore, if the correlation engine detects a timeout, its only means to resolve the issue may be to send an alert to the responsible compliance officer; in this case compliance is finally enforced on a higher level and not always with fully predetermined procedures.

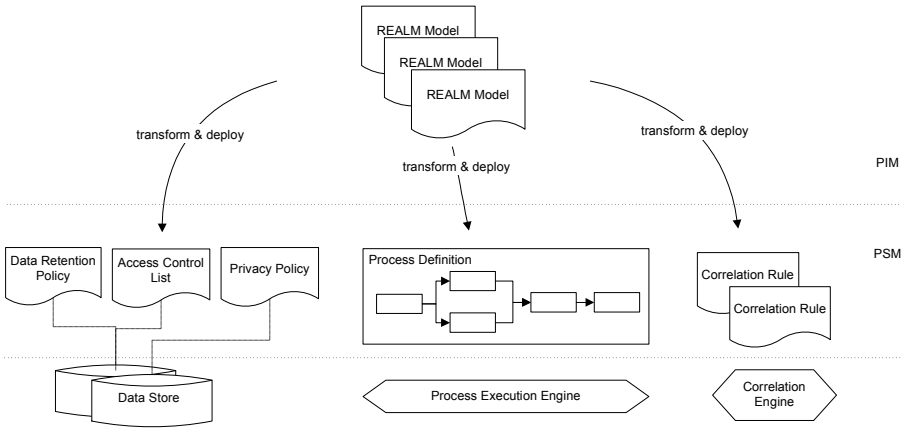

Fig. 2. Exemplary model transformations for deployment

The need for a refined model containing specific assignments from a set of possible instantiations of vague terms is a characteristic of modeling regulations for enterprise deployment. This contrasts to deciding compliance of finished cases, legal discourse, or law comparison, which can all be done with the immediate model alone. For instance, if a regulation states that a response to certain

requests must be given within a "reasonable time", then an enterprise has to decide how fast it can provide such answers and how much earlier it wants to start an internal escalation process. Hence the immediate model contains the concept "reasonable time", while the refined model contains the concrete time assignment, "2 days". Retaining both models provides traceability in case the interpretations of or best practices for the vague term change, or the enterprise later reconsiders its initial decision, as in the case of our example, of 2 days.

Whether the refinements are compliant is ultimately a legal interpretation for which an enterprise typically involves internal or external legal consultation. Capturing regulations formally does not obviate this need. However, formalizing a regulation into the immediate and refinement models does facilitate identifying where such advice is needed and where decisions are applied, enhancing transparency by moving interpretation out of the hands of individual process owners or system administrators.

4 REALM Metamodel

In order to formalize regulations in the necessary detail, we need a language that is expressive enough to capture the variety of requirements occurring in actual regulations We formalize the objects and relationships occurring in a regulation in a concept model (in other words a domain ontology) and the constraints on these concepts by rules in a real-time temporal object logic. The structural information of the actual regulations is captured with a metadata model.

4.1 Concept Model

The REALM concept model captures the concepts and relationships occurring in a regulatory domain. For example, a regulation requiring that *'Banks must verify the identity of each customer'* includes the concepts *bank*, *verify*, *identity*, and *customer* and implicit relations such as between *bank* and *verify* and between *verify* and *identity*.

REALM provides a set of predefined abstract types such as *Person* and *Organization*, *Process* and *Action*, *Artifact* and *Resource*, *Principle* and *Purpose*, as well as *Location*, and *Cost*. We also include certain basic types such as String, Integer, and Boolean, thereby ensuring compatibility with UML. Concrete concepts like *Customer* should, wherever possible, be subtypes of the predefined types, here *Person*. Individual instances like specific customers of a specific bank are instances of these types. Using the predefined abstract types, together with the following predefined relations, has two main benefits: easier and less error-prone construction of the following temporal formulae, and easier specification of automated model transformations as needed for deployment.

The concrete syntax of the predefined types and relations is given by a UML profile. In a regulation model based on this profile, a class stereotype denotes that a concept is of a certain predefined type and association stereotypes denote predefined relations. Our choice to make actions a separate predefined type in REALM, although object methods might seem more natural in the context of

UML, was made with deployments in mind: Top-level deployments will often be to business process or workflow models, where actions are also separate.

