Preface

Business rules and external regulations are increasingly becoming important in information systems engineering. Systems, processes and services have to operate according to the rules, they have to be easily or dynamically adaptable in case of changes, and the rules and regulations have to be identifiable and traceable. This has strong implications for the architecture, modeling, management, and execution frameworks of enterprise systems.

Contracts, rules, standards and regulations are everywhere and apply to a large variety of fields: financial services, health, insurance, e-business, enterprise application integration, privacy, transport, safety, security. Because of their growing complexity, frequent changes, integration, globalization, and on the other hand the need for instant change and flexibility, the importance of dealing with rules and regulations in information systems is increasing dramatically. The objectives of the workshop are mainly to share research and experience on methods used to model, validate and verify, deploy and adapt information systems based on extensive existing rules and regulations.

Modeling and deploying rules and regulations of various kinds in enterprise systems, is a highly interdisciplinary challenge. This workshop tries to provide a forum for researchers from diverse backgrounds (formal systems, enterprise architecture, business process management, regulatory compliance, distributed systems, business rules). Through this interdisciplinary workshop, we expect to encourage fruitful exchanges between these scientific communities in order to advance the field of information systems engineering.

June 2008

Jan Vanthienen
Stijn Hoppenbrouwers
Régine Laleau

Program Chairs
ReMoD'08

Organization

Workshop Organizers

PC and Workshop Chair:	Jan Vanthienen (Katholieke Universiteit Leuven, Belgium)
Co-Chairs:	Stijn Hoppenbrouwers (Radboud University Nijmegen, The Netherlands)
	Régine Laleau(University Paris-Est, France)

Program Committee

Grigoris Antoniou (Greek Foundation for Research and Technology, Heraklion, Greece)
Alcedo Coenen (CIBIT, Bilthoven, The Netherlands)
Robert Darimont (Respect-IT, Belgium)
Guido Governatori (University of Queensland, Australia)
Leo Hermans (Everest B.V., den Bosch, The Netherlands)
Stijn Hoppenbrouwers (Radboud University Nijmegen, The Netherlands)
Régine Laleau (University Paris-Est, France)
Michel Lemoine (Onera Toulouse, France)
Yves Ledru (LIG, University Grenoble, France)
Christophe Mues (University of Southampton, UK)
Tony Morgan (Neumont University, USA)
Fiona Polack (University of York, UK)
Erik Proper (Radboud University Nijmegen, the Netherlands)
Silvie Spreeuwenberg (LIBRT, Amsterdam, The Netherlands)
Shazia Sadiq (University of Queensland, Australia)
Jan Vanthienen (Katholieke Universiteit Leuven, Belgium) (Chair)
Gerd Wagner (Brandenburgische Technische Universität Cottbus, Germany)
Hans Weigand (University of Tilburg, The Netherlands)

Rule-Based Service Composition and Service-Oriented Business Rule Management

Hans Weigand, Willem-Jan van den Heuvel and Marcel Hiel

INFOLAB, Tilburg University, Warandelaan 2, Tilburg, The Netherlands
H.Weigand@uvt.nl, wjheuvel@uvt.nl, m.hiel@uvt.nl

Abstract. As business processes change constantly, there is a growing need for adaptive composite services. Unfortunately, existing service composition languages and techniques result in rather brittle and rigid processes, whose services live in a straitjacket. In this paper, we propose a rule-driven approach for service composition that is purely declarative, highly adaptive and integrated in a truly service-oriented approach to business rule management.

1. Introduction

Today's business environment dictates organizations to be agile so they are able to accommodate their business processes to rapidly changing market conditions, including updated or new legislations and regulations, swiftly changing consumer demands and novel technological innovations, e.g., new mobile platforms. Service Oriented Architecture captures an emerging paradigm that is quickly gaining broad industry acceptance, and enables the development of a new breed of (cross-) enterprise applications that are comprised of loosely coupled services, which holds the promise that these applications can be modified and/or extended on the fly..

One of the key impediments towards realizing this vision, unfortunately, is that currently services are predominantly composed using block-structured and graph-based languages, notably BPEL, resulting in static and brittle composite services, although some work has been done on trying to make them a bit more adaptable, e.g., [1]. Composite services that are developed in this way are liable of intermingling process logic with business rules, providing the perfect ingredients for unmanageable and rather repellent process/rule spaghetti. This has become even more problematic as companies have begun to apply languages such as BPEL for very agile, real-world applications, and have observed that rules are in fact much more dynamic than business processes. Consequently, updating these rules that are deeply buried in the scattered process definitions has quickly grown into a complex, labor-intensive and cumbersome task.

It has been suggested that business rules can be separated from the BPEL code in a kind of aspect-oriented flavor [2]. Although this alleviates the management problem to some extend, adaptations are still only possible as long as they concern the content of pre-identified business rules that fit into the fixed BPEL frame.

In this paper, we argue that business rules can be used in service composition without the need of such a BPEL frame, thus increasing the adaptability of the orchestration significantly. At the deployment level, we introduce a CA-rule engine that supports rule-based service composition. To keep the business rules manageable themselves, we describe a service-oriented approach.

This paper is organized as follows. In the following section we will introduce a realistic running example that motivates the rule-based approach. In section 3, we will elaborate on this approach and how it fits in a service-oriented architecture. In section 4, we introduce the FARAO approach towards service composition that is based on the use of a CA-rule engine, and analyze to what extent business rule compliance can be realized in this framework. The last section summarizes the main findings of our work, and sketches directions for future research.

2. Motivating Case Study

MultiTech (fictitious name) is a wholesaler SME that buys and sells mobile phones. Its primary business process revolves around (re-)selling mobile phones, which it acquires from various international vendors. In this fictional, yet realistic, case study we exemplify business services, rule services and the actors invoking them, concentrating on the purchase-and-payment cycle of MultiTech.

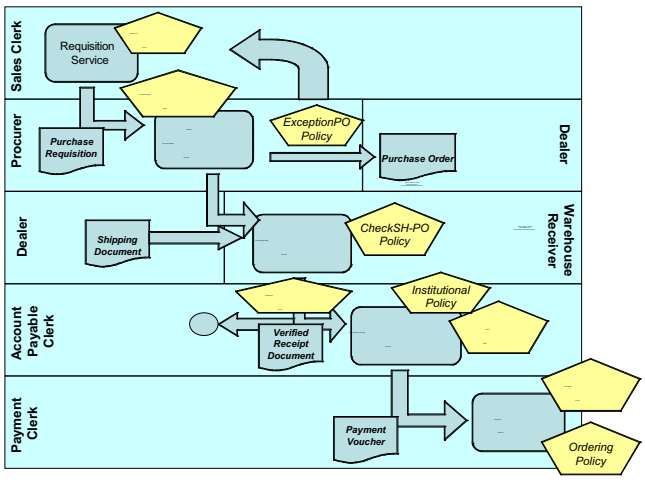

Fig. 1 Swimlane model of the Purchasing and Payment Processes

This cycle is organized as follows. The cycle starts with an authorized sales clerk requisitioning mobile phones. After his permission to requisite a particular product is ascertained, he may issue a purchase requisition to the inventory manager.

The inventory manager then sends the verified purchase requisition to the purchase agent, whose authorization to deal with this particular kind of order is then checked (`PermissionPolicy`). The purchase agent transforms the purchase requisition in a

purchase order, while ensuring that the used master data complies to the supplier's product code; in case of a problem, an exception is raised (`ExceptionPO-Policy`). Also, the internal stock level is checked; a policy describes the stock replenishment level. Thereafter, he sends the purchase order to the vendor, and issues two additional copies of the purchase order, one to the warehouse clerk and one to the payment clerk.

Service	Business policy	Definition
RequisitionService	PermissionPolicy	The sales clerk should be authorized to requisition particular (quantities of) products, e.g., Sales Clerk "Klaus" is allowed to requisition not more than 1000 mobile phones a time.
CreatePurchaseOrderService	PurchaseOrder-Policy	A sales order can only then be created if, and only if, the purchase requisition is complete, the order would cause stock dropping below the allowed replenishment level, and he is authorized.
CreatePurchaseOrderService	ExceptionPO-policy	A purchase order cannot be created if the ordered product does not exist cf. the master data; in this case, a new purchase requisition is required.
CheckAgainstPOService	CheckSHPO-Policy	An incoming shipment document must be checked against a purchase order document within 24 hours after receipt.
PurchaseProcessService	VerificationPolicy	If the verified document is not valid, then the ordering process is terminated.
PurchaseProcessService	InstitutionalPolicy	A purchase must be financially reported on a real-time basis (cf. institutional policies declared in Sarbanes-Oxley)
PurchaseProcessService	SODPolicy	The actor invoking this service should be another than the actor invoking the requisition process to obey segregation of duties.
PaymentService	EscalationPolicy	In case of dubious credits, the payment should be escalated to a human manager.
PaymentService	OrderingPolicy	A payment can only be invoked, after the ordered products have been received, and checked against the PO.

Table 1. MultiTech Business Policies

The vendor invoices MultiTech by sending an invoice along with its shipment to the warehouse clerk. After receipt of the goods and its accompanying shipment document, the warehouse clerk uses the purchase order and the receiving report to verify the correctness of the delivery. Then he sends the verified receiving report to the accounts payable clerk.

If the purchase is not valid, the process is terminated according to the `VerificationPolicy`, e.g., in case the budget has been exceeded. Also, the accountant is liable of reporting any payments directly to the government cf. Sarbanes-Oxley act, section 409 (`InstitutionalPolicy`). Also, to ensure segregation of duties, and circumvent potential fraud, the accountant cannot be the same person as the sales clerk (`SODPolicy`).

The accountant creates and sends a payment voucher to the payment manager, together with the verified receiving report, purchase order, and verified invoice; only after all this information is available the payment can be processed (`OrderPolicy`). Note that in case of excessively large purchase orders in a specific time period, the payment process is escalated to the management for further consideration (`EscalationPolicy`).

The example makes clear that business policies are first-class citizens in the modern enterprise and directly influence business services.

3. Business Rules and Service Composition

In this section, we first define and classify business rules and policies. In 3.2, we review previous work on rule-based service composition, and in 3.3 a new service-oriented approach to business rule management is described.

3.1 Business Rules and Policies

We follow the fundamental distinction between business rule and policy [3]. Policies arise from internal sources such as business needs, from corporate-level guidance, from external laws and regulations, and from ethical motivations. Based on the OMG Business Motivation Model (BMM) such policies "govern or guide an enterprise," specifying business design aspects that complement information and operation models [4].

Business policies are usually written in natural languages to cater for evaluation by domain experts, viz. business analysts. That evaluation assumes human interpretation, as the ambiguities of natural languages must be resolved and application of policies to specific business contexts generally requires analysis of impacts, consequences, and trade-offs. Thus, policies provide guidance but insufficient detail for implementation. Considerable research has been conducted into the conceptualization of business policies using languages such as ORM ([5], [6]), ILOG and OCL.

The application of policies in specific contexts leads to business rules, meaning highly structured, discrete, atomic statements "carefully expressed in terms of a vocabulary" [4] to enforce constraints (integrity rules), to deduce new information (derivation rules) or to trigger actions on satisfied conditions (reaction rules) [7]. If a business rule “defines and constrains some aspect of the business” [4], we can distinguish between norms or constraints (*constraining*) and definitions (*defining*). The former category can, without loss of generality, be expressed as prohibitions, indicated in deontic logic with the F modality, whereas the latter typically take the form of derivation rules.

Following [3], we posit that business rules are about business requirements, rather than about execution. They model “what” is required, rather than “how” it should be implemented. Hence we distinguish business rule languages from (executable) production rule languages such as ECA-Rules [8] and “IF…THEN” (CA)-rules [9]. SBVR is an OMG proposal for the representation of business rules in Structured English [10].

In order to operationalize constraints, often information has to be added. According to [11], a constraint (called norm frame in their terminology) should consist of 5 elements: a norm condition, a violation condition, a detection mechanism, a sanction and repairs. The *violation condition* is a formula denoting the state when the norm is violated. Although in simple cases, there is a 1-1 relationship between norm and violation condition – for example, if the norm is F(α) for an observable action α then the violation condition is DONE(α) – it is not possible to derive one from the other in all cases. For example, when a certain action is not defined in the operational context, or when the norms cannot be interpreted in isolation. The *detection mechanism* provides the procedure necessary to determine whether the violation holds at a certain moment. For example, the OBL(α BEFORE d), expressing that action α must be performed before deadline d, can be checked efficiently by a trigger that fires when the deadline d has been reached (based on a clock signal), and that checks DONE(α). Note that the detection mechanism here is more specific than the violation condition. The *sanction* is an action that is to be performed when a violation has been detected, whereas a *repair* is an action that tries to undo or compensate a violation. Following this approach, it is clear that the translation from norm to executable rule is not a simple transformation.

In SOA, a series of (partially overlapping and conflicting) specifications and standards have been proposed that can be used to render business policies and rules. WS-Policy entails a family of semantic-agnostic languages to express assertions about constraints and capabilities of service end-points. These constraints and capabilities can be either very generic, or domain-specific, e.g., defining security-, transaction or reliability policy constraints (cf., WS-Security, WS-Transactions and WS-Reliaiblity). KAoS [12], Cassandra [13] and Rei [14] denote executable policy specification languages from the semantic web community, which are based on RDF and OWL. RuleML [15] and the Semantic Web Rule Language [16] constitute two general-purpose executable rule languages.

3.2 Rule-driven Service Composition –state of the art

Service composition sits at the heart of the Service Oriented Architecture, allowing service requesters to assemble several services that meet their requirements, into composite services. Unfortunately, languages like BPEL, suffer from severe problems, especially with regard to their flexibility and adaptability. Instead, rules have been investigated as an alternative declarative approach, boasting the following key advantages:

- **Intuitive formal semantics**: Rule-based languages exploit a limited set of primitives with the formality of an underlying logical and/or mathematical framework, and the quality of being meaningful to the domain expert.
- **Direct support for business policies**: business rules *enact* business policies in that policies can be transformed into business rules in a straightforward and transparent manner. These business rules are externalized and managed separately from the processes in which they are applied
- **Flexibility**: rule-based compositions are believed to be more flexible than BPEL-like compositions, given their ability to pursue alternative execution paths

in case a particular execution path fails, without having to redefine the composite service and redeploy it on a service engine.
- **Adaptability**: given the declarative nature of rule-enabled service compositions, they can be modified and/or extended to accommodate context-specific situations, e.g., in terms of external services or the deployment platform.
- **Reusability**: Since rules isolated from the business process context, they can be more easily reused in other service application contexts.

Recently, considerable efforts have been invested in rule-engines to support service compositions. In particular, we herein wish to mention the following key contributions. Firstly, in [7], a service-oriented rule engine was introduced that allows enterprises to access business rules by invoking distributed service-enabled ruleML engines that sit at the service supplier's service end-point. [17] introduces an alternative service execution environment in which rules can be defined, and subsequently injected in WSDL specifications, after which they can be deployed on a service executor. [2] introduces AO4BPEL, an aspect-oriented extension to BPEL that is able to weave business rules into BPEL frames. Alternatively, in [18] an approach is suggested to incorporate business rules in BPEL specifications, while enforcing them in rule engines that work in concert with BPEL engines, and coordinate themselves through an ESB. This approach basically works as follows; an interceptor is used to catch incoming and outgoing BPEL service invocations (activities), after which a business rule broker service is initiated, through which applicable business rules can be accessed. Depending on the interceptor mode (before/after), the BPEL activity is either fired or the control flow is continued.

In addition to research prototypes that were developed for the purpose of validation, several service-oriented rule engines are nowadays available, viz. the Oracle Fusion Middleware Rule Engine. This rule engine allows specifying business rules as ILOG facts that can be inserted into BPEL specifications. This is achieved by allowing users to map BPEL variables to facts in a rule repository.

3.3 A service-oriented approach to business rule management

Business rules are an example of crosscutting concerns, especially those encountered in composite services with a coordinating function, and run the risk of getting scattered over the system. In a service-oriented approach it is possible to encapsulate a certain business policy into a service. The advantage is that this service can be called from anywhere, and rule redundancy can be avoided (cf. [19]). However, given the presumed autonomy of services in SOA, it is not immediately clear how compliance to such rule services is ensured. This situation is similar to the situation in Multi-Agent Systems (MAS), where autonomy is a fundamental property of agents as well. In MAS, the problem is addressed by an institutional approach. As one of the earliest papers on this topic, [20] described a market place architecture for agents that draws on exception handling third parties that act like "institutions" as we know them from human societies (e.g. notary, registry). To realize such an institutional approach, [20] suggests three concepts. First, each agent in the system is assigned a "sentinel" that mediates the interactions of the agent with other agents. These sentinels monitor message traffic, detect violations to commitments, and apply resolution handlers. The

sentinel incorporates domain-independent exception handling expertise. Secondly, the system includes institutional or ancillary services such as a reputation service that can be called upon by the sentinels. Thirdly, agents cannot just enter the system; the only way to join is to register at the Registrar service, who only allows entrance after having assigned the agent a sentinel. Is it possible to use this solution approach in SOA?

As we just noted, the second element of the solution can be easily applied in SOA. We can introduce institutional services as services, and as far as they represent not only *mechanisms* (which is the focus of [20]) but also *policies*, it is possible to implement them using a rule-based approach. The first element requires more attention, as sentinels are clearly not part of SOA. However, there are recent developments within SOA that provide each service with a *service manager* (e.g. [21], [22]). This service manager can be realized as a service and has the possibility to adapt the service via a management interface (MOWS). In ASOA [22], the service manager follows a monitor-plan-act cycle as envisioned in autonomic computing, which is close to the specified behavior of the sentinel.

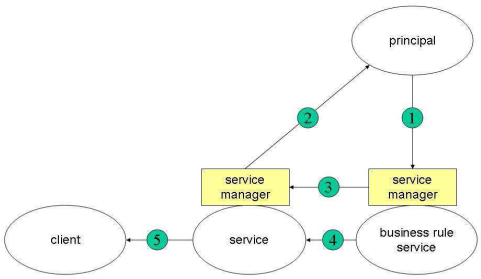

Fig. 2. Service-oriented business rule management, an institutional approach

In order to apply an institutional solution approach and to solve the compliance problem, we make the fundamental assumption that services have a dual orientation. One orientation is the client-orientation that lies at the heart of SOA. However, implicitly or explicitly, there is also another orientation towards someone who wants this service to be delivered. This party can be called the *principal*, and the relationship is one of *delegation*. Delegation means that a party wants to achieve something – typically providing a service to some customer – but rather than doing it himself, he asks another party ("agent") to do it on his behalf. Conceptually, the relationship can be characterized as a service offered by the "agent" to the principal. The delegation provides us with a mechanism to introduce services. A service X is introduced by a service provider – which we identify with a service manager – by replying to a request from the principal to deliver service X.

Now it becomes clear how a service can be bound to a business policy. When the principal requests the service manager to deliver service X, the request contains a reference to all the policies that should be respected as well. By adopting the request, the service manager commits himself to respect these policies. Within these policies,

a distinction can be made between generic rules, such as for detecting norm violations and reporting, and context-specific rules; the latter can be offered as separate services, and a generic rule only says that the service manager should call these services for this or that occasion. Fig. 2 describes the process of service introduction. A certain business rule service is assumed to exist representing some policy, for example, the PermissionPolicy of MultiTech. The principal uses this service (1). The service manager of the "agent" provides a service delivery service to the principal (2) upon his request. In performing this service, it uses and invokes the business rule service (3). Typically, in the case of a composite service, this implies that the service execution itself involves the business rule service (4), by orchestration. This all being in place, a client can call the service (5).

This scenario offers a solution to the compliance problem, but it does not assume a central Registrar authority. Each principal can impose its own policies. However, what the principal imposes is not an autonomous decision, as it depends on the policies imposed on him by powers above him.

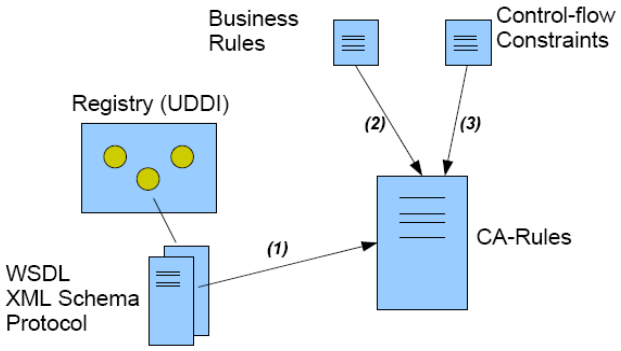

Fig. 3 The FARAO approach towards service composition

4. Framework for Designing, Reusing and Evolving Business Rules in Service Compositions

FARAO stands for a FrAmewoRk for Adaptive Orchestration. The ultimate goal of FARAO is to support the development of adaptable service orchestrations and to prepare for adaptivity by providing a manageability interface to a service manager, such as described in xSOC [23] and ASOA (Adaptive Service Oriented Architecture) [22]. Fig. 3 conceptualizes the relations between the ingredients of our service orchestration. Given a set of services to be orchestrated, the designer starts with retrieving the interface and data descriptions, typically from the registry. From these descriptions, Condition-Action (CA) rules are derived that manage the data flow. We have chosen for CA-rules rather than ECA-rules as the latter introduce more dependencies between the rules. In step (2), these rules are extended with business rules that typically steer the decisions in the orchestration. In step (3), the designer has the opportunity to add additional control-flow constraints, if required. In ASOA, all

three steps will be delegated to the service manager who executes them autonomously or semi-autonomously.

