Dissertation, genehmigt von der Fakultät für Wirtschaftswissenschaften der Universität Fridericiana zu Karlsruhe, 2008

Model-Driven Management of Internal Controls for Business Process Compliance

Dipl.-Inform. Kioumars Namiri

Referent: Prof. Dr. Rudi Studer
Koreferent: Prof. Dr. Günter Müller

2008 Karlsruhe

Zusammenfassung

Internationale, EU-weite und nationale Gesetze fordern die Realisierung von effektiven *internen Kontrollsystemen* in Unternehmen. Eines der Hauptziele der *internen Kontrollen* ist die Vermeidung von Risiken, die aus den operativen Geschäftsprozessen eines Unternehmens erwachsen, um u.a. vertrauenswürdige Bilanzabschlüsse zu erreichen.

In dieser Dissertation werden Vorgehensweisen basierend auf Modellen von internen Kontrollen und Geschäftsprozessen vorgestellt, die zu einer Reduzierung der Aufwände zur Sicherstellung der Geschäftsprozesscompliance beitragen. Dies wird durch eine höhere Anpassbarkeit, Wiederverwendbarkeit und Benutzbarkeit von internen Kontrollen erreicht. Mit Anpassbarkeit wird das Bedürfnis adressiert, schnell und unkompliziert neue bzw. modifizierte Kontrollen für Geschäftsprozesse zu modellieren. Wiederverwendbarkeit adressiert die Möglichkeit, die Kontrollen auf einem hohen Abstraktionslevel zu beschreiben, damit sie in verschiedenen fachlichen Kontexten wieder verwendet werden können. Benutzbarkeit verlangt ein minimales technisches Know-How für die Modellierung und Anwendung der internen Kontrollen in Geschäftsprozessen.

Es wird eine Abstraktionsschicht über die Geschäftsprozesse und ihren Managementaktivitäten beschrieben, in der die notwendigen Kontrollen formal modelliert und gegen die laufenden Ausführungsinstanzen eines Prozesses überprüft werden. Der Ansatz dieser Dissertation ist regelbasiert und modell-getriebenen, der auf der konzeptionellen Trennung des Entwurfs von internen Kontrollen und Geschäftsprozessen basiert. Die Wirksamkeit der Kontrollen wird dann durch eine enge Integration von internen Kontrollen und Geschäftsprozessen zu ihrer Ausführungszeit erreicht. Der Benutzbarkeit des Modellierungsansatzes wird durch eine musterbasierte Vorgehensweise unterstützt.

Abstract

This thesis tackles the problem of high costs and effort for achieving the compliance of business processes to regulations in the area of Enterprise Risk Management (ERM). Common to these regulations are requirements on the presence of effective internal controls in companies. The current shortcomings faced by companies in this respect are the low level of automation with regard to translating compliance requirements into a set of internal controls and assuring the effectiveness of these controls during the execution of business processes. The high cost of business process compliance is due to the fact that in many organizations a large number of the steps in designing and testing controls on business processes are still manually executed.

In order to overcome the above challenges this thesis develops an abstraction layer above business processes. This layer is responsible for ensuring business process compliance. In this layer the controls are formally modeled and evaluated against existing process models and their execution instances. The thesis describes a novel, model-driven approach for the automation of business process compliance through monitoring the effectiveness of controls. This is enabled through the conceptual separation of the design of controls and business processes at a model-level, and a tight integration of controls in the business process instances at the execution-level. In order to address the usability of the models and the approach, this thesis advocates the use of control patterns in the abstraction layer responsible for business process compliance. The control patterns should give compliance experts and business process experts the ability to specify and design their compliance requirements accordingly. These control patterns are then mapped to formal models that are used by technical experts to implement the control patterns in business processes.

To complement this abstraction layer which uses models of the entities involved in business process compliance, a verification and validation approach is presented: The verification of business process models assures that business processes be built in a compliant manner as required in a formal specification. The validation assures the compliant behavior of business process executions, i.e. the business processes work as described in the formal model of controls.