Relations in regulations can be understood in the same way as in logic and UML. An equivalent term in logic is predicate; related terms in UML are associations and association classes. REALM provides several predefined relations over the predefined types. For example, *Do* is a predefined abstract relation between a person or organization and an action. We write it $Do(a, b)$, where a is an instance of a subtype of *Person* or *Organization* and b an instance of a subtype of *Action*. This predicate evaluates to true if a executes action b at the point in time where the predicate is evaluated. Other important examples of predefined relations are $On(a, b)$, $Input(a, b)$, and $Output(a, b)$ where a is an action and b an artifact; they have the natural meanings. Similar natural relations have been defined between most pairs of the predefined types.

Many time constraints in laws refer to the beginning or the end of an action. We use subscripts '$_S$' and '$_F$' for this. More precisely, we attach these subscripts to the relations. For instance,

$$Do_S(a, b) \quad \text{and} \quad Do_F(a, b)$$

evaluate to true only at the point in time where the person or organization a starts executing action b or finishes it, respectively. This notation with just one subscript is for the standard case of a relation with one action parameter.

For readability of the later formulae, we define a syntactic scheme for composing predefined binary relations into n-ary relations. Given relations $R_1, \ldots, R_n$, we define

$$R_1 \cdots R_n(a_1, \ldots, a_{n+1}) \;:\Longleftrightarrow\; R_1(a_1, a_2) \wedge \ldots \wedge R_n(a_n, a_{n+1}),$$

where $R_1 \cdots R_n$ is the string concatenation of the relation names. Clearly this is only defined if the second parameter of R_i has the same type as the first parameter of R_{i+1} for $i = 1, \ldots, n-1$. Our naming scheme of predefined relations allows unique parsing of the concatenated names by the initial capitals. For instance, we can now write $DoOn(bank, open, account)$ for $Do(bank, open) \wedge On(open, account)$.

4.2 Compliance Rule Set

Compliance rules in REALM are expressed using a real-time temporal object logic. Recall that this logic as such, as well as the use of a temporal logic for regulation modeling, are important novel aspects of REALM. A REALM compliance rule set is based on a REALM concept model.

Structural requirements typically do not need a rule in the REALM compliance rule set because they can be expressed with existing features of UML within the concept model, in particular with multiplicities. For instance, one can thus model that a bank must have exactly one auditing committee, or that each account has a personal identification record attached to it.

We build upon the Timed Propositional Temporal Logic from [14]. The advantage of this treatment of real time is a good balance between expressiveness

and complexity [19]. We add the object model instead of atomic propositions, the specialization to actions with lifetimes, and syntactic abbreviations such as the use of time with different units. The basis are timed state sequences. The following definition assumes a definition of a set States of states of our concept model, which we have only sketched above, and builds on a basic discrete time type Time (e.g., milliseconds).

Definition 1 (Timed State Sequence). *A timed state sequence* $\rho = (\sigma, \tau)$ *over a REALM concept model consists of a state sequence* $\sigma = \sigma_0\sigma_1\sigma_2\ldots$ *and a time sequence* $\tau = \tau_0\ \tau_1\ \tau_2\ldots$ *with* $\sigma_i \in \mathsf{States}$ *and* $\tau_i \in \mathsf{Time}$ *for all* $i \geq 0$. *The time sequence must be monotonic, i.e.,* $\tau_i \leq \tau_{i+1}$ *for all* $i \geq 0$, *and provide progress, i.e., for all* $t \in \mathsf{Time}$ *there exists* $i \geq 0$ *such that* $\tau_i > t$. ◇

Temporal formulae are based on state formulae, i.e., formulae evaluated on one state σ_i of a timed state sequence. State formulae are based on the relations from the concept model, with the usual logical connectives and with OCL navigation among the objects. For instance, a notation *a.customer* for an account instance a refers to the customer owning account a if there is an association from class *Accounts* to class *Customer* with multiplicity 1. We model instances as passivated after the end of their lifetime, but still available for navigation.