4.1 Data dependencies

The FARAO lifecycle model starts with a data-driven approach where the process structure is derived essentially from the data dependencies between the services involved in the orchestration. For example, if an orchestration involves both an Inventory service that returns, among others, the actual price of the product and a message to the customer with a quote, there is a data dependency between the two services that (implicitly) enforce that the former precedes the latter. If there is no data dependency between two services, there is no need to schedule one before the other, and by refraining from an arbitrary ordering we increase flexibility.

As hinted at in the above, we generate a CA-rule for each message that the orchestrator has to send. From the WSDL of the service in question, we derive the structure of the document it expects. Range restrictions on the message elements are copied into the conditions of the CA-rule. If there is not a range restriction, a NOT NULL condition is generated. In the action part, we put a *send* action that takes the service and its input document as parameters.

Rules refer to data items. In order to increase adaptability, we require that the orchestration is based on a shared ontology. WSDL-S [24] provides a mechanism to add semantics to web services. This allows, among others, that the message elements of the service are mapped to a given ontology. By requiring the WSDL descriptions of the services to be semantically annotated, we can let the rules refer to data items in terms of the shared ontology. In this way, changes in the service interface do not influence directly the orchestration, as long as the services adhere to the shared ontology.

4.2 Inference rules

The CA rules generated from the data dependencies provide an executable orchestration, but it only works well to the extent that the data items in the documents are seamlessly integrated. This is not always the case: sometimes an inference step is needed. For example, if one data item is "credit rating" and another is "creditworthy", then we need a rule to correlate the two that essentially prescribes when a person is creditworthy (for example, if credit rating > 10).

The general format of these inference rules is:

IF <condition> THEN $a_1 = v_1$ .. $a_j = v_j$

Technically, these inference rules are not CA-rules. We coerce them into CA rules by giving the consequent part an assignment interpretation: if the conditions are satisfied, then assign values v_1 .. v_j to the variables a_1 .. a_j

Example: After the accounts payable clerk has got the information from the CheckAgainstPO service it must decide whether or not to further process the order. The business rules for this decision can be formulated as follows:

```
Rule 1: IF verified-shipping-doc != "ok" THEN shippingstatus = reject
Rule 2: IF verified-shipping-doc == "ok" THEN shippingstatus = accept
```

These rules are to be used in combination with the rule (for the action) that processes the verified shipping document. This rule contains the condition that `shippingstatus = accept`. The rules 1 and 2 can be fed directly into the CA-engine, but they may also be part of a business rule service included in the orchestration. In the latter case, they are much easier to maintain of course. In a real-life implementation, a combination of the two approaches can consist in a caching solution, where the rules from the business rule service are moved to the CA-engine of the service temporarily. This approach saves on the communication overhead attached to service invocations. The cache has to be refreshed when the business rules are modified at the source.

4.3 Control flow constraints

The most prominent control flow constraint is the precedence constraint, where a certain service can only be executed after some other service has happened or some state has been reached. In Linear Temporal Logic, such a precedence constraint is usually described as: ¬β `UNTIL` α, where α and β are arbitrary propositions. In the case of orchestration, we restrict ourselves to constraints in which β is a service call. Then the meaning of the constraint is that this service cannot be called as long as α is not true.

For example, `¬send(PaymentVoucher) UNTIL (PurchaseProcessing = "ok")` which enforces payment voucher is not issued to the service `PaymentService` before the `PurchaseProcessing` service has been concluded (MultiTech `OrderingPolicy`). A fundamental restriction of SOA is that services are autonomous, so the orchestrator cannot verify himself whether a certain service is finished; he is dependent on return messages of that service. If no return message is returned, there is no way to enforce precedence. Hence we restrict the α part of the precedence constraint to propositions on data (and not on the completion of some service as such). Within the present context, the β part is restricted to the event of sending a document.

In FARAO step (3), the control flow constraints are inserted into the CA-rules. In step (1), a CA-rule has been generated for each outgoing document. Let this rule be of the form "`IF D THEN send(M)`", and let a control flow constraint be `¬send(N) UNTIL C`". If M=N, then we derive the rule "`IF D AND C THEN send(M)`".

Atomic prohibitions such as the `PermissionPolicy` in MultiTech can be injected into the CA-rule condition in a similar way (not worked out for lack of space).

4.4 Business Rules Implementation in FARAO

As far as business rules are concerned, we made a distinction between definitions and constraints. In the above, we have indicated how definitions can be incorporated in (the CA-engine of) FARAO as inference rules. Precedence constraints can be injected into the CA-rules. However, the interpretation of a norm frame requires more than precedence constraints. Not all norms are enforceable. In that case, the norm frame includes detection and remedy parts, among others. In FARAO, these can be directly implemented as CA-rules, although preferably, the service orchestration itself only

contains detection rules and the remedy is left to the service manager or an institutional service.

A type of rules not mentioned so far are permissions. If we follow the "everything is permitted unless forbidden" regime, permissions are not strictly needed. However, often permissions function as "second-order" constraints, determining which prohibitions can be added and which can not. In other words, they prohibit certain prohibitions, in which case they can be treated as constraints.

5. Conclusions

At present, organizations typically rely on block-structure, light-workflow specifications such as BPEL, to realize their business processes as composite Web-services. Unfortunately however, this style of composition assumes that at run-time, a detailed and complete process layout is "carved in stone", making its adaptation cumbersome, complex and time-consuming, requiring re-compilation of the process engine, and causing disruptions in, potentially mission-critical, business processes.

In this paper, a declarative and rule-driven framework to dynamic service composition, labeled "FARAO", is introduced, while its ramifications are further explored and illustrated with a realistic case study. The "heart-and-soul" of FARAO constitutes business rules that prescribe the way in which services can actually be aggregated dynamically into processes. The business rules are fed into the engine in a service-oriented way, that is, by a principal requesting a service delivery in accordance with given policies and by the service manager accepting this request. The business rules are maintained and updated outside the operational services. Given the platform independence offered by SOC, this can be anywhere inside or outside the company.

Our current research efforts concentrate on the implementation of the FARAO framework to experiment with rule-based service composition. A topic for future research is the mapping of our business rule representation to standard business rule languages, and to define a transformation from this language to the operational FARAO environment using a model-driven engineering approach.

References

[1] Reichert, M., Rinderle, S.: "On design principles for realizing adaptive service flows with bpel". In: Weske, M., Nttgens, M., (eds), EMISA. Volume 95 of LNI., GI, pp.133–146, 2007.
[2] Charfi, A. and Mezini, M., "AO4BPEL: An Aspect-oriented Extension to BPEL", World Wide Web, V.10, nr. 3, pp.: 309-344, 2007.
[3] N. Nayak et al., "Core business architecture for a service-oriented enterprise", IBM Systems Journal, Vol 46, No. 4, pp. 723-742, 2007
[4] The Business Motivation Model, Business Rules Group and the Object Management Group (OMG), http://www.businessrulesgroup.org/bmm.shtml
[5] Hargreaves, A. "Expressing Business Rules with Object Role Modeling", Proceedings of the 17th NACCQ, 2004.
[6] Halpin, T. "Business Rules and Object Role Modeling", Database Programming and Design, Oct 1996.

[7] Nagl, C., Rosenberg, F., Dustdar, S., ViDRE - A Distributed Service-Oriented Business Rule Engine based on RuleML. In: Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06), 16-20. October 2006, Hong Kong, China.
[8] Chen, L et al., "ECA Rule-based workflow modeling and implementation for service composition", IEEE Transactions on Information & Systems, Vol. 89, Nr. 2, pp. 624-630, Feb. 2006.
[9] Geminiuc, K. "A Services-Oriented Approach to Business Rules Development", In: SOA Best Practices: SOA Cookbook, Oracle, 2007. Available at: http://www.oracle.com/technology/pub/articles/bpel_cookbook/geminiuc.html .
[10] Semantics of Business Vocabulary and Business Rules (SBVR), The Object Management Group, Sept 2006, http://www.omg.org/cgi-bin/apps/doc?dtc/06-08-05.pdf
[11] Vasquez-Salceda, J., H. Aldewereld, F. Dignum, Implementing norms in multiagent systems. In: G. Lindemann, J. Denzinger, I. Timm, R. Unland (eds), Multi-Agent System Technologies. LNAI 3187, Springer Verlag, pp. 313-327, 2004.
[12] Bradshaw, J.M., S. Dutfield, B. Carpenter, R. Jeffers, and T. Robinson. "KAoS: A Generic Agent Architecture for Aerospace Applications", in Proc. of the CIKM'95 Intelligent Information Agents Workshop. Baltimore, MD, 1995.
[13] Becker, M.Y.; Sewell, P Cassandra: distributed access control policies with tunable expressiveness**.** Proc. Fifth IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY 2004), pp. 159-168, 2004.
[14] Kagal, L., Finin, T., Joshi, A., 'A Policy Language for a Pervasive Computing Environment', Proceedings of IEEE 4th Int. Workshop on Policies for Distributed Systems and Networks (POLICY 2003), Lake Como, Italy, 2003.
[15] Schroeder, M. and G. Wagner (Eds.): Proc. of the Int. Workshop on Rule Markup Languages for Business Rules on the Semantic Web., Italy, June 2002. CEUR-WS Publication Vol-60
[16] Horrock, I. et al., "SWRL: Semantic Web Rule Language", http://www.daml.org/rules/proposal/ , Dec. 2004.
[17] Kamada, A. and M. Mendes, "Business Rules in a Service Development and Execution Environment", Proc. of the Int. Symposium on Communications and Information Technologies, pp. 1366-1371, IEEE, 2007.
[18] Rosenberg, F. and S. Dustdar, "Business rules integration in BPEL: A service-oriented approach", Proceedings of the 7th International Conference on E-Commerce Technology, IEEE, 2005.
[19] Rosenberg, F. and S. Dustdar, "Towards a distributed service-oriented business rule system", Proceedings of the Third European Conference on Web Services (ECOWS'05), IEEE, 2005.
[20] Dellarocas, C., Klein, M., and Rodriguez-Aguilar, J. A. An exception-handling architecture for open electronic marketplaces of contract net software agents. In Proc. of the 2nd ACM Conf. on Electronic Commerce, EC '00, pp.225-232. ACM, 2000.
[21] Papazoglou, M.P. and W.J. van den Heuvel, "Service oriented architectures: approaches, technologies and research issues", VLDB Journal, Vol.16(3):389-415, 2007
[22] Hiel, M., H. Weigand and W.J. van den Heuvel, "An Adaptive Service-Oriented Architecture", In: Mertens, K, R. Ruggaber, K. Popplewell, X. Xu (eds), Enterprise Interoperability III, pp.197-208, Springer, 2008.
[23] Papazoglou, M.P., Extending the Service-Oriented Architecture. In: *Business Integration Journal*, February 2005, pp. 18-21.
[24] W3C, "Web Services Semantics", Version 1, W3C member submission, 2005.

Supporting Corporate Governance with Enterprise Architecture and Business Rule Management: A Synthesis of Stability and Agility

M.W. (Matthijs) van Roosmalen, S.J.B.A. (Stijn) Hoppenbrouwers

Radboud University Nijmegen
Matthijs@van-roosmalen.com, S.Hoppenbrouwers@cs.ru.nl

Abstract. Business rule management (BRM) and enterprise architecture (EA) both offer support for corporate governance. They do this in different ways, with EA emphasizing a stable framework while BRM offers more agility to the enterprise through control of changing business rules. This paper explores the combination of BRM and EA in deployment to support governance, and argues for a synthesis between the two. Such a synthesis offers an organization the benefits of both stability and overview demanded by regulatory bodies, as well as agility in the face of rapidly changing compliance demands.

Keywords: corporate governance, compliance, business rules, business rule management, enterprise architecture

1 Introduction

1.1 Context

It is often stated that today's society is characterized by a high degree of turbulence and uncertainty, in which changes occur frequently and in rapid succession [1]. Global and interconnected forces such as globalization, shifting demographics, demanding consumer markets, environmental concerns and political activism are driving these changes [2]. Governments of the OECD countries have stepped in with regulations to contain some of the uncertainty and prevent corporate, political and environmental scandals. These regulations increasingly demand that organizations can prove having a clear insight into their operations and ensure compliance with applicable laws [3]. Well-known examples are the Sarbanes-Oxley Act in the U.S. and the Basel II framework.

This has led to challenges for organizations balancing their internal concerns from strategy formulation to execution and IT support with external demands on compliance from supervisory, regulatory and enforcement authorities. As both market conditions and legislation are subject to more and rapidly changing regulations, the cost of compliance rises [4]. This draws valuable time and resources away from the core business processes and pursuing new opportunities for competitive advantage. There appears to be a conflict in the demand for a stable governance framework that

supports transparency and accountability, and the ability to make quick changes to this framework, possibly harming its integrity. This organizational conflict between stability and change has also been referred to as the paradox of flexibility [43]. The ability of an organization to change quickly in response to external influences will be referred to here as agility [5].

1.2 EA and BRM in Corporate Governance

It is argued in this paper that the approaches of enterprise architecture (EA) and business rule management (BRM) offer complimentary positions concerning corporate governance[1] in light of the conflicting demands of stability and agility. When deployed together in an organization, these approaches may facilitate a synthesis where stability and agility do not conflict, but rather co-exist and complement each other in attaining successful governance. This contention is supported by the goal-oriented analysis of EA and BRM [6], which identified governance and flexibility as major areas of synergy.

In related work on business rules in the context of EA, particular attention has thus far gone out to the role of rules in architecting the enterprise [7] and documenting and modeling them [8][9][10]. This includes the positioning of business rules in enterprise architecture design and development methods and frameworks [11], such as the Zachman framework [12]. With so much emphasis on the architecting and design aspects, the deployment aspect has so far been largely neglected. Deployment in this context refers to the integration and application in the organization – in other words, actually using EA and BRM in order to realize their implied benefits.

This paper specifically concerns the deployment of BRM in conjunction with an enterprise architecture, rather than the development and design of the architecture and the business system. By focusing on deployment, it aims to address the significant knowledge gap that currently exists in this field. In particular, the consequences and benefits of deploying BRM and EA for the practice of corporate governance are identified. Besides contributing to the academic body of knowledge on these young disciplines, this is relevant for organizations dealing with complex governance issues, as well as those offering services or products related to EA or BRM.

First the aspects of EA and BRM that relate to their contribution to corporate governance will be discussed separately, during the course of the next two sections. In the fourth section, the synthesis between them is introduced and explicated. Finally some general conclusions and suggestions for further research will be given.

---

[1] The definition of corporate governance adopted in this paper is the inclusive definition given by Turnbull: "*Corporate governance describes all the influences affecting the institutional processes, including those for appointing the controllers and/or regulators, involved in organizing the production and sale of goods and services.*" [13].

2 Enterprise Architecture as a Stable Governance Framework

2.1 Defining Enterprise Architecture

The field of architecture is filled with different interpretations and applications of the term, which has resulted in a wide variety of definitions in the literature today. There are also a large number of frameworks, tools, descriptive languages, models and supporting methods in existence that can be applied widely to architectures at different levels of aggregation [14][15][16].

This paper considers EA to be the architecture that prescribes and describes an organization at its highest level, and at its most holistic. It is about the entire organization and all of its elements; not specific sub systems such as IT or particular business units. This is appropriate in the context of corporate governance, because here too, the organization has to be considered as an inclusive whole. For this purpose the following definition of EA is adopted from [3]:

Enterprise Architecture. *A coherent whole of principles, methods, and models that are used in the design and realisation of an enterprise's organisational structure, business processes, information systems, and infrastructure.*

Furthermore, EA itself is viewed on a meta-level of abstraction, which means that the properties of EA discussed in this paper are as independent as possible from specific approaches and implementations. Both guiding architectural principles that are used in architecting the enterprise, as well as more detailed models and visualizations of the architecture, are considered. This is of course done from the deployment perspective.

2.2 Enterprise Architecture for Governance

As systems become more and more complex, many organizations lack the required cohesion between different systems for them to be effectively maintained and controlled. This can be caused by historical mishaps such as integrating business processes by connecting originally separate systems and choosing the fastest and easiest solution to a particular need in an isolated area [17]. The result is a heterogeneous mix of systems spread throughout the organization, without a common structure, which is nearly impossible to oversee and maintain due to its complexity and size. This problem was the original driver for EA as conceived by Zachman [18], and is still recognized as an important issue today, but now includes the integration and alignment of business and IT [3]. As IT is becoming more embedded and integrated into organizations, the governance of IT from the enterprise perspective becomes increasingly important [19].

These developments have also affected the public sector and the field of e-government, where it is argued by Bellman and Rausch that it is crucial to adopt a holistic view encompassing both IT and business [20]. The thorough insight into the structure and processes of the organization along with its IT that is provided by

having EA in place makes it easier to ensure regulatory compliance and report on the internal situation to the required authorities. This allows crucial management decisions to be made more rapidly and securely. EA thus guides the translation of corporate goals into concrete actions that are in line with both regulatory demands and internal policies [21].

It has been said that EA functions as a map for the boardroom, which has the purpose of positioning decisions and overseeing their consequences in the broader context of the enterprise [17]. In other words, it serves the governance of the enterprise. There are four main ways in which EA contributes directly to corporate governance:

First, the EA facilitates comprehensive decision making by providing a holistic overview of the enterprise, which yields the insights necessary for understanding the ramifications of these decisions [22].

Second, the framework provided by EA is a solid basis for planning and setting goals and targets for various organizational units, as well as keeping track of who can be held accountable for them.

Third, EA enables the management and introduction of common standards and practices that are used and agreed upon. This may include standards regarding ways of working, policies, guidelines, IT and communication standards, and even best practices [23].

Fourth, EA supports the identification of risks throughout the enterprise, which is a boon to risk management. This overview created by an EA can be used to help identify and keep track of the responsibilities and owners with respect to various processes and risk-sensitive systems and areas.

2.3 Stability and Episodic Change

An important area of application for enterprise architecture that borders the domain of corporate governance is the directing of organizational change. This can be seen from two perspectives; the stable situation which is only changed occasionally and in revolutionary bursts, and more evolutionary changes that take place within a defined context and framework.

EA often deals with the migration from a state before the architecture (IST) to a more desirable new state that is prescribed by the architecture (SOLL). Considering the definition of EA given earlier: it guides the design of the business system. Once the migration to the desired state is complete, the architecture is preserved for a longer period of time, typically at least a few years. This is good because it allows the stable governance structures and procedures outlined in the previous section to be realized and put into practice. Having some measure of stability is a necessity for many of EA's contributions to corporate governance, such as a shared reference framework, agreed upon standards and insight into responsibilities and risks.

However, all enterprises invariably move through a life cycle from their initial concept in the mind of an entrepreneur through a series of stages or phases, just as their products and service offerings do [24]. The enterprise architecture by definition needs to change as the enterprise it governs moves from one stage in its life cycle to the next. Weick and Quinn refer to this kind of change as *episodic change* [25], but

the term *revolutionary change* is also used e.g. by [26]. Weick and Quinn state that these episodes of change undergo a trajectory consisting of three phases: unfreeze – transition – refreeze.

A major part of managing such revolutionary organizational change is often dealing with cultural and psychological factors regarding different stakeholders, in order to overcome resistance to change. This process is referred to as unfreezing. EA aids this process by reducing the resistance to change by offering a framework in which all the enterprise's objectives are positioned and the rationale for pursuing any of them at any particular time can be seen by everyone [27]. In the transition phase, the new or evolved EA guides the design and realization of the new enterprise from the IST to the SOLL state. The EA is then deployed in the refreeze phase, where it will remain stable until the next episode of change in the life cycle of the enterprise.

Even though EA is characterized by stability and only occasional episodes of great change, this does not mean it opposes or contradicts smaller, more evolutionary changes from happening. The stable framework of EA is also a valuable tool for facilitating changes in the organization that fall within the space prescribed by the architecture. In this manner the EA serves as a guiding framework through which the change efforts can be directed. What EA does generally not do however, is provide the means to make these changes as such.

3 Business Rule Management for Agile Governance

3.1 Defining Business Rule Management

Business rules are essentially all the rules that exist in an enterprise environment and are under the jurisdiction of the business. Organizations typically have thousands of such rules governing the business operations [28]. Various experts define business rules in a slightly different way, but all agree on their importance and that their main concern is that they should correlate directly to the business [29]. In this paper, the definition of the Business Rules Group will be adopted, because it is a widely accepted definition with a sufficiently thorough basis that is specific enough to be practically useful [30].