Acknowledgements

Many people contributed to the work presented in this thesis. First of all, I would like to thank Prof. Dr. Rudi Studer for giving me the opportunity to do this research and for his support throughout this thesis.

In addition, I would like to thank my co-advisor, Prof. Dr. Günther Müller at University Freiburg. In addition to his precious scientific advice, he gave me great self-confidence in my work.

I am especially grateful to my supervisor at FZI, Nenad Stojanovic, who greatly encouraged my work and gave me very helpful support throughout my completion of this thesis.

The fruitful discussions with Shazia Sadiq from the University of Queensland during her visit at SAP Research in Karlsruhe sharpened many of my ideas. From SAP Research in Karlsruhe, I would like to especially thank Elmar Dorner, who told me best which steps to take in order to complete a PhD thesis. From SAP Research in Karlsruhe, I would like to further thank Rainer Ruggaber, Ulrike Greiner, Orestis Terzidis, Michael Altenhofen, Daniel Oberle, Norman May, Susan Marie Thomas, Zoltan Nochta, Jens Lemcke, Christian Drumm, Matthias Born and Olaf Grebner for their scientific support. From the SAP team in St. Leon Rot I would like to thank Günther Liebich, who gave me the opportunity to verify my research results in the context of SAP Financials product lines. The master students Maurice Kügler, Javit Gellaw and Rene Fitterer at SAP Research contributed to my work, especially during the implementations in the context of ATHENA and ICCOMP projects.

I am much obliged to Souni Breil, Susan Marie Thomas and Asma Alazeib, who proof-read the English of my thesis.

Above all, I thank my family and friends. I could have not completed this thesis without all their encouragement and confidence.

Kioumars Namiri

Table of Contents

List of Figures

1 Introduction

1.1 Motivation

Compliance has become a major topic in today's business world. The term compliance became popular in recent years with the advent of regulations such as the Sarbanes Oxley Act (SOX) [SOX02] filed in 2002. The SOX came as a response to a number of incidents of corporate accounting fraud and theft of consumers' personal data. Companies within the jurisdiction of the SOX must frequently adapt their operations to relevant regulations and periodically demonstrate compliance by submitting reports to audits. Non-compliance to current regulations such as the SOX can lead to large penalties (e.g. increased fines and the possibility of imprisonment), which has significantly increased the expected cost of non-compliance. It may be for this reason that Forrester Research projects that the compliance software market will expand to $1.3 billion by 2011 [Rasmussen, 2006].

Gartner Research gives a general definition of the term compliance [Bace et al., 2006]: "Compliance is the process of adherence to policies and decisions". Gartner further categorizes different types of compliance requirements according to their sources. These sources are regulatory compliance, commercial compliance or organizational compliance. Regulatory compliance is concerned with laws that a business must obey, or else risks legal sanctions up to and including prison for its officers. Commercial compliance requires that a company adheres to a set of rules and policies in the course of its business with trading partners and customers. Organizational compliance deals with the creation of internal compliance standards within individual companies.

The source of the regulatory compliance requirements with which this Research is concerned is Enterprise Risk Management (ERM). The Committee of Sponsoring Organizations (COSO) defines ERM in [COSO-ERM04] as "the methods and processes used by organizations to manage risks (or seize opportunities) related to the achievement of their objectives". Regulations such as the SOX and Basel II [BaselII08] fall into this category. It is common practice in the area of ERM regulations to document and implement effective **internal controls** in the company. The SOX goes so far as to declare it a management responsibility. Internal controls is defined by COSO in [COSO92] as a "process designed to provide reasonable assurance regarding the achievement of objectives in effectiveness and efficiency of operations, reliability of financial reporting, and compliance with applicable laws and regulations".

The realization and effectiveness of the internal controls involves different areas of company structure: Management, internal auditing consultants and compliance experts, external regulation bodies, business process experts (including system developers and technical consultants) and employees. Each of these brings a different view to, and plays a different role in the enterprise, uses different terminology when speaking about the same domain and requires need-specific system support. This is one of the main reasons why the introduction and operations and in fact every aspect of internal controls compliance (e.g. SOX 404) is considered to be expensive and time consuming [Hartman, 2005].