Temporal formulae are built with the usual (not real-time) temporal modalities such as $\Box$ (always), $\Diamond$ (eventually) and $\blacklozenge$ (sometimes in the past), as well as so-called freeze quantifiers and relations on times. A freeze quantifier corresponds to the introduction of a variable for a point in time. For instance, a formula part $\Diamond\, t.\phi$ means that eventually formula ϕ will hold, and we introduce the time variable t for this point in time. Time relations within ϕ can refer to t, e.g., to express that t is at most 2 days later than some other time t', introduced similarly with a freeze quantifier. Examples are given in Section 5.

As an example of how temporal modalities are defined, we only present the definition of the always modality without freezing. The fact that formula ϕ is true for the timed state sequence ρ is denoted by $\rho \models \phi$. The state sequence ρ^i for $i \geq 0$ is defined by deleting the first i elements from ρ. Then

$$\rho \models \Box\,\phi \ :\Longleftrightarrow\ \forall i \geq 0 : \rho^i \models \phi.$$

4.3 Metadata

Besides concept models and compliance rule sets, REALM models may include metadata providing information about the modeled regulatory requirements. Two kinds of metadata are required:

- *Structural metadata* link the formalized regulatory requirements to their source. For instance, REALM model elements may be annotated with the name of the regulation, the modeled paragraph or section, a plain text description, and a hyperlink to an online source or interpretation.
- *Lifecycle constraints* include the enactment date of the modeled regulation, the validity duration, the expiry date, or validity constraints of individual compliance rules or model elements.

As an example, a concept model may be annotated with a metadata element of type *Section*, whereas an individual compliance rule may be annotated with a metadata element of type *ExpiryDate* if this specific rule expires before other rules of the same compliance rule set.

Relating REALM model elements to structural metadata is fundamental to the traceability and comprehensibility of formalized regulatory requirements. Associating model elements with lifecycle constraints provides traceability to instantiations of temporal values defined in the refined model.

REALM does not specify a single legislation metamodel but rather integrates with existing models. Examples of existing metamodels defining structural metadata for legislative texts include MetaLex [20], PAPI [21] and EnAct [22]. MetaLex, for example, also provides lifecycle metadata for regulation parts through attributes such as *date-enacted*, *date-effective* and *date-repealed*.

5 Example

In Figure 3 we present a fictional regulation to demonstrate how a set of regulatory requirements can be captured using the REALM metamodel. The example closely resembles the latest requirements made on financial institutions under the U.S. Patriot Act, Section 326, in particular the rule that implements it, 31 CFR paragraph 131.121.

Banks must implement procedures for verifying the identity of each customer; these procedures must ensure that the bank knows the true identity of each customer. At a minimum, the bank must obtain the following information prior to opening an account:

1. *Name;*
2. *date of birth;*
3. *residential address;*
4. *identification number.*

The bank must verify the identity of each customer, using the information obtained in accordance with the above requirements, within a reasonable time after the account is opened.
The bank must also implement procedures for responding to circumstances in which the bank cannot ensure that it knows the true identity of the customer. These procedures should describe when the bank should close an account, after attempts to verify the customer's identity have failed.
The bank must implement procedures for making and retaining a record of all information obtained according to the above requirements.
The bank must retain the recorded information for five years after the date the account is closed.

Fig. 3. Exemplary regulation text following 31 CFR paragraph 131.121

We now formalize these regulatory requirements. The REALM concept model is shown in Figure 4. Recall that the concrete syntax is a UML class diagram

using the REALM UML profile, with stereotypes denoting predefined types and relations. Let us explain a few non-obvious choices in this concept model: We model the retention of an account as an explicit action because it may involve active steps, such as ensuring continued accessability, even if the software systems change in the five years after the account is closed. We model names etc. as attributes of both the customer and a record. The former denote the true attributes of this customer, the latter the information that the customer provides but that may still need verification. The attribute *successful* of the action type *VerifyIdentity* is by default initialized to false and becomes true if the correctness of the identity was established to the satisfaction of the bank.