Business Rule. *A statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business.*

There are many different types of business rules according to various identification schemes and classifications; an overview can be found in [10]. They exist on two different levels: the business level and the information system level. In this view, business rules at the information system level are specified in a way understandable by machines, so that their processing can be automated. This does not mean that rules are essentially different on each level, but merely that they are represented in a different way. Some business rules are only present on the business level, but do not need to be implemented in a solution at the information systems level, and are for

example enforced through human efforts. Business rules always exist on the business level however, since the focus of BRM is on business and not technology.

BRM is in fact a mechanism for governing and controlling aspects of an organization, using business rules. A good working definition of BRM is given by von Halle [31].

Business Rule Management. *A formal way of managing and automating an organization's business rules so that the business behaves and evolves as its leaders intend.*

A thorough and comprehensive methodology for BRM can be found in [32]. It stresses the importance of modeling and deploying business rules in relation to enterprise models and business goals as well as information systems design. BRM is typically aided by sophisticated tools that manage large repositories of rules and provide support for the elicitation and authoring of the rules themselves, known as business rule management systems or suites (BRMS). Where formal execution and enforcement of the rules are automated at the level of the information systems, a business rule engine (BRE) is deployed. Such an engine makes use of reasoning algorithm technology to compute the applicable rules in a given situation and whether they are being complied with [33].

3.2 Governance through Business Rules

Business rules tend to focus on *what* needs to be done, leaving the *how* open to specific situations, implementation choices and the personal freedom of those following the rules [34]. Business rules therefore set boundaries for acceptable and desired behavior, allowing some room for creativity while maintaining a sense of fairness and consistency of output. This is a property of the way business rules are deployed, that distinguishes them from other rule-bound ways to regulate behavior such as strict formalization and fully specified instructions.

Perhaps the most important contribution of BRM to the organization is that the business rules can be changed relatively quickly. This allows the organization to respond more quickly to new risks and threats, increasing the capacity for agility. This added agility makes business rules suitable for guiding and controlling parts of an organization that are highly susceptible to change, both from within and from the environment. This is relevant in the primary processes of the organization, which implement the strategy set out by the organization in order to meet its business goals, but also in supporting and controlling processes which ensure compliance and are naturally rich in rules.

This has profound potential for corporate governance. Even though an organization may be too complex to capture everything in rules, the aim is to capture the right aspects that are crucial for efficient and responsive control. When the compliance demands from external regulatory influences change, this translates into changing business rules for the affected organization. BRM supports these rapid changes of the rules as well as their deployment and enforcement. If the currently applicable rules are immediately known at all times, this response time is further shortened [35].

BRM also provides insight into the rules that govern the enterprise. This goes for any given situation at which it needs to be clear which rules apply and should be satisfied. Deploying BRM forces organizations to make their policies and rules explicit. This enables them to always be available to the right persons; the ones who need to comply with them. Edwards states that business rules are "*core to establishing and maintaining a compliance competent organization*" [36]. Dissemination of knowledge of the applicable rules is therefore an important contribution made by BRM to compliance.

Also important is knowledge regarding the consequences of any violation of the rules and the likelihood of this happening. This touches upon the area of risk management. By having access to the rules, insight is gained into the risks, making it possible to assess them with greater accuracy. Business rule technology also offers possibilities for simulating different scenarios based on simulated changes in the rules. This helps to identify potential compliance risks in future situations, for example when new laws are about to go into effect.

3.3 Agility and Continuous Change

In contrast to EA, BRM is all about providing the means to make rapid changes to the way the business is run. Because the business rules are separated from the processes, activities and information systems of the organization, they can be more easily managed and changed [37]. This allows the organization to respond to changes in the environment. The detection of such changes is often considered to be in the domain of environmental scanning and business intelligence [38]. In the context of corporate governance, BRM clearly allows for more sense-and-respond agility towards the marketplace and regulatory bodies demanding compliance.

The changes that BRM facilitates are often short term, isolated in specific areas and not deeply rooted in the organization's culture and values. These characteristics on the dimensions of time, complexity and culture are typical of what Weick and Quinn refer to as *continuous change* [25]. This kind of organizational change is also known as incremental or *evolutionary change* [26]. Such changes are cyclic and without a clear end state, as opposed to episodic change which is linear (from IST to SOLL) and between stable states. This is where the agility offered by BRM is evident, the rules can always be changed to reflect the current demands and the set of rules is never constrained by a long term end state. Weick and Quinn state that continuous change consists of an enduring cycle of the three phases freeze – rebalance – unfreeze.

A change intervention made by BRM is a good example of what happens in the freeze phase. The rules and patterns governing the current state are made visible and tangible so that they can be changed. BRM yields insights into the rules that are relevant in the light of new circumstances, and which need to be altered. It also gives the organization the means to rebalance the situation. In this phase, the situation is reevaluated and the rules changed in such a way that the organization is compliant in the new state. Finally, there is the phase of unfreezing, in which the rules are once again interpreted and applied by individuals. It is crucial that this leaves these

individuals the appropriate degrees of freedom to improvise and learn, which is possible because the rules specify the *what* and not the *how*.

While BRM typically supports continuous, evolutionary change, this does not imply that it prevents revolutionary change from happening. BRM has been suggested as a powerful tool in business restructuring and re-engineering efforts [39], which are revolutionary in nature. However, the core of the agility in governance offered by BRM is due to the fact that it enables continuous change that benefits the compliance of the organization. What BRM may lack due to its malleable nature is a consistent and stable framework providing overview, in order to keep track of changes and support more complex revolutionary changes when they become necessary.

4 Synthesis between EA and BRM

4.1 Comparative Goal Analysis

The complimentary position of BRM and EA is made clear by the comparative goal analysis of their normative organizational goals [6]. This analysis has identified 35 unique goals of both BRM and EA, for a total of 70 goals. These goals have been structured and modeled in the form of hierarchical goal trees, which revealed areas of similarity as well as differences. The leaf goals of the different goal trees were then analyzed for their compatibility, which resulted in the network of goal relationships shown in fig. 1. Some clusters representing common or mutually compatible goals can clearly be seen. Main goals that emerged from the goal analysis have been included in the text of this paper as relevant. For details on the goal analysis, see [6].

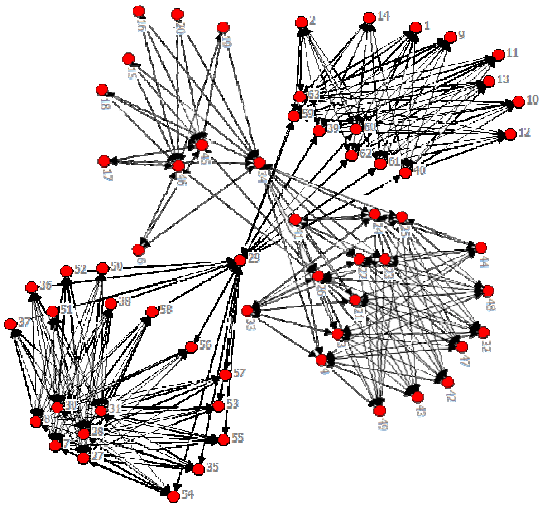

Fig. 1. The network of organizational goals of both EA and BRM on the leaf level of the goal tree hierarchy, as drawn in UCINET [40]. These goals have been clustered and analyzed in order to identify complementarities between the two approaches.

One of the areas where the goals of EA and BRM complement each other is that of corporate governance [6]. The different ways in which these two approaches support governance and compliance have been outlined in the previous two sections of this paper. These complimentary contributions may offer benefits regarding the flexibility, reliability and effectiveness of corporate governance. There are also some differences however that need to be reconciled in order to realize the potential benefits of the joint deployment of EA and BRM.

These differences concern the approach to change and stability. EA puts the most emphasis on preserving a stable state and using it to direct the business, only occasionally engaging in episodes of revolutionary change. BRM on the other hand is focused on enabling rapid changes to fine-tune the business and respond to environmental influences, in a way that is continuous and evolutionary. It is the assertion of this paper that both of these approaches, in the form of stability and agility, contribute to successful governance.

4.2 Synthesis of Stability and Agility

In order to benefit the most from the joint deployment of BRM and EA, a synthesis must be reached which incorporates both a stable governance framework and sufficient agility to cope with rapidly changing demands. In such a synthesis, BRM makes the EA more flexible, while EA provides the missing governance overview to the BRM. Here it is useful to make a distinction between higher order governing of business design and strategy execution, which is likely to be more constant, and the day-to-day operations of the business, which may have a higher degree of liquidity.

A possible weakness of EA is that because it focuses on a high level of abstraction, it becomes too hierarchically structured, prescriptive and one-size-fits-all. When only major episodes of change are facilitated, it becomes constraining in terms of innovation and struggles to adapt to a turbulent environment. The combination with BRM gives an organization the means to make continuous changes. These changes should take place within the overall boundaries of the EA and are concentrated in the business operations that need to adapt to changing demands regarding for example compliance.

The swift and easy changes in the rules increase the adaptability of individual processes and services, but they should be managed at a higher level, where the necessary overview of the enterprise as a whole exists. This is where EA provides the insight and overview necessary to guide the lower level agility in the right overall direction. This concept of operational agility built upon a solid base of business values and insight for higher level guidance is particularly suitable for surviving and competing in turbulent environments [41].

Particularly with compliance in mind it is crucial to not only have an overview of the risks and responsibility structure within the organization, but also to have certain elements of this structure firmly in place. The demands from regulatory organizations are such that they require an orderly framework for clear-cut procedures to deal with legislation and governmental standards. Such a framework should be somewhat stable in order to accommodate the meeting of all compliance requirements. This is typically done at the EA level, affecting all units of the organization. The Business Motivation

Model [42] provides a model for positioning business rules in an organizational context, but lacks the prescriptive power needed to guide organizational change. When the governance framework is no longer viable, it needs to be reconstructed by means of an episodic change process guided by architecture.

What BRM advocates is an agile approach in which rules can be easily changed in order to meet changing requirements for compliance, often eliminating the need for revolutionary change. This becomes necessary when changes occur so quickly that the sluggish overall framework of governance is not able to keep up, and going through episodes of major upheaval for every change would be too costly. The redesign process is therefore sometimes better carried out in an evolutionary way [44]. While changes are frequent, they usually do not occur across the entire range of regulations at the same time, but tend to focus in specific areas. Therefore, both a stable overall framework and the ability to quickly change specific compliance measures are needed.

This is where BRM can enrich the EA framework for governance and compliance, by separating the rules and making them easily accessible and changeable within the boundaries laid out by the architecture. This reduces the complexity and waiting times involved in making changes required in response to specific external regulations, while also maintaining an enterprise wide overview and governance framework. If only BRM were to be used, this overview could be lost because critical parts of the business are expressed in a multitude of atomic rules.

These findings are in line with an emerging body of literature that argues that organizations combine evolutionary and radical change harmoniously [44]. Tushman and O'Reilly refer to organizations that control both revolutionary and evolutionary change as being ambidextrous [26]. This is the key to the synthesis between stability and agility in corporate governance, which can be achieved by deploying both EA and BRM.

5 Conclusion

This paper discussed the contributions of EA and BRM in support of corporate governance and the relationship between the two approaches. It was found that both have complimentary ways in which they support the common goal of governance, but differ regarding their approach to change. EA takes a higher level view of governance and supports a stable framework, while BRM facilitates agile operations and compliance. A synthesis between the two approaches in combined deployment allows for both stability and agility in governance. This synthesis supports corporate governance in dealing with the demands regarding stability from regulatory supervision and agility from changing legislation and a turbulent environment.

This has profound consequences for research into EA and BRM in the broadest sense and for the purpose of governance in particular. Both fields have a lot to gain from more integration between the two, because they complement each other's weaknesses in working towards the same goal. Future research should focus on the joint development as well as deployment.

The consequence for the practical deployment of BRM and EA in organizations that wish to improve their governance is that neither approach is by itself sufficient to deal with the demands regarding stability and agility and that they should be combined. Organizations will have to consider their environment and find the right mix of a stable EA and continuous changes in the governing business rules.

References

1. Chakravarthy, B.: A New Strategy Framework for Coping with Turbulence. Sloan Management Review. 38, 2 (1997) 69-82
2. Laudicina, P.A.: World out of Balance: Navigating Global Risks to Seize Competitive Advantage. McGraw-Hill, New York (2005)
3. Lankhorst, M.M. et al.: Enterprise Architecture at Work: Modelling, Communication, and Analysis. Springer-Verlag, Berlin Heidelberg (2005)
4. Hopkins, T.D.: Regulatory Costs in Profile. CSAB Policy Study Number 132 (1996)
5. Pal, N., Pantaleo, D.C. (eds.): The Agile Enterprise. Springer, New York (2005)
6. van Roosmalen, M.W.: Enterprise Architecture and the Business Rule Approach: A Goal-Oriented Analysis and Synthesis. Master's Thesis, Radboud University Nijmegen, The Netherlands (2008)
7. Dietz, J.G.L.: Architectural Principles & Business Rules. Presentation delivered at the Business Rule Platform Nederland, December 6th (2007)
8. Perkins, A.: Business Rules = Meta-Data. Proceedings of the 34th International Conference on Technology of Object-Oriented Languages. Los Alamitos, CA (2000) 285-294
9. Liles, D.H., Presley, A.R.: Enterprise Modeling Within an Enterprise Engineering Framework. In: Charnes, J.M., Morrice, D.J, Brunner, D.T., Swain, J.J. (eds.): Proceedings of the 1996 Winter Simulation Conference (1996) 993-999
10. Taveter, K., Wagner, G.: Agent Oriented Enterprise Modeling Based on Business Rules. In: Proc. of 20th Int. Conf. on Conceptual Modeling (ER2001), Yokohama, Japan. Springer-Verlag. LNCS 2224 (2001) 527-540
11. Iyer, B.,Gottlieb, R.: The Four-Domain Architecture: An Approach to Support Enterprise Architecture Design. IBM Systems Journal. 43, 3 (2004) 587-597
12. Zachman, J.A.: Enterprise Architecture: Managing Complexity and Change. In: Von Halle, B., Goldberg, L. (eds.): The Business Rule Revolution. Happy About (2006)
13. Turnbull, S.: Corporate Governance: Its Scope, Concerns and Theories. Corporate Governance. 5, 4. (1997) 180-205
14. Schekkerman, J.: How to Survive in the Jungle of Enterprise Architecture Frameworks: Creating or Choosing an Enterprise Architecture Framework. Trafford, Canada (2003)
15. Jonkers, H. et al.: Towards a Language for Coherent Enterprise Architecture Descriptions. Proceedings of the Seventh International Enterprise Distributed Object Computing Conference (EDOC'03) (2003) 28-38
16. Arbab, F., De Boer, F.S., Bonsangue, M., Lankhorst, M.M. Proper, H.A., van der Torre, L.: Integrating Architectural Models. Enterprise Modelling and Information Systems Architectures. 2, 1. (2007) 40-57
17. Rijsenbrij, D., Schekkerman, J., Hendrickx, H.: Architectuur, Besturingsinstrument voor Adaptieve Organisaties: De Rol van Architectuur in het Besluitvormingsproces en de Vormgeving van de Informatievoorziening. Lemma Utrecht (2004)
18. Zachman, J.A.: A Framework for Information Systems Architecture. IBM Systems Journal. 26, 3 (1987).
19. Korac-Kakabadse, N., Kakabadse, A.: IS/IT Governance: Need for an Integrated Model. Corporate Governance. 1, 4 (2001) 9-11
20. Bellman, B., Rausch, F.: Enterprise Architecture for e-Government. In: Traunmüller, R. (ed.): EGOV 2004. Springer-Verlag Berlin Heidelberg. LNCS 3183 (2004) 48-56
21. Ross, J.W., Weill, P., & Robertson, D.C.: Enterprise Architecture as Strategy. Harvard Business School Press, Boston MA (2006)

22. Johnson, P., Ekstedt, M., Silva, E., Plazaola, L.: Using Enterprise Architecture for CIO Decision-Making: On the Importance of Theory. Proceedings of the 2nd Annual Conference on Systems Engineering Research. Los Angeles, CA (2004)
23. Rood, M.A.: Enterprise Architecture: Definition, Content, and Utility. Proceedings of the 3rd Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. (1994) 106-111
24. Williams, T.J.: The Purdue Enterprise Reference Architecture. Purdue Laboratory for Applied Industrial Control, Purdue University (1992)
25. Weick, K.E., Quinn, R.E.: Organizational Change and Development. Annual Review of Psychology. 50 (1999) 361-386
26. Tushman, M.L., O'Reilly, C.A. III. The Ambidextrous Organization: Managing Evolutionary and Revolutionary Change. California Management Review. 38, 1 (1996) 8-30
27. Veasey, P.W.: Use of enterprise architectures in managing strategic change. Business Process Management Journal. 7, 5 (2001) 420-436
28. Gottesdiener, E.: Business RULES Show Power, Promise. Application Development Trends. 4, 3 (1997)
29. Steinke, G., Nickolette, C.: Business Rules as the Basis of an Organization's Information Systems. Industrial Management & Data Systems. 103, 1 (2003) 52-63
30. The Business Rule Group. Defining Business Rules ~ What Are They Really? (2001)
31. von Halle, B.: Business Rules Applied. Wiley New York (2002)
32. Bajec, M., Krisper, M.: A Methodology and Tool Support for Managing Business Rules in Organisations. Information Systems. 30 (2005) 423-443
33. Charfi, A., Mezini, M.: Hybrid Web Service Composition: Business Processes Meet Business Rules. Proceedings of the 2nd international conference on Service oriented computing, New York (2004) 30-38
34. Date C.: What not How: The Business Rules Approach to Application Development. Addison-Wesley Publishing Company (2000)
35. Cetin, S., Altintas, N.I., Solmaz, R.: Business Rules Segregation for Dynamic Process Management with an Aspect-Oriented Framework. In: Eder, J., Dustdar, S. (Eds.): BPM 2006 Workshops. Springer-Verlag Berlin Heidelberg (2006) 193-204
36. Edwards, J.: Compliance Competent Life Assurance Companies: A Partnership Approach. Journal of Financial Regulation and Compliance. 11, 1 (2003) 10-21
37. Ross, R.G.: Principles of the Business Rules Approach. Pearson, Boston: MA (2003)
38. Choo, C.W.: The Art of Scanning the Environment. Bulletin of the American Society for Information Science. 25, 3. (1999)
39. Rosca, D., Wild, C.: Towards a Flexible Deployment of Business Rules. Expert Systems with Applications. 23 (2002) 385-394
40. Borgatti, S.P., Everett, M.G., Freeman, L.C.: Ucinet for Windows: Software for Social Network Analysis. Analytic Technologies, Harvard: MA (2002)
41. Ahmed, P.K., Hardaker, G., Carpenter, M.: Integrated Flexibility—Key to Competition in a Turbulent Environment. Long Range Planning. 29, 4. (1996) 562-571
42. The Business Rules Group: The Business Motivation Model: business governance in a volatile world, release 1.2 (2005)
43. Volberda, H. W.: Building The Flexible Firm: How to Remain Competitive. Oxford University Press, New York (1998)
44. Jarvenpaa, S.L., Stoddard, D.B.: Business Process Redesign: Radical and Evolutionary Change. Journal of Business Research. 41, 1 (1998) 15-27

Modelling Parliamentary Workflows a Case Study in Belgian Parliaments

Christophe Ponsard[1], Gaetan Deberdt[2], and Joël Tournemenne[3]

[1] CETIC Research Center, Charleroi (Belgium) - cp@cetic.be
[2] Parlement de la Communauté Française (Belgium) - gaetan.deberdt@pcf.be
[3] Parlement Francophone Bruxellois (Belgium) - jtournemenne@pfb.irisnet.be

Abstract. Parliament work is regulated by a number of democratic rules about the way laws are proposed, discussed and finally voted. Despite a number variations, most parliaments share the same kind of workflow supported by one or two assemblies. Such workflows are most of the time described by a regulation stated in natural language and generally approved by the assemblies themselves. This document is subject to some interpretation, especially by the administration responsible of the day to day management. Currently this management is also on-going strong electronification with even a direct exposure of the parliamentary work on the internet for better transparency and control by the citizen. In this paper we report about our work of modelling the parliamentary workflows, starting from the official documents and in-place systems. The aim of this work is multiple: first, discover potential ambiguities and inconsistencies, then compare how similar are a number of parliaments and finally see how those models can be translated in the computer systems, especially in the perspective of the open-sourcing and mutualisation of such systems among different parliaments. Our practical experience of applying various modelling techniques is reported and discussed using two of the seven (!) parliaments running in Belgium. This comparison work relies both on a set of modelling requirements for such systems and on the SEQUAL reference framework for assessing the quality of models.