Gartner states in [Bace et al., 2006] that "Most companies realize that compliance is an ongoing process, not a project." In order to remain compliant in a sustainable manner, Forrester recommends involving the following technical areas of a company into the compliance approach [Rasmussen, 2006]:

- **Enterprise Content Management** (ECM): A company must be able to categorize, store, retain, and manage access to its sensitive information. In order to fulfill growing reporting requirements (e.g. SOX section 409), companies must have access to the right information rapidly, irrespective of location and format: important data is not always stored in structured formats such as databases. Most companies are drowning in unstructured data such as e-mails, word documents and excel files.
- **Business Process Management** (BPM): BPM is a key technology to compliance from two perspectives:
  i) it helps companies automate, manage and formalize the review and sign-off of their business processes
  ii) it provides a platform for collaboration between the key players involved in compliance in order to automate the risk and compliance definition processes.
- **Enterprise Applications** (EA): Business processes rest on the functionality provided by enterprise applications (such as HR systems, accounting systems etc.).
- **Business Intelligence** (BI): BI and business analytics provide various views on data in ECM as well as on the work achieved by BPM and EA.

In this thesis we focus on the relationship between business process management (BPM) and internal controls. The relationship can be described by the following process [PCAOB04]:

> Identify all the **significant accounts** in the balance sheet of the company. Identify all **relevant business processes** that affect those accounts. Define one set of **control objectives** for each relevant business process. These control objectives are specific to the enterprise and must hold true for that particular process. Continuously assess **risks** for the enterprise by identifying them for each relevant business process. Design and implement a set of effective **controls** in order to prevent or detect the occurrence of identified risks. The design of controls must be tested and it must be shown that they are used in **daily operations as designed**.

The application of the relationship between BPM and internal controls as presented in the process description above is called **business process compliance** in this thesis.

Ensuring the effectiveness of controls in business process compliance in practice today has a manual nature. This is because of *retrospective reporting nature* of compliance, wherein traditional audits are conducted for "after-the-fact" detection of possible control violations, often through manual checks conducted by consultants. The ability to increase automation in business process compliance grants a *preventive* nature to compliance by detecting possible control violations in advance during business process executions. This could lead to significant time savings in the design and achievement of business process compliance and therefore to cost reductions.

The aim of this thesis is to provide models and methods which will make it possible to achieve a higher level of automation in business process compliance than that which exists in today's business world. Automation is achieved in such a way that compliance has a preventive nature, i.e. control violations during execution of business processes will be preemptively detected.

1.2 The Challenge of Business Process Compliance for Standard Software Providers and their Customers

Today, most business enterprises do not implement their business processes from scratch. Instead, they decide to buy pre-built software-solutions from software vendors, where the business processes are built on top of it. This is especially the case in the area of ERP software. We call the providers of these kinds of ERP-software in this thesis Standard Software Providers.

The software provided by Standard Software Providers has to be adapted within the customer companies in order to ensure that the implemented business processes meet the needs and requirements of the customer. Usually, the same business process types are adapted differently from company to company in order to funtion within different and changing business environments. Changing business environments are caused by frequently changing business practices, by the capabilities of an enterprise and by its partner ecosystem. Companies have to configure the functions in the purchased software solutions accordingly. This practice is known as Customizing or *Business Configuration*. Business configuration is part of every implementation project for customers who have bought standard software. In [Soffer et al., 2003] configuration is described as an alignment process of adapting the enterprise IT-system to the needs of an enterprise.

The requirements dictated by regulations in the area of ERM add another dimension of complexity to business configuration: the customer must design the internal controls appropriately during business configuration to assure their effectiveness during daily operations. An approach which brings a higher level of **adaptability**, **reusability**, and **usability** to the internal controls compliance process is required. Adaptability is defined as an easy and fast way to the introduction of new or changed controls on business processes. Reusability refers to the possibility of describing the controls on a conceptual level in order to abstract them from the concrete implementation details of specific controls. Usability addresses the need to bridge the gap between the compliance experts and IT experts.