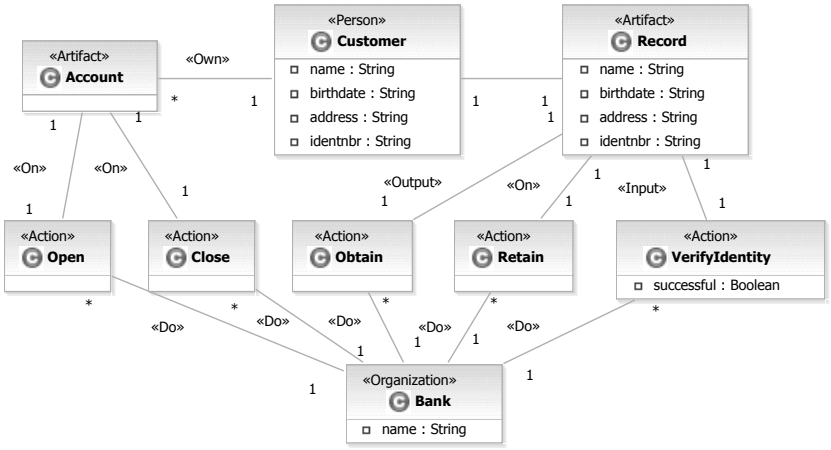

Fig. 4. REALM concept model for the example regulation

In the following, we present the REALM compliance rule set. Figure 5 illustrates the timing constraints on the actions for one account. Some details will become clear when the formulae are explained. Note that this is not an activity diagram; all the relations between actions which have no explicit constraints can be different than drawn here. One can in principle turn every set of constraints into an activity diagram with many alternatives, but as typical regulations consist of individual constraints, it is natural and useful for traceability to start by modeling the given constraints one by one as formulae. We plan to use time constraint diagrams as patterns for formula editing in the future. There is a certain similarity to the graphical interval logic of [23].

$$
\begin{aligned}
&\forall bank \in Bank,\ open \in Open, \exists obtain \in Obtain,\ r \in Record : \\
&\Box\ Do_F(bank, open) \\
&\quad \rightarrow \blacklozenge\ DoOutput_F(bank, obtain, r) \qquad (1) \\
&\qquad \wedge\ r = open.account.customer.record \wedge\ r.name \neq null \\
&\qquad \wedge\ r.birthdate \neq null \wedge\ r.address \neq null\ \wedge\ r.identnbr \neq null
\end{aligned}
$$

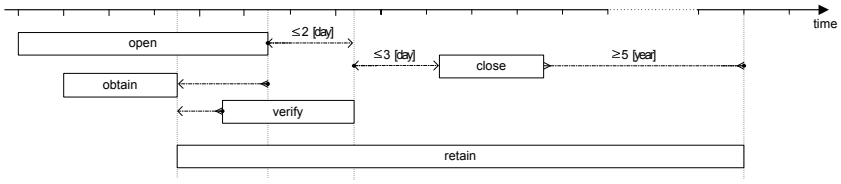

Fig. 5. Time constraints on the involved actions for one account

Formula (1) expresses the requirement that before a new account is opened, the bank must at least obtain the name, the date of birth, the address and an identification number of the customer. As to action durations, we required that the end of the action *obtain* is before the end of *open*. According to the concept model, the information is gathered in a record; we require that all four elements are non-null at the end of the action *obtain*. The fact that the record is about the customer opening the account is reflected by the equality about r; here we used OCL navigation. The temporal structure of the formula can be read as follows: "Whenever ($\Box$) the bank finishes opening an account, then sometimes in the past ($\blacklozenge$) it finished obtaining a record such that ..."

$$\begin{aligned}&\forall bank \in Bank,\ open \in Open,\ a \in Account\ \exists verify \in VerifyIdentity:\\ &\Box\, t_{open}.\ DoOn_F(bank, open, a)\\ &\quad \rightarrow \Diamond\, t_{verify}.\ DoInput_F(bank, verify, a.customer.record)\\ &\qquad \wedge\ t_{verify} - t_{open} \leq\ 2_{[day]}\end{aligned} \tag{2}$$

Formula (2) states that whenever a bank has opened a new account eventually it has to finish the verification of the identity of the customer, based on the data collected in the process of opening the account. The formula further formalizes that the verification action needs to be completed within reasonable time, which has been interpreted to mean at most 2 days in the given example. Thus this formula belongs to the refined model in the sense of Section 3.[3]

$$\begin{aligned}&\forall bank \in Bank,\ a \in Account,\ verify \in VerifyIdentity:\\ &\Box\ t_{verified}.\ DoInput_F(bank, verify, a.customer.record)\\ &\qquad \wedge\ verify.successful = false\\ &\rightarrow\ \Diamond\ t_{closed}.\ DoOn_F(bank, close, a)\\ &\qquad \wedge\ t_{closed} - t_{verified} \leq\ 3_{[day]}\end{aligned} \tag{3}$$