Key words: e-government, parliament, modelling, workflow, mutualisation

1 Introduction

All democratic countries of the world run some kind of parliamentary system whose main functions are to make law and control the work of the executive power, following the principle of separation of powers. Parliaments may consist of chambers or houses, and are usually either bicameral or unicameral. In bicameral systems, the lower house is almost always the originator of legislation, while the upper house is usually the body that offers the "second look" and decides whether to veto or approve the bills [23].

Law making also follows a general common process, starting from a bill either proposed by the executive or legislative body, then discussed by the assemblies.

After preliminary readings, it is generally sent to specialised committees which will work on it. This will result in a number of amendments and finally a vote. In case of adoption, the law is then promulgated by being officially signed by the authority (e.g the President or the King) and finally published. It is then generally followed by executive laws to enforce it. The following figure show this process for the Australian (bicameral) parliament.

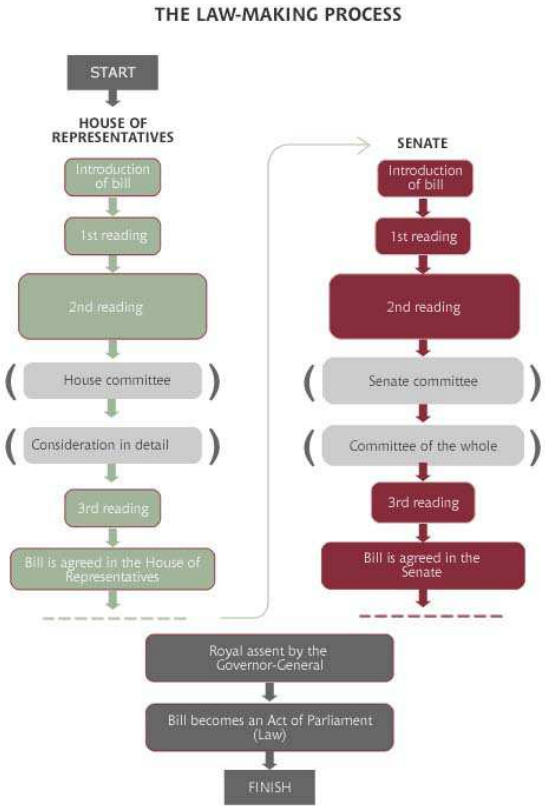

Fig. 1. Typically Parliamentary Workflow [15]

With the raise of ICT, e-democracy is on its way and is present at various levels: e-voting, e-forms, e-referendum... and among them e-legislation which is the part we are interested in here. The electronification process has already started in the 90's at the data and document level (scanning, OCR, meta-data, automated generation of documents, diffusion on web-site). More recently it is reaching the legislative processes themselves. After initial phases of incertainty and euphoria, this evolution is now reaching some maturity and being "institutionalised" [4][19]. The main goals identified in this process are to improve:

– *the procedural quality:* better modelling (less complex, less operational), supporting evolution and re-engineering;

- *the output quality:* drafting systems for improving the formal quality of legislation and regulatory impact assessment for improving the material quality of legislation;
- *the participatory quality:* introducing new communication tools into the representative system or even more visionary concepts for new democracy models.

Most parliaments have now a strong ICT department in charge of this work. As parliaments have the same business, this also means that they need the same kind of solution. Rather than reinventing the wheel, some assemblies have started to collaborate and mutualise their efforts, this is especially true in Belgium which has a complex organisation with many assemblies at region, community and federal levels. This effort also requires to be able to know precisely the commonalities and differences between those assemblies and thus to model them precisely.

This paper reports about a practical case-study done in two regional assemblies of Belgium in the context of mutualising their development with the longer term goal to open-source the resulting more generic software [7]. Those assemblies are the Parliament of the French Community (PCF in short) and the French Parliament of Brussels (PFB in short), which are respectively a medium-size and a smaller size parliament. The first step of this study was to precisely model those two assemblies, starting from the existing situation as documented in the regulation issued by the assemblies themselves and as observed on the field [16].

This paper is structured as follows. In section 2, we will discuss about the requirements on the adequate language to capture parliamentary workflow. Then, in section 3, we will see how a number of candidate languages fit those requirements by showing selected parts of our case studies. Section 4 will compare those models based both on the previous requirements and on a reference framework for assessing model quality. Based on this, a number of important lessons learned from those models will be discussed. Finally, section 5 will draw some conclusions and perspectives.

2 Requirements on the Modelling Language

The main requirements discovered during the study were the following:

- *[BEHAV] Ability to capture behaviors.* The language must be able to capture the dynamic nature of the parliamentary workflows.
- *[RESPO] Ability to capture responsibilities.* The language should be able to describe the various agents playing some role in the system and their responsibilities. More precisely what they control and under which circumstances.
- *[GOAL] Ability to capture the goals.* The language should be able to capture underlying goals of some operational construct. Goals can be either functional or non-functional (such as security, reliability, etc.)

- *[PRECISE] Precise language.* The language should be precise and unambiguous.
- *[UNDER] Easy to understand.* The language should be accessible to non specialist for validation purposes. Languages should preferably have a graphical semantics associated with it.
- *[TOOLS] Tool support.* The language should be supported by tools at modelling level and at run-time level, either directly or through some model transformation.

3 Study of Selected Languages

This section reports about various modelling techniques used to model parliamentary work. It does not claim to present all relevant techniques in an exhaustive way. A useful reference for this is [24].

3.1 Use Cases and Sequence Diagrams

A use case is a description of a system's behaviour as it responds to a request that originates from outside of that system. Use cases, stated simply, allow description of sequences of events that, taken together, lead to a system doing something useful.[2] Each use case describes how the actor will interact with the system to achieve a specific goal. One or more scenarios may be generated from each use case, corresponding to the detail of each possible way of achieving that goal (or possible exception/failure).

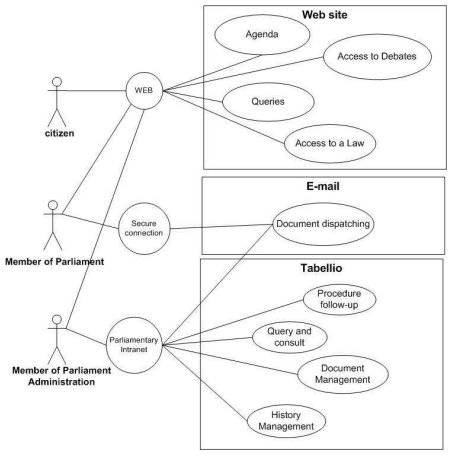

Fig. 2. UC Context Model of a Parliament Management System.

Notations for Use Case include UML Use Case (graphical) [11] and template-based descriptions (textual) [5]. They are usefully complemented by sequence

diagrams for graphically describing the generated scenarios with a very comprehensive view of the system with the time on the vertical dimension and the interaction between agents structured horizontally. Note while UML 1.X sequence diagrams were limited to rough traces, UML 2.X supports many structuring operators like conditionals, options, even loops, with the danger to capture too much complexity in a single scenario.

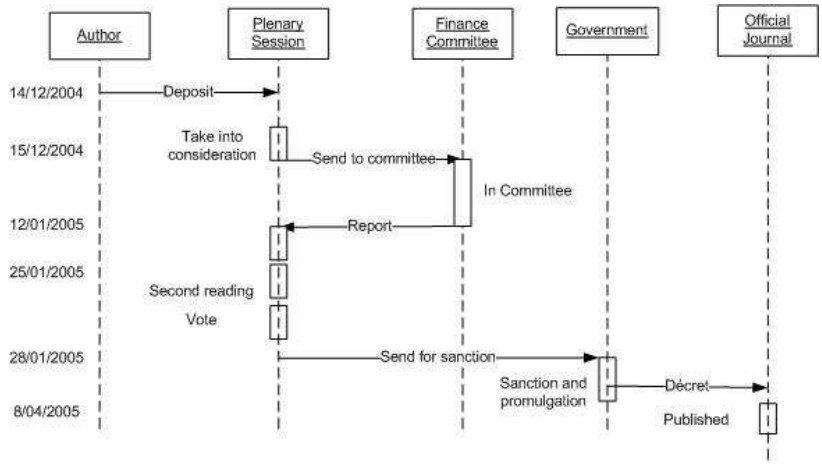

Fig. 3. Sequence Diagram for some procedure.

UML Use Cases diagrams have a good capacity to capture the context of the system and the general responsibilities but lacks the capacity to reflect the dynamic behavior. Textual templates can describe some part of the behavior but not very precisely. Sequence diagrams used together enable a more precise capture of behaviours but generally partial and at instance level. Goals can be captured using methods like [5] however generally mainly at functional level.

3.2 Goal Models

From [22], a goal is an objective the system under consideration should achieve. Goal formulations thus refer to intended properties to be ensured; they are optative statements as opposed to indicative ones, and bounded by the subject matter. Goals may be formulated at different levels of abstraction, ranging from high-level, strategic concerns (such as "Efficient Management of Parliamentary Work") to low-level, technical concerns (such as "Publishing of Voted Laws on Parliamentary Website"). Goals also cover different types of concerns: functional concerns associated with the services to be provided, and nonfunctional concerns associated with quality of service - such as safety, security, accuracy, performance, and so forth. Within the scope of this paper, we will use the KAOS goal-oriented language [8].

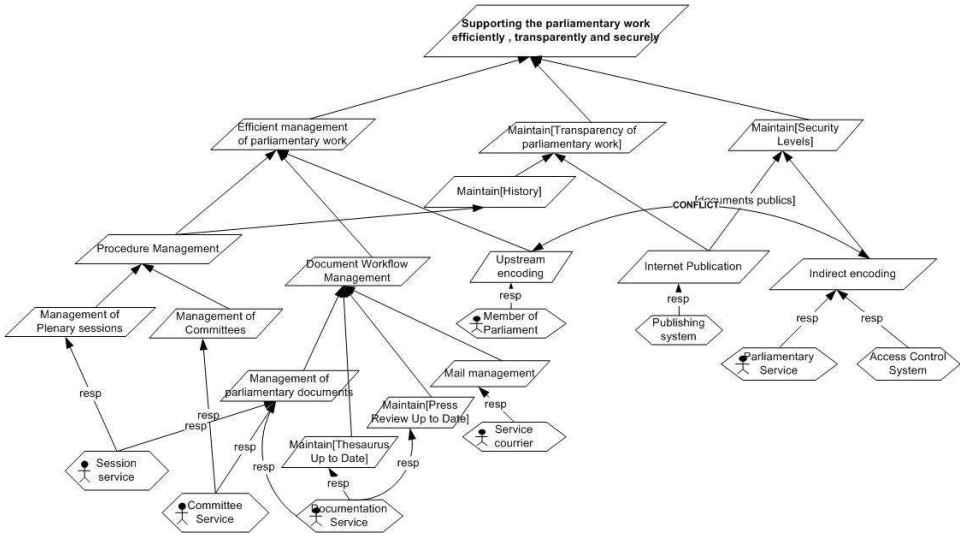

Fig. 4. Goal Model of the Parliament Administration.

Goal models enable to capture, structure and reason about system properties and agent responsibilities. Languages like KAOS have precise semantics. The goal level is defined using temporal logics [14], semantics refinements and operations are also precisely defined [9][13]. Not however that the operational level is not very practical to use especially to describe workflows as the language is not designed for this level.

3.3 Final State Machines and State Diagrams

A finite state machine (FSM) is a model of behavior composed of a finite number of states, transitions between those states, and actions. FSM have been extended by Harel to statecharts to allow the modeling of superstates, concurrent states, and activities as part of a state. This notation is now standardised in UML State Diagrams [11].

State Diagrams are very popular. They are very easy to understand and thus to use to validate a behavior even with non-experts. The hierarchical structure allows also the system to be nicely described at progressive levels of details. There are precise semantics although several alternative semantics have been defined, leaving possible ambiguities but generally for specific cases. Goals can be associated with a FSM for example as invariant or obligation an FSM should enforce. This can be verified using model-checking tools.

FSM are also supported by tools for simulating the system or generating the behavioral part of the code (e.g. Rhapsody [21]). It is also easy to design such a generator. In our case study, the company responsible of the system development has such a framework, called XOooF which is now open-source [20]. The framework supports the partial generation of the application code

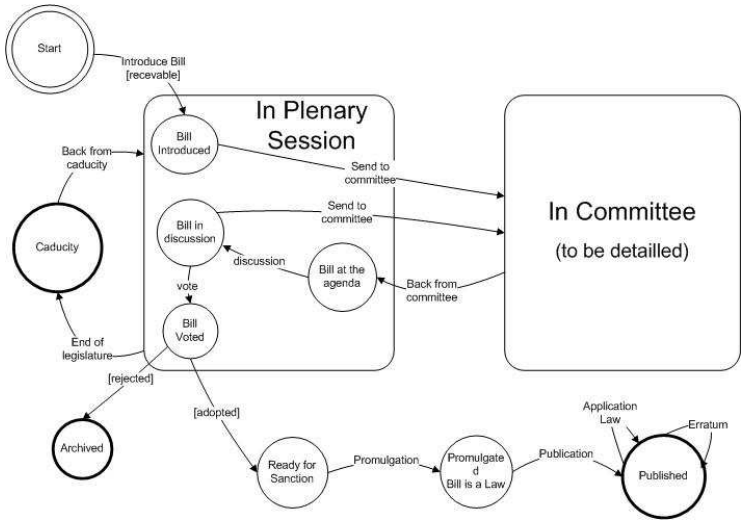

Fig. 5. Final State Machine for the Journey of a Bill.

from XML-based description of state machines. Some aspects not covered are persistency, advanced transactions and graphical user interfaces. Several target languages such as VB/COM, C#, Java and Python are supported.

3.4 Business Process Oriented Languages

Many notations have developed for modelling business processes, with different coverages (activities, products, decisions, context), specification levels (organisation, orchestration, web-services) and underlying semantics. To leverage this, BPMN (Business Process Modelling Notation) is a current standardisation effort aiming at unifying the expression of basic business process concepts (e.g., public and private processes, choreographies) as well as advanced modelling concepts (e.g., exception handling, transaction compensation) [1]. The connection of BPMN with more operational standard such as BPEL (Business Process Execution Language) is however not entirely solved as discussed in [17] but seems to be evolving favorably.

UML - more software-oriented - also support this kind of modelling through the activity diagram which can represents business and operational step-by-step workflows of components in a system. Activity diagrams can be unstructured or organised using swimlanes (somehow similar to sequence diagram lifelines) which enable a better capture of the action responsibilities. Figure 6 shows a typical process model of the parliament work using those notations.

At semantic level, BPMN remains semi-formal although quite complete. Some attempts have been made to more deeply formalise parts of it [3]. Back to our example described with UML, the semantics were changed between UML

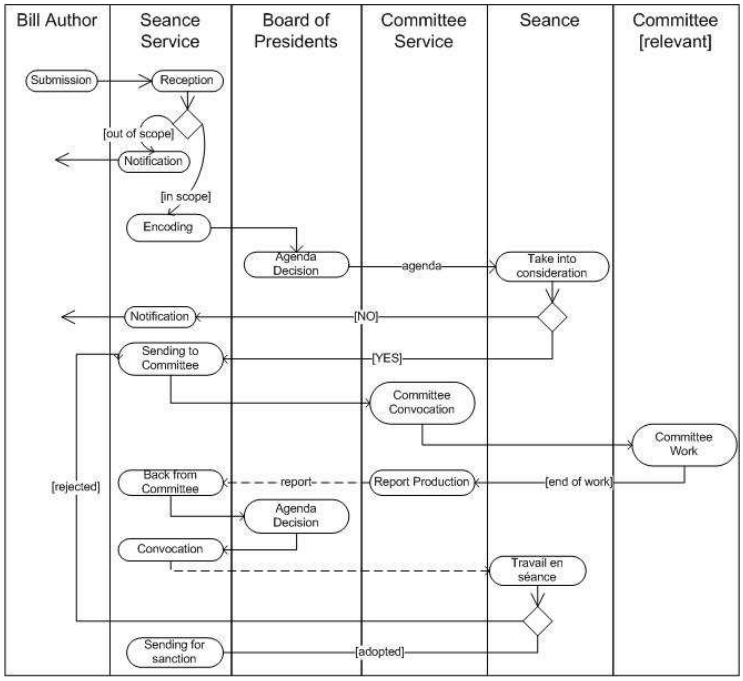

Fig. 6. Activity Diagram for the Journey of a Bill.

1.x w (variation of the UML State Diagram) and UML 2.x (semantics based on Petri nets) [18]. This is a good evolution as Petri nets have better mechanisms for controlling concurrency and synchronisation which is important in workflow management. Petri nets are frequently used as formal underlying model and are also supported by tools (e.g. Flexo was considered for the Belgian case study [10]).

4 Lessons Learned

In this section, we will first compare the qualities of the previous models w.r.t. the requirements described in section 2. This discussion will also rely on the SEQUAL reference framework for assessing the quality of models [12]. The rest of the section will put those conclusions in a wider perspective by going back to the e-government goals defined by Schefbeck [19].

4.1 Comparison Table of Modelling Languages

Table 1 summarises our comparative work based on our requirements described in section 2.

Model	Use Cases	Goal Trees	State Diagrams	Business Process Models
Behaviour	through sequence diagrams	partially	very good, hierarchical, scalable	very good
Responsibility	at context level	very good	poor	through swimlanes
Precision	semi-formal	formal (KAOS)	formal (FSM)	formal (petrinets)
Understand.	good	good	very good	very good
Tools	UML tools	Objectiver	UML tools, Rhapsody	BPEL tools, some UML tools...

Table 1. Modelling Languages Comparison Table (domain requirements)

To consolidate this comparison, we used the SEQUAL reference framework which defines a number of model qualities: empirical, syntactical, semantical, pragmatic, societal, knowledge and language [12]. Those quality factors are compared in table 2. Some of those qualities are already addressed in our requirements: empirical is understandability, semantical is precision. The organisational quality is defined as how well the goals of modeling are reached by the model. This is exactly the purpose of table 1, so this factor is a synthesis of that table.

Model	Use Cases	Goal Trees	State Diagrams	Business Process Models
Empirical	Poor-to-medium (depending on template used)	medium-to-good (depending on refinement checking strategy)	medium (difficult to structure)	good (control flow)
Semantical	semi-formal	formal (KAOS)	formal (FSM)	formal (petrinets)
Pragmatic	good	good	very good	very good
Knowledge	good (capture of scenario/functions)	very good (capture of system goals)	poor (states/transitio not directly linked to domain)	good (business process level)
Organisational	medium	medium	medium	good
Language	generic	generic	generic	more specific

Table 2. Modelling Languages Comparison Table (SEQUAL)

The main lessons learned from those tables is that a single language does not fit all our requirements. Activity diagrams seem the most adapted for our purpose given the current evolution of methods and tools while in the past, state machines were more the reference framework.

Other notations are useful to use in a complementary matter. Especially in the reengineering, and comparative study it is important to make sure the goals

are fully aligned because variation in goals will inevitably result in variation at the workflow level and it is important to understand if some variation is a design decision or more fundamentally bound to a goal.

4.2 Procedural Quality

The use of modelling techniques helped greatly in the process of understanding the way the assemblies are working, their commonalities and differences.

During the elicitation phase in the first assembly (PCF), the various models were built from a number of sources of domain knowledge: the official regulation of each parliament, interviews with the staff and the documentation of the existing system. Building those models allowed us to have guidelines for completeness (e.g. asking about missing transitions) and for conflict identification (e.g. different actions reported by different sources). It allowed us to discover a number of undocumented choices left open by the regulation and to understand the rationale behind those choices. This resulted in an improvement of the documentation of the procedures, which are not only meant for developing a new system but also helpful as training material for new collaborators.

The work in the second assembly (PFB) did not start from scratch but was carried out based on the models from the first assembly (PCF), assuming their would be only few differences. This assumption was confirmed with the following main differences:

- *Syntactic variations* in the vocabulary used (e.g. the term for a law, for the board of presidents...)
- *Small behavioral differences*, typically variations in some transitions. Those are easily implemented at specification level and propagated to the implementation by regenerating the impacted code.
- *A more fundamental difference is the distribution of roles*: as PFB is smaller, the same people would typically handle a several tasks. As the model was built using roles, this has however no impact on our models.

4.3 Output Quality

Prior to our study, a strong model-based approach was already in place in PCF (based on finite state machines) and partially at PFB (based on a document management workflow).

The impact on the output quality was especially visible at PCF with a chain of model-based tools supporting the whole parliamentary process, from the gathering of minutes to the diffusion of the reports on the website.

The traceability of the parliamentary process is also excellent based on the accumulation of state traces in the system.

4.4 Maintainability and Reusability

The long term goal initiated by the case study is to eventually be able to share common code between assemblies and even to open-source such code. The current closed source model has a number of limits, especially when a number of

assemblies share common needs and have to develop their own solutions separately and at high cost. This process has already started under the Tabellio project [7]. A number of generic enough modules have been open-sourced together with the XOoof FSM-based framework.