Formula (3) models that a bank must respond to circumstances where it cannot successfully verify a customer's identity and has to close a tentatively opened account. In our refined model we require that this happens within three days.

$$\begin{aligned}&\forall bank \in Bank,\ obtain \in Obtain\ \exists retain \in Retain:\\ &\Box\ Do_F(bank, obtain)\ \rightarrow\ DoOn_S(bank, retain, obtain.record)\end{aligned} \tag{4}$$

[3] Recall that "." after a temporal operator and a variable is part of freeze quantification conserving the evaluation time. Otherwise it is OCL navigation.

Fig. 6. Annotating REALM models with metadata

Formula (4) expresses that a bank has to retain the record of obtained customer information.

$$\begin{aligned}&\forall bank \in Bank,\ a \in Account,\ retain \in Retain\ \exists close \in Close :\\ &\Box\ t_{delete}.\ DoOn_F(bank, retain, a.customer.record)\\ &\quad \rightarrow\ \blacklozenge\ t_{close}.DoOn_F(bank, close, a)\\ &\qquad \wedge\ t_{delete} - t_{close} \geq\ 5_{[year]}\end{aligned} \tag{5}$$

Formula (5) contains the constraint that a customer record may only be deleted if the account has been closed for at least 5 years.

$$\begin{aligned}&\forall bank \in Bank,\ c \in Customer,\ r \in Record,\ verify \in VerifyIdentity :\\ &\Box\ (DoInput_F(bank, verify, r)\\ &\qquad \wedge\ verify.successful = true\ \wedge\ c = r.customer)\\ &\quad \rightarrow\ r.name = c.name\ \wedge\ r.birthdate = c.birthdate\\ &\qquad \wedge\ r.address = c.address\ \wedge\ r.identnbr = c.identnbr\end{aligned} \tag{6}$$

Formula (6) finally states that if the bank considers a verification successful then the record entries are the correct customer data. We also made this a temporal formula because the example regulation does not require that the bank continually watches whether the real customer data change. This requirement, in contrast to the previous ones, cannot be deployed in the strict sense, because the real customer data are out of the enterprise's scope of control. We formalize it nevertheless in the immediate model for traceability of the informal measures to be taken.

To conclude the picture, Figure 6 depicts an instance model of how a REALM compliance rule set can be annotated with metadata.

6 Conclusion

We have introduced REALM, a metamodel for formally expressing regulatory requirements with special emphasis on use in proactive compliance management in enterprises. A REALM model of a regulation consists of three pillars: a concept model of the terms in the regulation, a compliance rule set in a novel real-time temporal object logic, and metadata designating the source regulation and validity dates. Using an example based upon the U.S. Patriot Act, we demonstrated that temporal aspects are important and that a real-time logic is needed. Temporal aspects are also the first aspects that need to be considered when adapting business processes. REALM is embedded into the larger context of an integrated compliance management process, which tracks regulatory

requirements across their entire lifecycle, lends itself to model-driven transformation and deployment and allows for continuous compliance monitoring and enforcement.

Acknowledgments

We thank Günter Karjoth, Luke O'Connor, Matthias Schunter, Naishin Seki, Markus Stolze, Akihiko Tozawa, and Michael Waidner for their valuable contributions and comments, and June Y. Felix, Mark Greene, and Jürg von Känel for overall support.

References

1. McCarthy, L.T.: A language for legal discourse – i. basic features. In: Proc. 2nd International Conference on Artificial Intelligence and Law (ICAIL '89), ACM (1989) 180–189
2. Prakken, H.: On dialogue systems with speech acts, arguments, and counterarguments. In: Proc. of JELIA'2000, The 7th Workshop on Logic for Artificial Intelligence. Springer Lecture Notes in AI, Springer Verlag (2000) 224–238
3. Ryan, H., Spyns, P., Leenheer, P.D., Leary, R.: Ontology-based platform for trusted regulatory compliance services. In: On the Move to Meaningful Internet Systems 2003: OTM Workshops 2003. Volume 2889 of Lecture Notes in Computer Science., Springer (2003) 675–689
4. Kabilan, V., Johannesson, P., Rugaimukamu, D.M.: Business contract obligation monitoring through use of multi tier contract ontology. In: On the Move to Meaningful Internet Systems 2003: OTM Workshops 2003. Volume 2889 of Lecture Notes in Computer Science., Springer (2003) 690–702
5. Sergot, M.J., Sadri, F., Kowalski, R.A., Kriwaczek, F., Hammond, P., Cory, H.T.: The british nationality act as a logic program. Communications of the ACM **29** (1986) 370–386
6. van Engers, T.M., Gerrits, R., Boekenoogen, M., Glasse, E., Kordelaar, P.: Power: using UML/OCL for modeling legislation - an application report. In: Proc. 8th International Conference on Artificial Intelligence and Law (ICAIL '01), ACM Press (2001) 157–167
7. Powers, C., Adler, S., Wishart, B.: EPAL translation of the The Freedom of Information and Protection of Privacy Act (2004) IBM and Information and Privacy Commissioner/Ontario, `http://www.ipc.on.ca/docs/EPAL%20FI1.pdf`.
8. Kerrigan, S., Law, K.H.: Logic-based regulation compliance-assistance. In: Proc. 9th International Conference on Artificial Intelligence and Law (ICAIL '03), ACM Press (2003) 126–135
9. Cranefield, S., Purvis, M.: UML as an ontology modelling language. In: Proc. of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends. Stockholm. Sweden. Volume 18 of Proc. 16nd International Joint Conference on Artificial Intelligence., CEUR Publications (1999)
10. Flake, S., Mueller, W.: An OCL extension for real-time constraints. In: Object Modeling with the OCL. Number 2263 in Lecture Notes in Computer Science, Heidelberg, Germany, Springer Verlag (2002)

11. Flake, S., Mueller, W.: Past- and future-oriented time-bounded temporal properties with OCL. In: Proc. of the 2nd International Conference on Software Engineering and Formal Methods (SEFM 2004), Peking, China, IEEE Computer Society Press, Los Alamitos, USA. (2004)
12. Ziemann, P., Gogolla, M.: An OCL extension for formulating temporal constraints. Technical Report 1/03, Universität Bremen (2003)
13. Pnueli, A.: The temporal logic of programs. In: Proc. 18th IEEE Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society Press (1977) 46–57
14. Alur, R., Henzinger, T.A.: A really temporal logic. Journal of the ACM **41/1** (1994) 181–204
15. Sernadas, A., Sernadas, C., Costa, J.F.: Object specification logic. Journal of Logic and Computation **5** (1995) 603–630
16. Distefano, D., Katoen, J.P., Rensink, A.: On a temporal logic for object-based systems. In: Fourth International Conference on Formal methods for open object-based distributed systems IV, Kluwer Academic Publishers (2000) 305–325
17. Miller, J., Mukerji, J., (ed.): MDA guide version 1.0. (2003) omg/2003-06-01, `http://www.omg.org/cgi-bin/doc?omg/03-06-01`.
18. Mitra, T.: From business modeling to Web services implementation: Part 1: Modeling a business process (2005) IBM developerworks, `http://www-128.ibm.com/developerworks/websphere/library/techarticles/0502_mitra1/0502_mitra1.html`.
19. Alur, R., Henzinger, T.A.: Real-time logics: Complexity and expressiveness. Information and Computation **104/1** (1993) 35–77
20. E-POWER Consortium: Metalex, version 1.02 (2003) `http://www.metalex.org`.
21. Zeni, F., McGrath, S., Hatter, C.: Pan African Parliamentary Interoperability (PAPI), Report and Documentation (2004) United Nations Department of Economic and Social Affairs, `http://www.parliaments.info/PAPI/`.
22. Arnold-Moore, T., Clemes, J.: Connected to the law: Tasmanian Legislation using EnAct. Journal of Information Law and Technology (2000) `http://www2.warwick.ac.uk/fac/soc/law/elj/jilt/`.
23. Dillon, L.K., Kutty, G., Moser, L.E., Melliar-Smith, P.M., Ramakrishna, Y.S.: A graphical interval logic for specifying concurrent systems. ACM Transactions on Software Engineering and Methodology **3** (1994) 131–165