For the process to be successful, the code quality should however be improved prior to its open-sourcing and this is currently on-going. A major evolution is the transition to a workflow management systems which is not based on code generation as before but on a workflow engine, based on the Plone framework and in coordination with other e-Government initiatives such as PloneGov [6]. Here again, the underlying model proves fundamental has it will drive the configuration of the new system and the definition of the data migration procedure between repositories.

5 Conclusions and Perspectives

In this paper, we explored various way to model parliamentary workflows using different languages. The comparison was driven by a real-world case study and performed using both specific requirements and the SEQUAL reference framework. As expected, a single language cannot fit all requirements and qualities. However business process models seem the most adapted for the needs of modelling parliamentary workflows. Other notations such as goal models allow the analyst to have a deeper insight of the system and to better understand variations between different assemblies and better manage the evolution of a given system.

Among the other lessons learned, the use of adequate models greatly helped in the understanding of the way each assembly was working and how similar they were. Models are also fundamental to deploy tool support. Firstly, in a model-driven architecture perspective, models greatly ease the development of solutions by removing the need to write and test substantial part of the system. Secondly, in a mutualisation perspective, those tools can even be shared together with some representative models and guidelines on how to adapt them. The reuse is then maximal, reducing maintenance costs and allowing easy tuning to the need of other assemblies, especially those of developing countries. This also opens a number of interesting perspectives to further improve the way democracy works: speeding the process, introducing more transparency, etc.

At the methodological level, there is room for many improvements. The comparison work done here is very coarse grained. In order to draw more conclusions about the models and the way to use them, more precise metrics have to be defined and measured together with the reference quality framework. This work will be considered in the current re-engineering phase of the workflow system.

Acknowledgements

This work was financially supported by the Walloon Region and European Union (ERDF and ESF). We also warmly thanks the staff of the respective parliaments.

References

1. *Object Management Group/Business Process Management Initiative*, http://www.bpmn.org/.
2. Kurt Bittnera and Ian Spence, *Use Case Modeling*, Addison Wesley Professional, 2002.
3. M. Brambilla, *LTL Formalization of BPML Semantics and Visual Notation for Linear Temporal Logic*, Tech. report, January 2005.
4. Daniel Brassard, *How can information technology transform the way parliament works ?*, Parliamentary Information and Research Service - Library of Parliamentarians of Canada, 2005.
5. Alistair Cockburn, *Writing Effective Use Cases*, Addison-Wesley, 2001.
6. PloneGov Consortium, *The PloneGov Project*, http://www.plonegov.org.
7. Tabellio Consortium, *Tabellio : an Open Source Collaboration for Assemblies*, http://www.tabellio.org.
8. A. Dardenne, A. van Lamsweerde, and Stephen Fickas, *Goal-Directed Requirements Acquisition*, Science of Computer Programming **20** (1993), no. 1-2, 3–50.
9. R. Darimont and A. van Lamsweerde, *Formal refinement patterns for goal-driven requirements elaboration*, 4th FSE ACM Symposium, San Francisco, 1996.
10. Denali, *FlexoBPM*, http://www.denali.be.
11. Martin Fowler, *UML Distilled - Third Edition*, Addison-Wesley, 2004.
12. J. Krogstie and A. Solvberg, *Information Systems Engineering: Conceptual Modelling in a Quality Perspective*, Kompendiumforlaget, Trondheim, 2003.
13. E. Letier and A. van Lamsweerde, *Deriving Operational Software Specifications from System Goals*, FSE'10, Charleston, November 2002.
14. Z. Manna and A. Pnueli, *The Reactive Behavior of Reactive and Concurrent System*, Springer-Verlag, 1992.
15. Australian National Audit Office, *Managing Parliamentary Workflow - Best Practice Guide*, April 2003.
16. C. Ponsard, *A Comparative Analysis of the French Community Parliament and the French Parliament of Brussels (in French)*, http://www.tabellio.org/documentation/manual/analyse-comparative-pcf-pfb-cetic, 2005.
17. J. Recker and J. Mendling, *On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages*, Proc. of 11th Int. Workshop on Exploring Modeling Methods in Systems Analysis and Design, June 2006.
18. W. Reisig, *Petri Nets: An Introduction*, Springer-Verlag New York, Inc., New York, NY, 1985.
19. Gunther Schefbeck, *E-Parliament: Legislative Standards and Good Practice*, Proceedings of International Workshop on E-Parliament: Managing Innovation, Geneva, Switzerland, 2007.
20. SoftwareAG, *XOooF*, http://xooof.sourceforge.net, 2006.
21. Telelogic, *Rhapsody*, http://www.telelogic.com/products/rhapsody.
22. A. van Lamsweerde, *Goal-Oriented Requirements Engineering: A Guided Tour*, Invited minitutorial, Proc. RE'01, August 2001.
23. Wikipedia, *Parliament*, http://en.wikipedia.org/wiki/Parliament, 2008.
24. Michael zur Muehlen, *Workflow-based Process Controlling: Foundation, Design, and Application of Workflow-driven Process Information Systems*, Logos Verlag, 2004.

Regulating Organizations: The ALIVE Approach*

Huib Aldewereld, Loris Penserini, Frank Dignum, and Virginia Dignum

Institute of Information and Computing Sciences
Universiteit Utrecht
P.O.Box 80089, 3508 TB Utrecht
The Netherlands

Abstract. Regulating organizations requires a fine balance between central control and (local) adaptability. In this paper we report on our approach using explicit organization and coordination models based on the research performed within the European FP7 project ALIVE. One of the principal aims of ALIVE is to combine coordination and organization mechanisms in order to provide a flexible, high-level means to model the structure of interactions between services in the environment. Our main focus is on the implementation and integration of organizational structures and on the translation of (abstract) norms (e.g., laws and regulations) into (concrete) software systems. Such a way of abstracting from low level system complexity by the use of human- and social- oriented system requirements is very promising to cope with requirements changes, e.g., useful to develop adaptive (service-based) systems.

Key words: Organizations, implementation, norms

1 Introduction

The deployment of regulations in human societies and in information systems show remarkable resemblances. Both fields need to cope with relating the abstract level of the regulations with the concrete practice (either the "work floor" of the human organization or the "low-level" software implementation, e.g., using service-based systems). A common tendency when relating the regulations to the practice is to directly connect the top (abstract) level with the concrete implementation, but in this paper we argue that the use of intermediate level(s) allows for a greater flexibility and, at the same time, an increased robustness of the system. This new source of (human- and social- oriented) system requirements pave the way for new challenges in software engineering. That is, recent software engineering approaches have dealt with how to endow single (agent-based) systems with the ability to cope with context changes, without taking

* This work has been performed in the framework of the FP7 project ALIVE IST-215890, which is funded by the European Community. The author(s) would like to acknowledge the contributions of his (their) colleagues from ALIVE Consortium (http://www.ist-alive.eu)

into account that such adaptivity properties can be easier and better studied and handled at organizational level, as often it happens in real life. A challenging aim of this paper is to bridge adaptivity at organizational level.

The research presented in this paper is part of the European FP7 project ALIVE, which aims to create a framework for software and service engineering, based on combinations of coordination and organization mechanisms [1, 3, 13, 14] (providing a flexible, high-level means to model the structure of interactions between services in the environment) and Model Driven Design (providing for automated transformations from models into multiple platforms). Although the main goal of the ALIVE project is to enhance the development and deployment of service-based (information) systems, we will show that the same approach can be applied to the deployment of regulations in human organizations.

The approach taken in the ALIVE project is to gradually translate the regulations from the abstract level of organizational regulations into a system description at the concrete level of service-based implementations. First, the abstract regulations are translated into operational norms and structures, which are more concrete than the regulations themselves. This translation is done by adding operational information to the regulations (i.e., how a given regulation can be achieved in a given context/domain). These operational artifacts [10], however, still abstract from the specific choices needed for the implementation (e.g., different checks to be made, specific system calls, etc.). The use of such an intermediate level is advantageous, because it, in essence, specifies the global objective of the organization in concrete terms, while still describing a family of implementations (i.e., the intermediate level allows for a flexible implementation). This means that this intermediate level contains enough information to make implementational changes without having to go back to the most abstract level of the organizational regulations (i.e., you change the implementation by choosing a different member of the family of implementations that is specified at the operational level).

One way of deploying regulations in an organization is by regimenting the participants of the organization and constrain them in such manners that they can only perform behaviour which the organization considers legal. That means, all possible actions are *a priori* defined by the organization. While, at first glance, this appears to be a fruitful approach, it has the major disadvantage that the system loses much of its flexibility and robustness. If, however, the participants are allowed to perform actions that are not described as allowed (such actions could be illegal, but could also be not considered a priori), the participants can (e.g., through exploration) come up with more effective ways of doing things and react to unexpected situations which were not taking into consideration when the organizational regulations were recorded. In this case, however, the safety of the system has to be guaranteed by sanctioning participants for doing illegal actions.

Throughout this paper we will use a generic, simple example based on a simple regulation to *regulate the temperature of the building at a comfortable level without wasting energy*. This can be seen or described as the regulation or norm

that the organization has to comply to; namely, the thermostat is *obliged* to keep a comfortable level of heat in the building without wasting energy. Before we translate this regulation into service specifications, combining the services needed to achieve this objective, we first create an operational description of the general objective. That is to say, we give an operational meaning to the regulation, which is informing the organization, still on an abstract level but more concrete than the norm/regulation itself, how the objective is to be reached. There are alternative mappings to operational descriptions possible for this example regulation. For now, let us assume that the operational meaning of the regulation is that *the temperature in the building needs to be 18 °C whenever there are people around*. Finally, this operational description of the regulation is used to combine the services (at the implementation level) in such a way that the regulation of the organization can be fulfilled. In this case, this might mean the combination of the following services:

- a service to get the day of the week;
- a service to get the time of day;
- services to get the temperature of every room in the building;
- a service to translate temperatures in °Fahrenheit to °Celsius;
- a service to regulate the heater/air-conditioner of the building.

The services for the time of day and day of the week are needed to determine whether there are people in the building (i.e., the system does not need to use the heater/air-conditioner during evenings and weekends). The translation service from °F to °C is only needed if not all services (that measure the temperature or regulate the heater/air-conditioner) are speaking the same "language".

In this paper we compare the approach of ALIVE, based on previous research done [1, 3, 13, 14], to the regulation of organizations in general. In the next section we give a broad overview of the ALIVE project. In section 3, we explain how the use of intermediate levels helps the deployment of regulations. We present our ideas about how the transition of regulations from an abstract organizational point of view can be made to the concrete practice. Moreover, we present our ideas about how to cope with adaptivity and how this affects organizational structures. We end the paper with some conclusions.

2 The ALIVE approach

New generations of networked applications based on the notion of software services that can be dynamically deployed, adjusted and composed will make it possible to create radically new types of software systems. In turn, this will require profound changes in the way in which software systems are designed, deployed and managed – exchanging existing, primarily top-down "design in isolation" engineering, to new approaches which are based on integrating new functionalities and behaviours into existing running systems already active, distributed and interdependent processes.

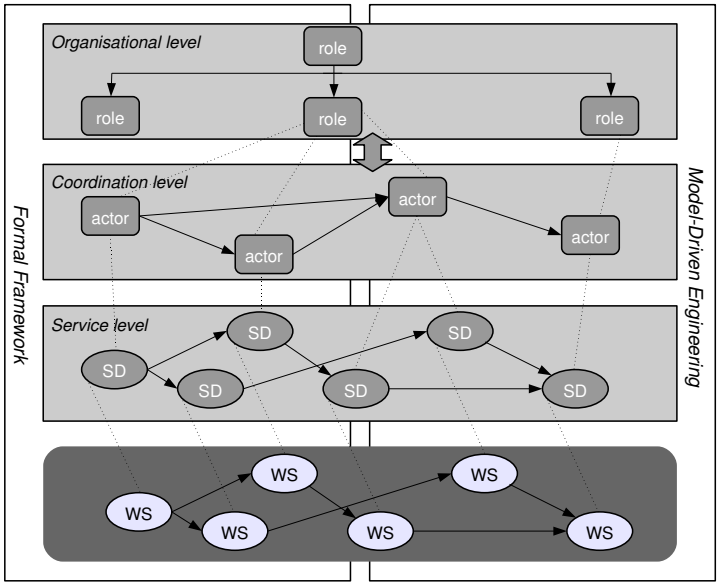

Fig. 1. The ALIVE framework for software and service engineering.

The ALIVE project is based around the central idea that many strategies used today to organize the vastly complex interdependencies found in human social, economic behaviour will be essential to structuring future service-based software systems. More specifically, the project aims to combine cutting edge Coordination and Organization mechanisms and Model Driven Design to create a framework for software and service engineering for "live" open systems of active services.

The project extends current trends in service-oriented engineering by adding three extra layers (see Figure 1).

- The *Service Layer* augments and extends existing service models with semantic descriptions (SD) to make components aware of their social context and of the rules of engagement with other web services (WS).
- The *Coordination layer* provides the means to specify, at a high level, the patterns of interaction between services, using a variety of powerful coordination techniques from recent European research in the area.
- The *Organization Layer* provides context for the other levels – specifying the organizational rules that govern interaction and using recent developments in organizational dynamics to allow the structural adaptation of distributed systems over time.

In the following sections we focus mainly on the connections between the organizational level (where the regulations reside) and the service level by using

an intermediate level. We show how the ideas of ALIVE relate to the deployment of regulations in human organizations and allow for flexible adaptation.

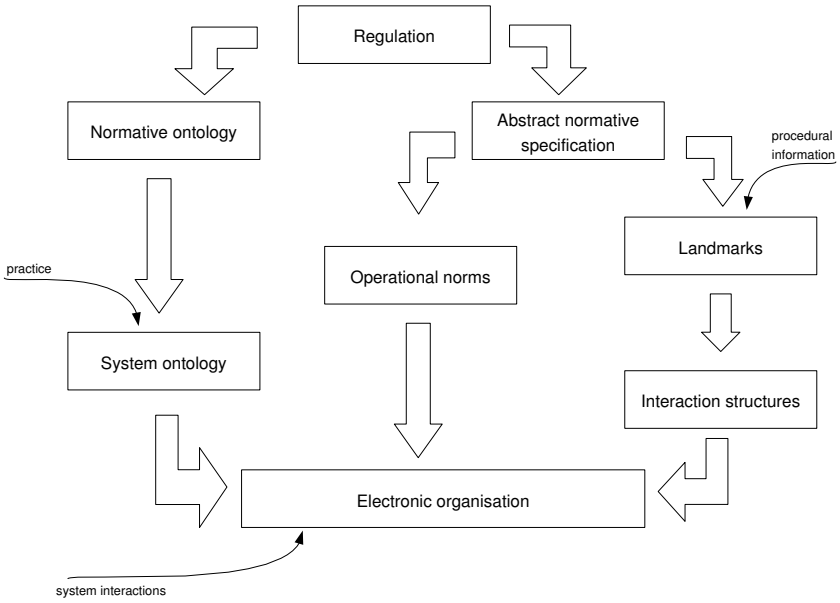

Fig. 2. From laws to electronic organizations.

3 From Abstract Regulation to Implementation

The deployment of regulations in the ALIVE approach consists of a gradual transition from the organizational level to the service-based implementation (the bottom two levels of the model in Figure 1). Given that organizations are characterized by their rules and conventions [1, 3], this process of implementing organizational regulations is then as proposed in Figure 2.

First, a formal representation of the regulations is created, giving an *abstract normative specification* of the allowed interactions in the organization (e.g., in deontic logic). Given our example, this means a formalization like, e.g., $O_{thermo}(temperature(comfortable))$ and $F_{thermo}(waste(energy))$. Which states that *thermo* is obliged to make sure that the temperature is comfortable and *thermo* is forbidden to waste energy. The creation of a formal representation of the regulations also creates a basis for the ontology that is needed (we call this basis the *normative ontology*). The normative ontology is built from: 1) the concepts and relations used in the formalization step, and 2) information taken

from the ontological definitions in the regulations themselves. In our example, the normative ontology contains the concepts of $comfortable$, $temperature$, $energy$, etc. The normative ontology and the formal representation of the regulations can be seen as the organizational level of Figure 1.

The normative specification is also used as the basis of the implementation of the regulations. The process from normative specification to implemented norms is as follows: 1) the abstract norms are translated to concrete *operational norms* (although these are only useable for a certain context, i.e., this particular organization, and less expressive than abstract norms, concrete norms are a lot easier to implement); 2) the operational norms are translated into constraints and procedures that will see to it that the norm is enforced in the organization. In our example, the operational norm is *the temperature in the building needs to be 18 °C whenever there is people around.* The design of *interaction structures* that can be used in the organization consists of the following steps: 1) the important characteristics of the norms that express how interactions should be in the organization are extracted from the norms to create a prototypical interaction structure on a high level of abstraction (we call these important steps derived from the norms *landmarks*, and the structure that expresses the ordering over these landmarks a *landmark pattern*); 2) by using procedural information and the expected capabilities of the system components an *interaction structure* is created to give a default manner for achieving certain objectives in the organization. For our example we can create an interaction structure by using landmarks (e.g., $L_1 =$*check temperature* and $L_2 =$*adjust heater*, with the temporal ordering that $L_1 < L_2$): $if(today = normal\_weekday)\ then\ temperature := requestTemperature(service_{temp});\ if(temperature \leq 18)\ then\ turnHeaterOn.$ The operational norms and landmark patterns provide the intermediate level of the transition from organizational regulations to a implementation. This level can be seen as the coordination level as shown in Figure 1.

Finally, the *system ontology*, which contains all concepts used in the norms as well as those used in the implementation is build from the normative ontology. The normative ontology is extended with the concepts and relations that follow from the operational and procedural information that was added to create the operational norms and the interaction structures. Moreover, concepts describing the system states and actions need to be added and linked as well.

As shown in Figure 2, the following four elements are of prime importance when implementing regulations in organizations:

- A common **ontology**, defining the meaning of concepts, the roles used in the organization and the relations between different contexts.
- A **normative specification** of the allowed interactions in the organization.
- **Interaction structures** to specify conventions in procedure mechanisms, giving a typical interaction profile which should work in any circumstance.
- An active **enforcement mechanism** to make sure that the participants of the organization adhere to the normative specification.

The ontology is needed to specify how the participants interact, defining the communicative propositions that are used, and defining the roles and role hi-

erarchy that is used throughout the norms. The normative specification is the basis of the organization, specifying the legal and illegal actions in the environment. Denoted in a formal language, this specification can be used to derive the last two elements of the framework. The interaction structures define standard ways in which the legal interactions can take place in the organization. They provide a means for non-norm aware participants to perform their task in the organization, or provide a guideline for norm-aware participants to follow (to show how things can be done, though are not necessarily the only way to do it, and can be deviated from if need arises). The norm enforcement is necessary to guarantee the safety of the system. Since we do not restrict the participants of the organization to only perform the allowed actions, the organization is required to check and enforce the proper ways of acting upon the participants in the organization. Much like in the real-world, instead of equipping all cars with speed-limiting devices, one specifies that speeding is illegal, and checks whether everyone adheres to that norm (even if one would opt for the regimented option of installing speed-limiting devices in cars, one would still have to check that no one tampers with the device and violates the norm).

An important step of this deployment process is the addition of operational information (taken from practice or procedures) to create an intermediate level (in Figure 2; the *operational norms* and the *landmarks*) that tries to capture the essence of the organizational level, but brings it closer to the actual implementation. Let us look at the addition of such information in more detail.

Adding Operational Information

The translation from organizational regulations from natural language to a formal representation (the abstract normative specification) is only the first step of the process of implementing the regulations. Usually the regulations are expressed at a high level of abstraction, to allow the regulation to cover a wide variety of situations and to be used for an extensive period of time without the need for modifications, it is hard to link these regulations to the concrete situations that arise in the practice. To make the normative specification useful in the deployment of the organization, an interpretation of the norm is needed, which should contain concrete (organizational) meanings of the vague and abstract terms used in the norm and which possibly contains procedural information that can be used to simplify the enforcement of the norm. This process of interpreting the norms to make them useable for a single context, i.e., the organization, is referred to as *contextualization* [1].

The contextualization process is meant to give a link between the abstract terms and concepts used in the abstract normative specification and the concrete situations and concepts that exist in the practice. Where norms contain terms such as 'fair' and talk about actions like 'discriminating', these concepts have no clear meaning in the implementation. There are, however, states and (sequences of) action(s) in the implementation that can be classified as an interpretation of one of these vague concepts in the *context of the organization*. These interpretations are highly context dependent and can differ from organization

to organization. For example, in accordance with the example described in the introduction, the abstract norm *regulate the heat of the building at a comfortable level without wasting energy* is contextualized (e.g., based on the preferences of the people that work in the buildings) to *the temperature in the building needs to be 18 °C whenever there are people around*. In another implementation, however, it could be something different, e.g., *the heater should be turned off at night or when the temperature is above 68 °F*.

Although norms that result from the contextualization process are concrete and contain only concepts that are meaningful in the organization, these norms still require further explicification before they can be implemented. Norms only have a declarative meaning, i.e., how things should be, while abstracting from operational meanings, which expresses how it should be achieved. Moreover, there is more than one way to enforce a single norm and procedural information (which is not part of the norm) will have to be used to decide how the norm is best implemented. This second translation process of adding additional operational and procedural information to the norms is referred to as *operationalisation* [1].

In the next section, taking advantage of recent results from adaptive system engineering approaches, we show our vision about how to cope with adaptivity requirements at the organizational level.

4 Introducing Adaptivity in organizations

Implementing regulations for organizations that are completely static is quite straightforward. The real challenge comes when the circumstances change and the organization needs to adapt to the new situation while still trying to abide by the regulations. In this section, we focus on those features of the proposed organization framework to effectively deal with context changes, namely, how the organizational models for the intended (service-based) system adapts to different kinds of changes. Before going into detail how our approach achieves adaptivity qualities, let us first look at how adaptivity is handled in other (recent) approaches.

A very compelling research topic within the area of software engineering regards methods, architectures, algorithms, techniques, and tools that can be used to support the development of adaptive systems. That is, software engineers are looking for techniques to model important requirements for adaptive software systems such as the ability to cope with changes of stakeholders' needs, changes in the operational environment, and resource variability. On one hand, a quite recent and interesting example of adaptive systems is IBM's work on autonomic software systems [5, 7]. Such a software type is characterised by properties of being able to automatically re-configure itself when new components come into or are removed from the system (self-configuration); being able to continually tune its parameters for optimisation (self-optimisation); being able to monitor, analyse, and recover from faults and failures when they occur (self-healing); and being able to protect itself from malicious attacks (self-protection).

On the other hand, promising software engineering approaches have recently adopted goal-oriented methodologies with an extensive use of goal models (GMs), which have been initially proposed in Distributed Artificial Intelligence as a means for capturing agent intentions and guiding agent coordination [6, 8] within dynamic environments. Within such methodologies, requirements are elicited, specified, and elaborated using the concept of goal, which can be used to model stakeholder and organizational objectives, but also an agent goal. In other words, the goal concept allows designer to represent high-level (strategic) concerns.

In [9, 11], GMs allow a designer to represent and reason about stakeholder objectives and agent goals in a given application domain in order to derive requirements for adaptive software. According to these approaches, GMs give support in exploring and evaluating alternative solutions which can meet stakeholders expectations (objectives) and in detecting conflicts that may arise from multiple viewpoints (see also [12]).

The above approaches identify several crucial components that a development framework should take into account to effectively deal with software adaptivity. Nevertheless, how such requirements affect organizational structures has not been completely addressed. Finally, in [2, 4] interesting approaches to cope with reorganization issues have been presented. The principal aim in [2, 4] has been to develop a modelling language to describe organizational structures, and how their objectives are related to changes in the environment. Specifically, a simulator framework, where agents play modeled organizational roles having different objectives, has been adopted to test reorganization behaviours to cope with changes.

Adaptivity within organizations

Results from the above approaches, related to specifying the system adaptivity, are useful to properly interpret and reflect such requirements at the organizational level. Specifically, we aim at illustrating by simple example scenarios that our framework (see Figure 1) can distribute the complexity –to handle context changes– among its different layers. This latter property is important to improve the flexibility and robustness of the (service-based) system. Adaptations on the lower level might violate specific procedural interpretations of a regulation, but still comply to the more abstract regulation specified on a higher level. When such a situation occurs one can now change the operationalizaion of the regulation such that the new practices conform to these procedures, while preserving the same regulation at an abstract level.

Principal sources/causes of dynamic changes in the context can be described as follows.

Stakeholder needs. Changes in the stakeholder needs happen frequently in open organizations where new roles may be added and old ones are detached in order to better reflect the market changes. In other words, stakeholder needs have to be strictly related with organizational objectives to effectively deal with changes of needs, e.g., re-adapting to new organization market strategies. Ac-

cording to our example, let us assume an enterprise [1] has to pursue the objective *make employees comfortable* and to do that it depends on the work and quality of thermostat devices of all departments provided by a thermostat organization. Then, let us consider that because of the market strategy, the area-manager changes her needs, delegating to each department-manager the objective *minimise heating costs* to pursue too, which requires reorganising its internal service providing structure. This change may result in selecting another service to play the role *thermostat* that is cheaper, by e.g. auctioning a new offer. Notice that, according to Figure 1, this change is sensed at organizational level but handled at coordination level.

Environment conditions. Depending on the kind of application domain, such requirements have to reflect real life situations into the organizational behaviour, e.g., symptoms to be forecasted (at design-time) and then anticipated (at runtime) to avoid failures in pursuing objectives. Moreover, such requirements deal also with how norms can affect and are related to organization objectives. According to our example, let us assume that the thermostat has a digital power meter (*services to get the temperature of every room in the building*) in order to maintain its objective *regulate the heat of the building to a comfortable level without wasting energy*. This service periodically verifies whether the consumed energy in each building correctly stays into a specific range. Let us also assume that, during the winter time, a couple of employees went on holiday but forgot to close their office windows. This environment change (symptom) could cause the failure of the previous objective (expressed in the normative specification) if no countermeasures (enforcement mechanisms) have been considered in advance to properly handle such a fault symptom. The enforcement has to trigger another objective achievement, e.g., asking to the building attendant to check all the windows, therefore such a change mainly affects the coordination level of Figure 1. Moreover, to get such a process completely automated, the reasoning mechanisms have to be supported by (domain) ontologies that describe symptoms, faults, recovery objectives, and roles along their relationships.

System functionalities. Although the modelling of the organizational knowledge level has a key role within the whole framework, role and objective concepts need to be properly grounded into specific system functionalities (agent capabilities and/or service functionalities) in order to really affect and sense the environment. In other words, changes in environment and in stakeholder needs (discussed above) inherently are reflected in the orchestration process of the service level of Figure 1, following an implicit top-down approach (see Figure 2). In the other hand, changes in system functionalities deal with a bottom-up propagation, namely, several dynamic issues can arise from the service-level and, consequently, need to be related and handled by the organizational and coordination levels. Let us consider the example sketched in Section 1, where the thermostat has to maintain the regulation *the temperature in the building needs*

[1] According to the example of Section 1, this enterprise acts as the committer for the thermostat organization (supplier), i.e., roles commonly played within any organization.

to be 18 °C whenever there are people around (O_1). To achieve this objective, the information system has to orchestrate different services and then combine their results collected every time, e.g., get the day of the week ($s_1^{O_1}$), get the time of day ($s_2^{O_1}$), translate temperatures from °Fahrenheit to °Celsius ($s_3^{O_1}$) because the thermostat device works in °Fahrenheit, calculate whether the sensed temperature is in the established range ($s_4^{O_1}$), and sense the environment for people presence ($s_5^{O_1}$). Now, let us assume that at the time O_1 had to be achieved, the system recognizes that $s_3^{O_1}$ is not available. *Where and how to handle this sensed change?* Maybe the service level (the *where*) has been provided with some simple recovery function (the *how*) such as searching for an equivalent service. But, the most compelling scenario arises when the service level brings about some important failure (e.g. no other equivalent service available), propagating it to the next-up level. Again using different levels of abstraction in the specification now allows for different solutions using different types of knowledge present at those levels.

5 Conclusions and Future Work

In this paper we presented how the ALIVE approach can be used for the deployment of regulations and the reorganization of both human societies and information systems. The ALIVE project aims to create a framework that combines coordination and organization mechanisms in order to provide flexible, high-level models to assist software and service engineering. A main element of the ALIVE approach is to distribute the design of coordination and organization of service-based implementations over different levels of abstraction. At the highest level of abstraction (the organizational level) the context is defined in terms of abstract regulations and objectives. These abstract norms are operationalized in the next level of abstraction (the coordination level), where operational and contextual information is added taken from procedures and practice. The lowest level of abstraction (the service level) then deploys the operational norms and structures defined on the coordination level to create a service-based implementation.

The process of translating the abstract regulation to an implementation has been illustrated. In this process, the main elements are 1) an abstract normative specification, 2) a common system ontology, 3) a set of interaction structures describing default interactions, and 4) mechanisms for enforcing the norms to guarantee safety in the system. Moreover, we have shown that the translation or regulations to an implementation in practice is not a straight-forward, one-step process. The organizational regulations have to be *contextualized* and *operationalized* before they can be implemented. That is, the abstract regulations need to be translated into more concrete regulations which use only concrete concepts and relations (which are context dependent) and operational information, to express how the regulation is to be achieved/maintained, has to be added.

This paper also reports on how the proposed framework naturally fits for the modelling of context (requirements) changes to better reflect real organization behaviours. Moreover, taking advantage from system specification of self-

adaptive systems, we have shown how the framework can handle such requirements at organizational and coordination levels.

As future work, we are interesting to investigate how the organizational framework should behave to deal with context changes that are not within the organization knowledge, e.g., new objectives, roles, and relationships not described in the domain ontology. This challenging research aspect is very related to both the evolutionary design and the evolutionary qualities of agent systems, namely, how to automatically update organization models from new knowledge that emerges from the service level.

References

1. H. Aldewereld. *Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols*. PhD thesis, Universiteit Utrecht, June 2007.
2. F. Dignum, V. Dignum, and L. SonenBerg. Exploring congruence between organizational structure and task performance: a simulation approach. In *Coordination, Organisation, Instiutions and Norms in Agent Systems I*, LNAI 3913, 2006.
3. V. Dignum. *A Model for Organizational Interaction: based on Agents, founded in Logic*. PhD thesis, Universiteit Utrecht, 2004.
4. V. Dignum and C. Tick. Agent-based Analysis of Organizations: Performance and Adaptation. In *2004 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2007)*, California, USA, 2007. IEEE CS Press.
5. A. G. Ganek and T. A. Corbi. The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1):5–18, 2003.
6. N. Jennings. *Foundations of Distributed Artificial Intelligence*, chapter Coordination Techniques for Distributed Artificial Intelligence. Wiley-IEEE, 1996.
7. J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer, IEEE Computer Society Press*, 36(1):41–50, 2003.
8. V. Lesser. A retrospective view of fa/c distributed problem solving. In *Systems, Man and Cybernetics, IEEE Transactions on*, volume 21, pages 1347–1362. 1991.
9. M. Morandini, L. Penserini, and A. Perini. Towards Goal-Oriented Development of Self-Adaptive Systems. In *Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2008)*. ACM and IEEE digital libraries, to appear, 2008.
10. A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini. Coordination artifacts: Environment-based coordination for intelligent agents. In *Proc. of the 3rd Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS 2004)*, pages 286–293. ACM Press, 2004.
11. L. Penserini, A. Perini, A. Susi, and J. Mylopoulos. High Variability Design for Software Agents: Extending Tropos. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 2(4), 2007.
12. A. van Lamsweerde and E. Letier. Handling obstacles in goal-oriented requirements engineering. *IEEE Transactions on Software Engineering, Special Issue on Exception Handling*, 26(10), 2000.
13. J. Vázquez-Salceda. *The Role of Norms and Electronic Institutions in Multi-Agent Systems. The HARMONIA framework*. Whitestein Series in Software Agent Technology. Birkhäuser Verlag, 2004.
14. J. Vázquez-Salceda, V. Dignum, and F. Dignum. Organising multiagent systems. *JAAMAS*, 11(3):307–360, November 2005.

Towards Modeling Civil Aviation Safety Legislations

Eduardo R. López Ruiz[1] and Michel Lemoine[1]

[1]ONERA. 2, avenue Edouard Belin,
Toulouse, France. 31400
{eduardo.lopez-ruiz, michel.lemoine}@onera.fr

Abstract. Building on the work of the EDEMOI methodology, this paper proposes a twofold expansion of this methodology consisting in: (1) broadening its scope to include aviation safety legislations and (2) extending its usability to detect regression originating from regulatory amendments. Accordingly, this paper analyses the differences between safety and security legislations in civil aviation, develops on their similarities and proposes a tailored graphical model apposite for safety legislations. Finally, this paper defines the case-study in which the proposed graphical model will be initially implemented.

Keywords: Graphical specification, legislations.

1 Introduction

Aeronautics is an industry that is highly aware of the need to incorporate human factors science and engineering into its different domains to further improve safety and security. This includes the domain of printed materials. Accordingly, international aviation organizations, research centers and some aviation authorities have conducted human factor studies aimed at characterizing and improving the semiotic (i.e. the semantics, syntactics and pragmatics), visual and structural quality of their different printed materials[1] [1]. However, until recently, these studies and standards have mostly targeted the clarity, readability and legibility of the printed materials but not their *embedded logic*.

Yet, operational feedbacks have hinted that the effectiveness of these printed texts also depends on logical traits such as: their *consistency* and their *robustness*. This is of great consequence since (analogously with safety-critical software) these traits ensure that the benchmark legislation being enforced is not inherently rendered ineffective due to contradictory policies (either by themselves or globally), and that it exhaustively covers all the possible scenarios within its domain of application.

Consequently (and within the backdrop of September 11th, 2001), a group of French universities and research laboratories sought to enhance the rulemaking process used by civil aviation organizations and authorities, to create and validate *security* legislations. Their proposal was to incorporate simulation and counterexample checking tools into the legislations' validation phase, to better ensure

[1] Support material, training material, procedure manuals and checklists, etc.

their embedded logic. To this end, they propounded the utilization of formal methods to specify and validate aeronautical security-related requirements.

In fact, conscious of the necessities of civil aviation authorities, they proposed a specially conceived specification methodology that took into account the intricacies of formal notations and the familiarity needed for their comprehension (See Figure 1).

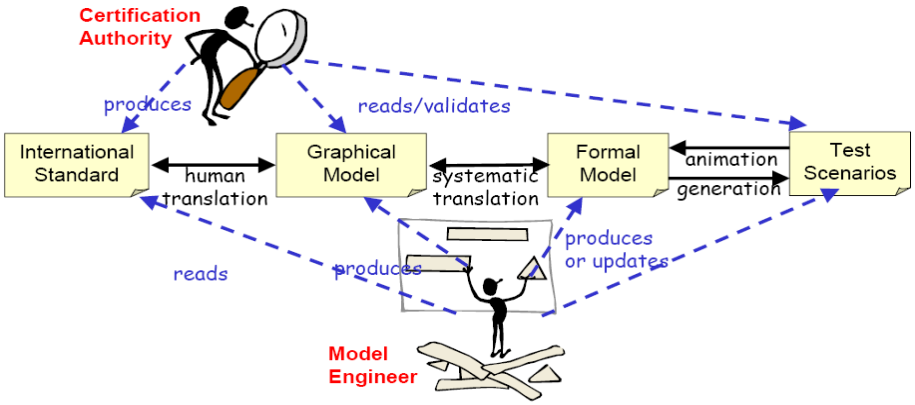

Fig. 1. In the first step of the EDEMOI approach, a *Model Engineer* extracts the security goals imposed by an *International Standard* and translates them into a *Graphical Model* that faithfully represents their structure and relations (while reducing the use of inherently ambiguous terms). Once this Graphical Model has been revised and validated by the *Certification Authority*, the *Model Engineer* performs a systematic translation of the Graphical Model to produce an implicitly valid *Formal Model*, which can be later analyzed using *Test Scenarios*.

This methodology, referred to as the EDEMOI methodology, has been implemented to the modeling of both international [2] and European security legislations [3]. In both cases, the analysis of passenger-related security standards was emphasized. These standards were translated into formal models using the B and Z notations and animated [4]. Thanks to this, its appropriateness (i.e. its aptitude to specify and assist in the design and validation of security requirements) has been established.

Still, as civil aviation authorities are concerned with ensuring both the security and the safety of civil aviation, and given that new legislations are evolutions of existing ones (prompting the study of their non-regression), an expansion of the EDEMOI methodology has been proposed. This expansion consists in: (1) broadening its scope to include aviation safety legislations and (2) extending its usability to detect regression originating from regulatory amendments.

Neither one of these two aspects can be considered as a simple, straightforward effort, given that there are fundamental differences between security and safety legislations. Therefore, their realization will entail a change in the techniques proposed within the EDEMOI methodology, to focus on the specificities of safety-related requirements. Additionally the study of the non-regressions is an endeavor on its own, based on the use of animation and proof techniques to compare successive versions and detect regressions.

In Section 2 of this paper, the differences between safety and security legislations will be discussed, emphasizing on how the EDEMOI methodology (its methods and tools) can contribute to improving safety legislations. Section 3 will highlight the new role that will be given to the legislations applicability criteria in the modeling of aviation safety legislations. Then, Section 4 shall propose a tailored graphical method apposite for safety requirements, and its use in a future case study to illustrate its benefits. Finally, Section 5 will draw the conclusions and perspectives of this work.

2 The Differences between Security and Safety Legislations

As mentioned previously, the expansion of the EDEMOI methodology has two objectives: firstly, to adapt it for a suitable implementation in the analysis of safety legislations and, secondly, to facilitate the analysis of regression between succeeding versions. In order to achieve the first objective, we need to have a very clear understanding of the differences between safety and security legislations. Furthermore, we need to correctly identify what civil aviation authorities seek in terms of improving safety legislations.

So, the first considerable difference between safety and security legislations is their purpose. That is to say, safety legislations focus on preventing accidental events (detrimental to civil aviation) while security legislations are focused on the prevention of intentional acts (detrimental to aircraft, airport infrastructures, persons, etc). For this reason, their legislative domains are markedly dissimilar in terms of coverage size and participating stakeholders.

Security legislations are implemented within a relatively small and contained domain, covering the airport areas (including off-site security zones), their perimeter and the aircraft's interiors. Conversely, for safety legislations, their corresponding domain is harder to limit, since civil aviation safety is a collaborative contribution of the aircraft's initial and continual airworthiness, its operation and also of navigation and control services. Moreover, the safety requirements for a specified element will vary in function of its geopolitical location and the type of operations it is performing. So, an aircraft that is entering European airspace will “automatically” be subjected to safety obligations that were not applicable the instant before.

Additionally, in terms of legislative evolution, it is primarily safety legislations that need to be more adaptive to the industry's constantly evolving state-of-affairs, helping steer developments instead of contriving their progress. This refers to the fact that, in aeronautics, advancements are the result of a fragile compromise between what is technologically achievable, what is economically profitable and what is cautiously acceptable. For this reason, civil aviation authorities must be careful not to impose unduly or unjustifiable safety requirements, as they might hinder future developments.

Nevertheless, safety and security legislations do have some commonalities. The most important is that they impose their requirements using 'directive statements'. Moreover, their requirements can be classified on the basis of their approach, and are said to be either *objective based* or *prescriptive* requirements. The difference between these approaches is that the first sets targets or goals to be met but provides flexibility

in terms of how they are met; while the second does not offer such flexibility and instead, details how these must be met. However, most requirements cannot be easily classified as either one or the other, but rather as a mix.

For example, a largely *objective based* safety requirement that has been central to the design of commercial aircraft is that (for the aircraft and its subsystems), there must be *an inverse relationship between the probability of a failure and the consequences of said failure*. In other words, the most dangerous failures should have the lowest probability of occurrence, yet it is up to the designer and manufacturers of aircraft (and their subsystems) to come up with ways to ensure this. In contrast, a largely *prescriptive* safety requirement could impose specific design constraints. For instance, authorities impose the number and types of emergency exits required for an airplane (given certain conditions), in order to maximize the probability of its safe evacuation.

Yet, given the intrinsic nature of all **'directive statements',** we are confident that the underlying principle of the EDEMOI methodology (i.e. the use of formal methods to specify and validate their embedded logic) is still valid for civil aviation safety requirements. However, the usefulness of this expansion will be limited to certain domains of safety legislation. Tentatively, to the domains whose legislation is not highly objective based, such as: the operation of aircraft, the provision of air traffic services, aerodrome operation and aircraft airworthiness standards. As was the case in security legislations, a formal specification outside of these domains would be, either irrelevant or particularly ineffectual.

In what concerns regression analysis, we believe that the graphical modeling technique presented in [2] helps facilitate the detection of certain types of regressions independently of their safety or security nature, mainly thanks to the model's tree-like structuring which is rooted from a safety/security property and expands outwards (See Figure 2).

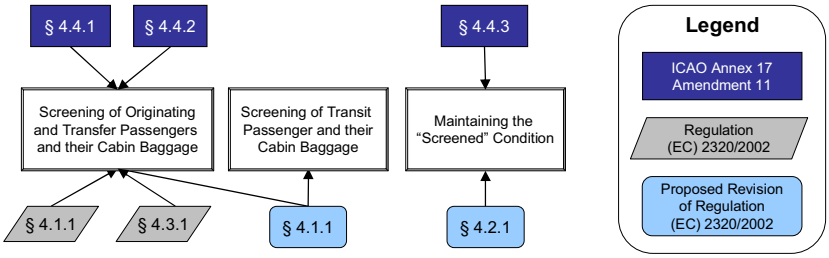

Fig. 2. The "security property" approach to modeling security legislations helps improve the management and traceability of such documents. Consequently, these types of model help detect certain forms of regressions. For instance, international standards such as ICAO's Annex 17 need to be further specified and adapted, before being enacted at a national level. In the case of Europe, this specification came in the form of Regulation (EC) 2320/2002. As illustrated above. The initial version of Regulation (EC) 2320/2002 (symbolized in a parallelogram) did not impose requirements concerning the prevention of unauthorized interference with screened passengers and baggage. However, both its founding text, *ICAO's Annex 17* and its *Proposed Revision* do contain such requirements (rectangle and a rounded-rectangle respectively).

Finally, having established the traits that are the most relevant (to civil aviation authorities) for safety legislations, we consider that a model around the requirements' *applicability criteria* will be an insightful tool for aviation authorities (as will be discussed in Sections 3 and 4).

3 The Applicability Criteria

Applicability criteria are used in legislations to explicitly define the set of elements upon which a set of requirements will be imposed. For example, the following statement (taken from ICAO's Annex VI) explicitly states a condition that is applicable to "All flight crew members…on flight deck duty".

> *ICAO - Annex VI §4.4.4.1 Take-off and landing. **All flight crew members** required to be **on flight deck duty** shall be at their stations.*

Moreover, this condition is only applicable during a particular moment, "[throughout the aircraft's] Take-off and landing [phases]".

Hence, in the case of civil aviation legislations, the applicability criteria will be a general element (e.g. "an aircraft"), an element in a specific state (e.g. "all flight crew members required to be on flight deck duty") or only a state (e.g. "during take-off and landing").

As these criterion and states are at the core of the legislative texts, their clear understanding is of high importance. This is why some legislations provide generic definitions of the elements and states invoked by their applicability criteria. For example, ICAO – Annex VI provides the following definition of a *Flight Crew Member*:

> "*A licensed crew member charged with duties essential to the operation of an aircraft during a flight duty period*".

At any rate, applicability criteria are a rich source of information. They can be used to deduce the different elements affected by the legislation, their allowed operations and states.

For instance, by combining the definition given for a *flight crew member,* with the requirement §4.4.4.1 (referred to above), we can tentatively deduce that all *required Flight Crew Members* will be in one of the two following opposed states: "*not on flight deck duty*" and "*on flight deck duty*".

Moreover, we suspect that there is a trigger operation that fires a transition of the flight crew member from the first to the second state (and that, from an implementation perspective, such trigger operation would occur only during the flight crew member's flight duty period. Hence the word *required* in §4.4.4.1).

The EDEMOI methodology used this type of reasoning to build the graphical models which comprised the legislation's application domain. In this case a "class" *Flight Crew Member* would be proposed, with two Boolean attributes:

"*on_flight_deck_duty*" and *"on_flight_duty_period"*. Similarly, a number of representative operations would be generated to modify these attributes.

Simply, legislations can be regarded as a function which associates a set of applicability criteria to their corresponding set of safety and security requirements. As a result, the applicability criteria are an important constituent of legislative documents, central to their implementation. As such, they are a very familiar concept for civil aviation authorities; and it could be expected that, for this same reason, aviation authorities would be responsive to graphical models founded on these criteria.

In addition, applicability criterion can help understand the underlying justification of a given requirement. In particular in the context of safety requirements, applicability criteria are chosen on the basis that they are criterion relevant to the known (or likely) safety risks. Therefore it seems desirable that a graphical model of the legislation should be able to (implicitly or explicitly) show this relation, to substantiate that a given safety requirement is not unwarrantedly or wrongly imposed.

For example, in 1964 the U.S. Federal Aviation Agency (FAA) sought to amend the flight engineer requirements set fourth in three of its safety documents (CAR Sections 40.263, 41.263, and 42,263[2]). These requirements imposed a three person flight crew (the pilot, copilot and a flight engineer) on all civil airplanes[3] with a *maximum certificated takeoff weight (MCTOW) of more than 80,000 pounds* and *on all four-engine airplanes weighing more than 30,000 pounds MCTOW (when deemed necessary for the safe operation of the airplane)*[5].

The underlying reason behind this requirement was that, in the early days of aviation, the weight of the aircraft (and the number of engines it had) was representative of its size, which in turn was representative of it operational complexity. However, by 1964, this was no longer true, and the implementation of these requirements resulted in the employment of an additional flight-crew member without it contributing materially to the safety of the flight.

For this reason, an amendment was adopted prescribing broad standards to establish the minimum flight crew. This involved a shift in the requirement's applicability criteria, moving from the airplane's weight (which is a quantifiable but loosely representative criterion) to the workload involved in the airplane's operation (which is an unquantifiable but largely representative criterion).

This situation -where a set of requirements are no longer adequately enforced because their applicability criterion is no longer representative of the operational reality- is reasonably common within civil aviation, and it is mainly caused by the adoption of break-away technology.

Now, given that the capability to properly sustain the integrity of the legislative structure depends heavily in the timely anticipation and prevention of legislative incompatibilities, it is imperative that the EDEMOI extension takes into account such situations, and provides a tool to facilitate their detection and emendation.

Under these circumstances, a graphical model that is centered on the legislation’s applicability criteria is very informative, and might prove valuable for undertaking this type of comprehensive legislative enhancements.

---

[2] Now 14 CFR Part 121.

[3] Used in operations governed by these parts.

4 Proposing an Extension

Given the specificities of safety legislations, we consider that an enhancement in the EDEMOI methodology concerning the use of graphical models is warranted.

As was discussed in the previous section, safety legislations cover a very wide domain, with various domain-specific legislations governing a unique aspect. However, as each of these legislations may deal with a different aspect of a same element, there is a need for a tool that helps verify their inter-legislative coherence. This can be achieved by mapping the associations between the applicability criteria (i.e. the elements and/or states) and the safety requirements.

Yet, given that safety requirements may be pressed to evolve (in reaction to changes in civil aviation), the mapping of their association to the applicability criteria should be complemented with that of the safety risk that they are targeting (Refer to Figure 2), and the safety outcome they mean to provide.

Therefore we propose the creation of an interactive (adaptable) graphical model, centered on the legislation's applicability criteria, which will afford a pithy description of the safety requirements by:

- mapping out the association between the applicability criteria and the safety requirements,
- singling out the known or likely safety risk addressed by the different safety requirements (as well as the elements invoked), and
- highlighting the structure and hierarchy of the legislative texts and documents.

This graphical model would build on the strengths of the previously proposed EDEMOI models. Especially in terms of: (1) highlighting the structure and hierarchy of the legislative documents, and (2) enabling the analysis of regressions (mainly those arising from the suppression of previously enacted requirements). For this reason, our interactive graphical model will be a complementary tool within the EDEMOI methodology, specially designed for safety legislations.

4.1 The Graphical Model Proposed

The graphical model that we are proposing –in order to answer to the specific needs of safety legislations- would result from the aggregation of multiple nuclear diagrams (in theory, one diagram per requirement).

These nuclear diagrams are intended to (1) delineate the **applicability criteria** of each requirement (including intricate relationships amongst these criteria, such as: signs of aggregation, conjunction, disjunction, sequential and/or temporal conditions, etc), (2) identify the **elements summoned** (affected or addressed), (3) associate the requirements with the **known (or likely) safety risks** they address, and (4) state the

desired **safety outcome** due to compliance with the requirement (i.e. the desired condition/state).

We propose a form of "spray diagram" combined with a "cause-and-effect diagram" (See Figure 3), in which the **applicability criteria** and **partaking elements** are connected to the **legislative provision** in which they are referenced. In addition, the diagram can include additional information such as the **mitigating measures prescribed** (if any), the **safety risk** being targeted and the measure's expected **outcome**.

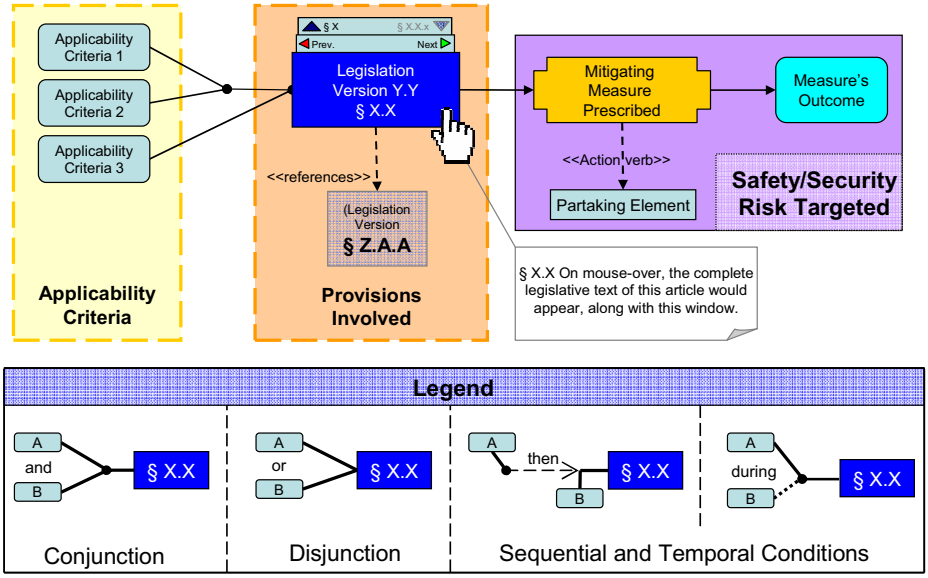

Fig. 3. The above figure is a representation of the nuclear structure of a safety requirement. An important characteristic of this diagram is the breakup of the requirement into three main particles: its *Applicability Criteria*, its *Provisions Involved* and the *Safety/Security Risk Targeted.* Indeed, this last particle encapsulates both the *Mitigating Measure Prescribed* and its expected *Outcome*. A partial caption is shown in the lower part. In the main diagram, the safety requirement is pertinent in either of two cases: (1) if Applicability Criteria 1 and 2 are both satisfied, or (2) if Applicability Criteria 3 is satisfied.

But, as we wish to continue conveying the structure (sections and sub-sections) of the legislative document, as the previously proposed EDEMOI model (See Figure 2), we are obliged to propose an interactive model whose visual structuring would be altered by the user, to facilitate specific browsing requirements. The extracted views of the model would resemble what is shown in Figure 4.

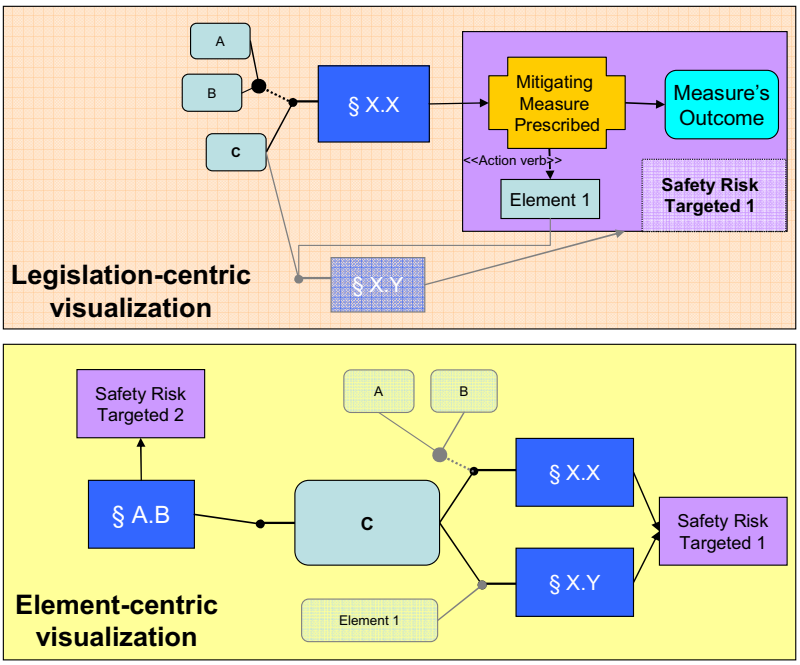

Fig. 4. Although the model's visualization will be centric (i.e. emanating from a single element), its root element will be changeable. The upper half of the figure illustrates the structuring characteristic to the 'Legislation-centric visualization', with requirement ***§X.X*** as its root element. The advantage of this visualization is that it allows a synthesized visual representation of the requirement. On the other hand, the 'Element-centric visualization' (shown in the bottom half of the figure) provides a holistic view of the safety requirements that bear on the ***C*** root element, along with the safety risk that they are meant to target.

Currently, the scope of this new graphic model is still being ascertained and its notation has not been finalized. Progress is being made through the implementation of this modeling technique in the assessment of the Very Light Jets (VLJ) [6] case study described in the following section. Furthermore, some security requirements have also been translated into this complementary notation in order to do an informal comparison of its "expressiveness" in this field.

4.2 A Future Case-study

The situation previously discussed in Section 3, where the safety requirements are no longer adequately enforced because their applicability criterion is no longer representative of the reality, is reappearing today. A case in point is that engine and material technologies have allowed the creation of high performance light jets. These jets, aptly named Very Light Jets, are capable of achieving the same flight performances (in terms of flight level and speed) as large commercial aircraft.

However, navigational equipments required for flights within controlled airspace are (until now) enforced based on the aircraft's design and physical characteristics

(See Figure 5). Because of this, the VLJs will be able to find themselves within the same flight bands as large commercial aircraft, with incompatible and rudimentary navigational equipment.

Of course, ultimately safety will take precedence. For this, the applicability criterion of navigational requirements will need to be amended. A shift from the current criteria is required; the aircraft's weight, engine type and passenger seating capacity can no longer be regarded as the main parameters for determining its legislative requirements. New criteria must be adopted, to effectively highlight that it is the aircraft's operating environment which is determinant for such equipments. Under such circumstances, a graphical model that is centered on the legislation's applicability criteria is very informative, and might prove valuable for undertaking such a comprehensive legislative enhancement.

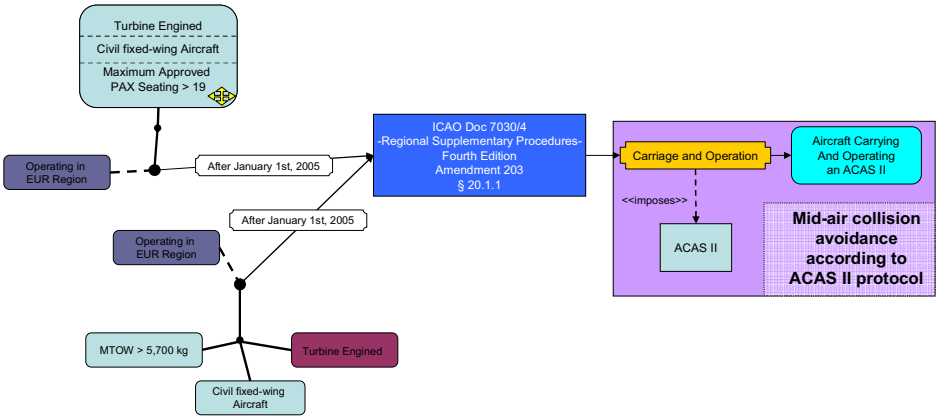

Fig. 5. ICAO's Regional Supplementary Procedures 7030/4 imposes requirements concerning the carriage and operation of the Airborne Collision Avoidance System (ACAS). The requirement (§20.1.1) states that *"[With effect from 1 January 2005] ACAS II shall be carried and operated in the EUR region by all…civil fixed-wing turbine-engined aircraft having a maximum take-off mass exceeding 5700 kg or a maximum approved passenger seating configuration of more than 19.*" Given these applicability criteria, the new VLJ aircraft would not be required to carry and operate an ACAS II. The diagram presented above is a visualization of this safety requirement. In it, the two discerning cases that are concerned with this requirement are shown. The aircraft's weight discriminant is presented in its 'component-representation' (i.e. each of the criterions is placed as an independent element), whereas the other is presented in its constituted version (i.e. as an element with fixed attribute values).

5 Conclusions and Perspectives

In this paper we argue about the creation of an interactive graphical model based on the safety legislations applicability criteria. The purpose of this graphical model is to extend the application domain of the EDEMOI methodology to include safety legislations (taking into account the specificities of these legislations and the concerns of civil aviation authorities). By itself, the extension which we propose follows a

branch of the original EDEMOI methodology, in which graphical models were used as tools in the analysis of security requirements (in contrast to their use as a stepping stone to the formal specification of the requirements [7]). Nevertheless, given the intrinsic nature of all 'directive statements', it is foreseeable that this extension will be equally useful in the analysis of security requirements.

Furthermore, the underlying principle of the EDEMOI methodology (i.e. the use of formal methods to specify and validate their embedded logic) is still valid for civil aviation safety requirements. However, the usefulness of this methodology will be limited to certain domains of safety legislation. Tentatively, to the domains whose legislation is not highly objective based. The reason for this is that, as was the case for security regulations, the interest of the formal model lies in its ability to be animated. Yet, highly objective based requirements impose abstract and/or unquantifiable targets that are incompatible with an insightful analysis through test-case animation. Moreover, given this abstract and/or unquantifiable nature, less of their important aspects can be viably formalized.

Some perspectives of this work include the complete analysis of the case-study discussed in Section 4.2, finalizing the notation and defining the scope of the modeling technique proposed in Section 4.

References

1. Degani, A.,: On the Typography of Flight-Deck Documentation. NASA Technical Memorandum #177605. Moffett Field (1992)
2. Laleau, R. et al.: Adopting a situational requirements engineering approach for the analysis of civil aviation security standards. J. Soft. Proc. Vol. 11. Issue 5, 487-503 (2006)
3. Lopez Ruiz, E.R.: Formal Specification of Security Regulations: The Modeling of European Civil Aviation Security. Master Thesis. Toulouse (2006)
4. Ledru, Y.: Using Jaza to animate RoZ specifications of UML class diagrams. In: 16th IEEE International Z User Meeting, IEEE Press, Columbia (2006)
5. Moore, G.S.: Notice of Proposed Rulemaking. 14 CFR Parts 40, 41, 42. Federal Aviation Administration. Washington, D.C. (1964)
6. United States Government Accountability Office (GAO) Report to Congressional Requesters.: Very Light Jets, Several Factors Could Influence Their Effect on the National Airspace. Washington D.C. (2007)
7. Dupuy, S., Ledru, Y., Chabre-Peccoud, M.: An overview of RoZ: A Tool for Integrating UML and Z. In: 12th Conference on Advanced Information Systems Engineering. Springer Press, (2000)

Checking compliance of a system with regulations : towards a formalisation

Laurence Cholvy and Claire Saurel

ONERA Centre de Toulouse
2 avenue Edouard Belin
31055 Toulouse, France
{cholvy,saurel}@cert.fr

Abstract. This paper addresses the problem of checking if an updated system is in compliance with the current regulations which apply on the domain.
We first present the applicative context in which this problem has been met. We sketch a formalisation of the problem of compliance and we show that is can be split in several sub-problems of different types, the solutions of which are discussed.

Keywords Compliance, regulations, formalisation.

1 The applicative context

This study is part of the ONERA program named IESTA[1], which aims at developping models, methods and a platform of simulation for analysing innovative concepts of Air Transportation Systems (ATS). This platform will allow its customers (air companies, aircraft manufacturers, official regulation providers, research laboratories and relevant ATS stakeholders) to virtually modify some parameters of the ATS (for instance modify landing procedure, modify types of airplanes, equipe aircrafts with new kinds of fuel...) and analyse the impacts of these scenarii on some environmental metrics.

In particular, this platform will allow its users to study the impacts of expected ATS modifications, on the noise level and chimical emission in the vicinity of a given airport. Modifications of interest will thus be the ones which, according to the simulation, lead to a reduction of noise or/and pollution levels.

In this context, we focus on the particular problem of the compliance of these modifications with the current regulations.

For instance, assume that the simulation shows that, for such a given aiport, modifying the landing procedure in such a given way leads to noise reduction. Before discussing the adoption of this interesting modification, it would be helpful to help users to check if it complies with the current regulations, or if it is incompatible with them or if it is not even ruled.

This is the problem we address.

More generally, the problem we are interested in can be described as follows:

[1] with financial support of DGA, FNADT, FEDER and Région Midi Pyrénées

- given a system composed of several components,
- given the set of regulations which rule this system,
- given a modification which is proposed for the system (modification concerning its structure or the way of performing its function),
- we first want to be able to check if the modified system is in compliance with regulations. If it is not, we want to help users to understand where are the causes of non compliance. Users will then have to revise the given regulations or to revise the proposed modification.

This paper is organized as follows. Section 2 tends to formalise the problem of checking compliance of a system with regulations. Section 3 analyses more deeply this problem and shows that it can be split in several sub-problems of different types. Section 4 focuses on the problem of providing the users an assistance to revise violated regulations. Section 5 mentions some relevant works, the scientific domains they belong to (Information Retrieval and Normative Reasoning) being candidate to provide us with solutions. Section 6 concludes this paper.

2 Towards a formalisation of the problem of checking compliance

The variables of the problem we address are the following.

Definition 1. *new denotes the updated system the compliance of which has to be checked.*

Definition 2. *KB denotes the background knowledge, i.e the knowledge about the considered environment.*

Example 1. In the ATS case, if the problem is to check the compliance of the ATS when aircrafts are given a new fuel, then KB may include characteristics and properties of this new fuel (density, volumic mass, inflammation temperature...), but also characteristics of airport environment (atmospheric pressure, physical models...). Any information describing the modified ATS when aircrafts are given a new fuel is in *new*.

¿From a more formal point of view, *new* and KB should be modelled in a common model. For instance, KB could be modelled by ontologies, hierarchy of concepts, dependance graphs between concepts, or more generally, and this will be supposed in the rest of the paper, by logical formulas. In the same way, *new* could be modelled by sets of nodes in the ontologies, sets of concepts, or more generally, and this will be supposed in the rest of the paper, by logical formulas.

Assumption In the following, we will suppose that *new* is compatible with KB i.e, $KB \cup new$ will be supposed to be a consistent set of formulas.

Consistency is a prerequisite to the problem of compliance since, if $KB \cup new$ is inconsistant, this means that new contradicts the domain knowldege, thus, building new is impossible. Consequently, the question of checking his compliance is not posed.

Definition 3. *A regulation r is a triplet: $r = < st_r, ref_r, norms_r, def_r >$, where*

- *st_r is an ordonned list of the levels which structure the text of r in an arborescent way.*
- *ref_r denotes the set of other regulations r refers to.*
- *$norms_r$ denotes the normative contents of r. If n is in $norms_r$, n is assigned a position label related to st_r which denotes its position in the arborescent structure of r.*
- *def_r is a set of the definitions of the concepts which are used in the regulation text.*

Note that several members of $norms_r$ may have the same position label. It means that they are in the same text unit in r.

What we want here to capture is that a regulation contains information of very different natures :

- conceptual definition information (def_r) : that is an optional part of the regulation. Concepts which are ruled by r are defined in def_r.
- rules ($norms_r$) : that is the core of the regulation. The rules apply on the real world by stating what is obligatory, permitted or forbidden under which conditions. Formal modelling rules requires the use of a logical formalism dedicated with normative reasoning i.e a deontic logic (see section 5). In the following, we will thus suppose that these rules are modelled by formulas of such a logic.
- In a regulation r, conceptual definition and rules are expressed according a given structure (st_r).
- information about other regulations ref_r : regulations which inspire the regulation, regulations which are abrogated by the regulation...One generally mainly find it in the head of the text of r.

Example 2. For r being [2], $st_r = [article, alinea]$. That means that r is composed of several articles, each of them being eventually composed of alineas.

Example 3. Consider now r_0 a regulation such that $st_{r_0} = [part, subpart, article]$. If the formula n is in the 2d article of the 3rd subpart of the 1st part of r_0, then the position label assigned to n is $[(part, 1), (subpart, 3), (article, 2)]$.

Example 4. Finally, for r being [2], ref_r includes regulations [16] and [3].

Definition 4. *$R = \{r_1, ...r_n\}$ denotes the set of all the regulations which apply on the domain.*

Example 5. In the ATS case, the set R of regulations which rule the aeronautic domain is composed of CEE regulations, national regulations such as environment code, civil aviation code, and lots of orders and procedures.

Several relations exist in R such as:

- $r_2 \geq_S r_1$ is true if regulation r_1 is a specialization of regulation r_2. $\geq_S$ is a partial pre-order defined upon R.
- $r_2 \geq_A r_1$ is true if regulation r_2 abrogates regulation r_1 : it means that r_1 doesn't apply anymore since r_2 applies. $\geq_A$ is a partial pre-order defined upon R.
- a binary relation *replace* so that $replace(a_{i_j}, a_{k_l})$ is true if the i^{th} article of regulation r_j replaces the k^{th} article of regulation r_l.

Some of relations of this type has been presented in [4]. Our model represents a slightly simplified form of real regulations, since as for instance, $\geq_A$ could be defined between text units of regulations, and not only between regulations (as for the relation *replace*) . We here suppose that instances of such relations concerning a regulation r are explicited in ref_r.

Example 6. Let r_1 and r_2 respectively being regulations [2] and [16]. $r_2 \geq_S r_1$, because the regulation dealing with rules concerning the Blagnac airport specializes the regulation about french public air transport.

Definition 5. *If $R = \{r_1, ..., r_n\}$ is the set of regulations which apply on the domain, we define $norms_R$, as the set of all the rules of all the regulations of R, i.e, $norms_R = \cup_{i=1}^{n} norms_{r_i}$*

$norms_R$ is thus a set of formulas of a particular deontic logic.

More formally, we assume that a formal model (formal language and formal inference denoted $\models$ in the following) has been chosen for modelling and reason with rules of $norm_R$, background knowledge KB and modification new[2].

In the rest of the paper, we will suppose that $norm_R$ is consistent i.e is a consistent set of rules.[3]

[2] Ideally, this formal model is a logic which allows to express and reason with any type of deontic notions which appear in regulations, any type of knowledge, causal or temporal, which appear in KB. Such a logic should then be a deontic logic ([5], [8]) allowing to reason with causality and time as well. Defining such a general logic remains to be done.

[3] Notice that consistency of sets of rules has been defined in [6] so that, $norm_R$ is a consistent set of rules if there is no situation (or state of the world) s, consistent with KB (i.e possible) such that : $s \cup norms_R \models false$. This general definition is not taken here for simplicity, but notice that if $norm_R$ is a consistent set of rules according to this definition, then $norm_R$ is a consistent set of rules

Checking compliance is then defined by checking one of the following assertions:

1. "case of permitted modification"

$$\forall \phi \quad KB \cup new \models \phi \quad \Longrightarrow \quad \models norms_R \rightarrow permitted(\phi)$$

   This expresses that all the consequences the system modification *new* (under context KB) are explicitely permitted by the rules in the regulations. In the first case, *new* could be accepted without any other modification since it is compliant with the regulations.

2. "case of forbidden modification"

$$\exists \phi \quad \exists r \in R \quad KB \cup new \models \phi \quad and \quad \models norms_r \rightarrow forbidden(\phi)$$

   This expresses that the system modification, *new*, has some consequences (under context KB) which are explicitely forbidden by one regulation In this case, *new* cannot be taken into account since it explicitely leads to violate regulations, unless modifying regulations themselves. Localizing the very rules which are violated by *new* is addressed in section 4.

   Notice that in the general case, the prohibition is not caused by only one regulation. So the case of forbidden modification should be described by:

$$\exists \phi \quad KB \cup new \models \phi \quad and \quad \models norms_R \rightarrow forbidden(\phi)$$

   However, in this paper, we assume that the prohibition is caused by a single regulation because it simplifies the presentation of localizing violated rules (see section 4).

3. "case of a non ruled modification"

$$\exists \phi \quad KB \cup new \models \phi \quad and$$

$$\not\models norms_R \rightarrow permitted(\phi) \quad and \quad \not\models norms_R \rightarrow forbidden(\phi)$$

   This expresses that the system modification, *new*, has some consequences (under context KB) which are neither explicitely permitted nor explicitely forbidden by the regulations. In this case, it will be possible to accept *new* only after an analyse and modifications of regulations so that consequences of *new* are permitted.

By definition, these three assertions are exhaustive. Furthermore, they are exclusive only if $norm_R$ is consistent. This is the reason why assuming consistency of rules is a prerequisite to the definition of compliance.

3 Decomposing the problem of checking compliance

Checking compliance can be decomposed into several sub-problems. The idea is to check compliance only on a subset of R, made of regulations which "apply at present" and "concerned by *new*". At this step, the property "being a regulation concerned by *new*" remains to be formally defined. This property could be defined so that the test of checking compliance is more efficient in time. It could also be defined so that we can help the user (in the second case) to find the precise articles of the regulations that are violated.

- **Problem pb 1 : find the "regulations which could be violated"**
  This problem can be divised into two sub-problems as follows:

  - **Problem pb 1.1 : find the "regulations which apply at present"**
    This problem consists in selecting the regulations which are not abrogated nor replaced by other regulations. In other words, the problem is to focus only on the regulations that apply at the moment.
    This problem may be defined by : find $max_{\geq_A}(R)$ [4]
    Information in $\cup_{i=1}^{n} ref_{r_i}$ will be hepful to solve this sub-problem .

  - **Problem pb 1.2 : find the "regulations concerned by** *new*"
    This problem is a problem of Information Retrieval, the information to be retrieved being regulations.
    In order to solve it, considering information in $\cup_{i=1}^{n} def_{r_i}$ (i.e definitions of the concepts used in the regulation text) will be necessary.

  The two above sub-problems may be solved in any sequence order : each of them contributes towards reducing the set of regulations to be considered in checking compliance.

  Let us denote R_{new} the set of the regulations of R which apply at present and which are concerned by *new*.

- **Problem pb 2 : checking compliance of** *new* **with** R_{new}

  This comes to check the three assertions:

$$\forall \phi \quad KB \cup new \models \phi \quad \Longrightarrow \quad \models norms_{R_{new}} \rightarrow permitted(\phi)$$

$$\exists \phi \quad \exists r \in R_{new} \quad KB \cup new \models \phi \quad and \quad \models norms_r \rightarrow forbidden(\phi)$$

$$\exists \phi \;\; KB \cup new \models \phi \;\; and \;\; \not\models norms_{R_{new}} \rightarrow permitted(\phi) \;\; and \;\; \not\models norms_{R_{new}} \rightarrow forbidden(\phi)$$

[4] If $\geq$ is a partial pre-order defined on R, then $max_{\geq}(R)$ is defined by:
$max_{\geq}(R) = \{r \in R : \forall r' \in R, r' \geq r \Rightarrow r \geq r'\}$

4 Towards assisting localization of violated rules

In this section, we suppose the case when *new* doesn't comply with R_{new}. I.e, we assume that the second assertion of problem pb 2 is true.

Definition 6.

$$R_{Forbidden} = \{r \in R_{new} \quad , \exists \phi \quad KB \cup new \models \phi \quad and \quad \models norms_r \rightarrow forbidden(\phi)\}$$

$R_{Forbidden}$ denotes the set of regulations which are involved in the cause of non compliance of *new* with R_{new} under KB.

In order to assist users in revising such regulations, several kinds of aids may be proposed to him. Below we sketch some induced problems.

- **Problem pb 3 : localize a cause of non compliance in a regulation**
  Let $r \in R_{Forbidden}$. This problem consists in:

  1. exhibiting the elements in $norms_r$ which are involved in a demonstration of "forbidden modification".
  2. finding their position label in r (according to st_r, as defined in definition 3).

- **Problem pb 4 : localize the least specialized regulations involved in non compliance**
  The problem is to identify the uppest regulations involved in non compliance, towards the specialization relation defined on R : in other words, it is to identify sources (in the specialization or hierarchical sense) of non compliance.

  Formally : find $max_{\geq_S}(R_{Forbidden})$

- **Problem pb 5 : propagate a cause of non compliance in a set of regulations**
  The problem is, given a regulation involved in non compliance, to identify all the regulations which specialize it. These regulations, because they take their inspiration from the violated regulations, are are also involved as causes of non compliance.

  Formally : let $r \in R_{Forbidden}$, find $\{r' \in R_{Forbidden}, r \geq_S r'\}$.

- **Problem pb 6 : explanation for non compliance** The problem is to give an informative explanation based upon non compliance demonstrations.
  This comes to a problem of Explanation Generation, which has been studied for many years [20], [1].

5 Relevant works

Among the different sub-problems we have raised in the previous sections, two of them are of particular interest. More specifically, these are: a problem of information retrieval, the information to be retrieved are regulations (cf problem 1.2) and a problem of normative reasoning (cf problem 2). These two very different questions have been addressed by many works we mention some of them below.

5.1 Information retrieval, regulation retrieval

Information Retrieval is a vast domain of research whose works aim to define models and methods or algorithms, to retrieve information among a large set of information, like the web space. See [22] for an interesting overview. The user's demand is formalised by a query Q (of the form "retrieve documents which contains terms $t_1...t_n$").

The three most used models in Information Retrieval are the vector space model, the probabilistic model and the inference network model.

According to the vector space model, the user query as well as the documents the query is addressed to, are represented by vectors of terms (words of a given vocabulary for instance). The score of a document is defined as a similarity degree between its vector and the query vector. Several similarity degrees are usually used, among which the dot product defined by: if $\boldsymbol{D}$ denotes the document vector and $\boldsymbol{Q}$ denotes the query vector, then the score of D for Q is:

$$sim(D, Q) = \sum_{i=1}^{n} d_i.q_i$$

where d_i is the value of the i^{th} component of (D) and q_i is the value of the i^{th} component of (Q). The value d_i (resp q_i) is called the weight of the di^{th} term in the document (resp, query).

Various methods for weighting terms have been defined. All of them are based on different parameters which are : term frequency (words that repeat several times in a text are considered salient), document frequency (words that appear in many documents are considered common and are not very indicative of document content), the number of documents that contain a given term, the document lenght (in bytes), the average document length (in bytes)...

As for Probabilistic models, they assume that documents in a collection should be ranked by decreasing probability of their relevance to a query (*probabilistic ranking principle*). Since knowing its true value is impossible, the probability of relevance of a document to a query has to be estimated. In this family, the models differ from the way they estimate that probability of relevance.

Last models are Inference network models. In these models, document retrieval is modeled as an inference process in an inference network.

Since we consider Regulation Retrieval as a particular case of Information Retrieval, solving pb 1.2 could be done by adapting a model of Information Retrieval. For doing so, the definitions of concepts used in a regulation (i.e the

def_r part) has obviously an important role to play in the process. Furthermore, the very structure of regulations (i.e the st_r part) is also something particular which must be taken into account by Regulation Retrieval models to be defined ([4], [9]).

Let us also mention [10], in which the authors define a tool to analyse a regulation and extract rights and duties expressed in the regulation. Legal texts are annotated in order to identify the agents, their rights (actions that the agents have the permission to perform under come conditions) and duties (the actions they have to perform under some conditions)... A semantic model in then built from these annotations.

Let us finally cite [13], in which the author defines a legal ontology of the french Law. Such an ontology could be used as a common concept description language and links with the def_r part of regulations should be establised.

5.2 Reasoning with regulations, normative reasoning

Problem pb 2 raises the question of checking if a given formula, (here $permitted(\phi)$ or $forbidden(\phi)$), is implied by some rules (here $norms_{R_{new}}$). This is a particular case of what is called "normative reasoning" i.e, reasoning with norms.

Reasoning with norms requires at least modelling deontic notions (permission, prohibition, obligation...). But it also requires modelling individuals (agents on which obligations, permission and prohibition apply) and properties on individuals. It sometimes also require modelling several dimensions of time (time of validity of norms, deadlines...) and different types of norms (defeasible norms, Contrary-to-duties...). To our knowledge, there is no general logical formalism which allow to reason with so many different notions. However, there are several kinds of formalisms which allow to model some aspects of the norms. These are deontic logics [8].

Most of deontic logics are modal ones, [5], since deontic operators are not very-functional operators (for instance, it may be the case that smoking is forbidden, even if somebody is smoking). Some of them are based on dynamic logics [15], or based on temporal logics, or both [7]. They also may be non monotonic [11], [23].

However, First Order Logic (FOL) can also been used to reason with deontic notions ([21], [14], [17], [19], [12]...) and is a compromise between expressivity and simplicity. In this case, normative reasoning comes to a problem of theorem proving in FOL which is solved (at least from a theoretic pioint of view) by different means: provers based on Resolution Rule, tableaux methods, or any method defined for the SAT problem.

Let us finally mention a very theoretical but interesting work, [18], in which the authors define a dynamic deontic logic for reasoning with consequences on permissions and prohibitions, that the modification of a policy generates. This aims at helping the user who wants to modify a regulation, by allowing him/her to derive the permissions which were valid before the modification and which are no more valid after; or the permissions which become valid after the modification etc.

6 Conclusion

In this paper we have addressed the problem of checking compliance of an updated system with regulations. The main contributions are a formalisation of this problem and its decomposition in several sub-problems of different types. We also sketched some functionalities which could help a user to revise regulations in case of non compliance. We finally quickly presented relevant litterature, more precisely Information retrieval and Normative Reasoning, which could offer solutions.

Notice however that this work is very preliminary and thus raises many open questions, the most important one being the definitions of the solutions of the different sub-problems and their applicability as well. The case named "case of a non ruled modification" has also to be studied. And the model of regulations used in this work, has to be refined in order to take into account a finer granularity of representation. Indeed, for instance, this model does not allow to represent relations between text units .

However, even preliminary, this work enlights the complexity of the problem of checking compliance and the varieties of questions to be solved.

Acknowledgments. We would like to thank the anonymous reviewers for their comments which helped us to improve the paper.

References

1. F. Benamara et P. Saint Dizier. WEBCOOP: A Cooperative Question-Answering System on the Web. 10th European Chapter of the Association of Computational Linguistics (EACL), Budapest, Hongrie, avril 2003.
2. Arrêté du 21 mars 2003 portant restriction d'exploitation de l'aérodrome de Toulouse-Blagnac (Haute-Garonne), NOR: EQUA0300268A
3. Réglement (CEE) n 2408/92 du Conseil, du 23 juillet 1992, concernant l'accès des transporteurs aériens communautaires aux liaisons aériennes intracommunautaires
4. B. Chabbat. Modélisation multiparadigme de textes réglementaires. Thèse soutenue le 8 décembre 1997, à l'INSA de Lyon.
5. B. F. Chellas Modal logic, an introduction. Cambridge University Press, 1980.
6. L. Cholvy Checking regulation consistency by using SOL-deduction Proc. of AI and Law conference, Oslo, Norway, june 1999.
7. F. Dignum and R. Kuiper. Combining dynamic deontic logic and temporal logic for the specification of deadlines. In Jr. R. Sprague, editor, Proc. of thirtieth HICSS, Wailea, Hawaii, 1997.
8. Proceedings of the DEON workshops. http://deon2008.uni.lu/publications.html
9. D. Jouve Modélisation sémantique de la réglementation Thèse de l4INSA de Lyon, novembre 2003
10. N. Kiyavitskaya et al. Extracting Rights and Obligations from regulations: Toward a Tool-supported Process 22^{nd} IEE/ACM Int. Conf. on Automated Software Engineering (ASE'07), Atlanta, november 2007.
11. J. Horthy. Deontic Logic as Founded on Nonmonotonic Logic Annals of Mathematics and Artificial Intelligence, 1993.

12. J.Y. Halpern, V. Weissman. Using First-Order Logic to Reason about Policies Proc. of the 6^{th} IEEE Computer Security Foundations Workshop CSFW-03, 2003.
13. Guiraude Lame. Construction d'ontologies partir de textes.
Une ontologie du droit dédiée à la rechecrhe d'information sur le web.
14. R. Lee. Bureaucracies as deontic systems *ACM Transactions on Information Systems (TOIS)* 6(2), pp 87 - 108 , April 1988.
15. J.-J.Ch. Meyer. A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. In Notre Dame Journal of Formal Logic, vol.29, 1988.
16. Arrêté du 12 mai 1997 relatif aux conditions techniques d'exploitation d'avions par une entreprise de transport aérien public (OPS1)- NOR: EQUA9700893A
17. Ong, R. Lee Detecting deontic dilemmas in bureaucratic rules: a first-order implementation using abduction Proc of *Proceedings of the second Workshop on Deontic Logic and Computer Science (DEON'94)*, Oslo, 1994.
18. R. Pucella, V. Weissman. Reasoning about Dynamic Policies. In proc. of the 6^{th} Int. Conf. on Foundations of Software Science and Computation Structure (FOSSACS 04), 2004.
19. U. Ryu, R.M. Lee. Defeasible Deontic Reasoning and Its Applications to Normative Systems. *Decision Support Systems*, 14(1), pp. 59-73, 1995.
20. Cl. Saurel. Méthode de génération d'explications négatives à partir d'une base de connaissances en logique des prédicats 8èmes journées internationales sur les systèmes experts et leurs applications, Avignon, 1988.
21. M.J. Sergot. Prospects for representing the law as logic programs. . In *K.L. Clark and S.. Tarnlund, editors, Logic Programming* , pp 33–42. .Academic Press, London, 1982.
22. A. Singhal. Modern Information retrieval: a brief overview IEEE Data Engineering Bulletin 24(4), 35-43, 2001.
23. L. van der Torre. Violated obligations in a defeasible deontic logic. Proc of ECAI 1994.