A further challenge of internal controls compliance is that it must consider business process executions in addition to business process designs. Actually, in the context of regulatory requirements such as the SOX, the law requires that the internal controls on different entities in enterprises be effectively applied during business process executions. This basically means that enterprises have to prove that their processes and internal controls **work as planned** in daily operations. Compliance requirements are tested and certified by external auditors as late as possible in the management life cycle of a business process, during the runtime phase of the business process - by checking the system logs or checking the business documents that reflect a financial transaction (such as a purchasing order). The reason that this approach is selected by external auditors is that, although state of the art ERP and financial systems contain different predefined control options, in many situations business level requirements (such as efficiency and fast business transaction response time) render enterprises unable to activate and configure the controls (they may even disable an already set control option). A change in the configuration may result in a company not being notified about the possible violation of a defined internal control, since the ERP systems do not proactively notify the compliance experts about the changes in their business configurations. The enterprise runs the risk of becoming non-compliant. Thus an effective business process compliance approach must consider the execution of business processes in addition to the the design of controls in business processes.

In order to overcome the above challenges, SAP conducted a project called ICCOMP (Internal Controls Compliance). Parts of the research which appears in this thesis were a product of this project. The context of the project was to address the aforementioned challenges in the context of Enterprise SOA (E-SOA) [Woods et al., 2006]. E-SOA acts as a blueprint for adaptable software architecture for developing service-oriented, enterprise-scale business solutions. The core of E-SOA is the notion of an enterprise service. An enterprise service captures business logic that can be accessed and repeatedly used by the customer to support a particular business process. Aggregating enterprise services into end-to-end business processes should provide the foundation for the task of automating enterprise-scale business scenarios.

The goal of the ICCOMP-project is to provide input for a future model-based architecture:

- to enable automated verification and monitoring of the effectiveness of business process's compliance according to the internal controls
- to perform the internal controls automatically, based on the current state of parameters (instances) of a business process
- to enable non-technically oriented auditing experts, through the abstraction layer introduced on the top of the compliances definition to build compliance based on the domain model that has to be provided.

1.3 Research Questions and Contributions

The approach selected for achieving the ICCOMP-project goals was based on the following three research questions:

1. What are the relationships between business processes (design and execution time) and internal controls?
2. Can internal controls be automated using a model-driven approach?
3. Is the usability given for the compliance experts using the provided models and approaches?

In order to answer these questions, this thesis imposes an abstraction layer above a business process, in which the controls are formally modeled and evaluated against existing process models and instances. It describes a novel, model-driven approach for the automation of internal controls in an enterprise, based on their conceptual separation from business process management (BPM). The approach advocates the use of an empirically determined set of control patterns in the proposed abstraction layer. Here are the main contributions of this thesis:

- **Modeling the intersection between business processes and internal controls**

In order to capture the relationship between internal controls and business processes in models, we introduce the notion of *controlled entities* in business processes. These are modeling entities that are subjected to business process compliance. In order to do this, we develop a precise model of business processes for achieving business process compliance and formally model the role of the controlled entities in a business process. The model and controlled entities and their relationships to business processes are designed in such a way that they are *reusable* and *adaptable*, which were two key challenges for customers building their business processes on top of products offered by standard software providers.

- **Identification and application of control patterns in business processes**

The roles involved in defining business process compliance have to be assigned with a *usable* access to developed models in order to define and deploy the necessary controls on business processes. We have empirically identified a set of frequently defined patterns of controls on business processes at different enterprises. These patterns provide the basis for the terminology in which the compliance experts communicate about the business process compliance domain. A formal model of these patterns is developed and their relationship to the controlled entities is captured. This way a compliance expert can comfortably define controls in a pattern-based manner without necessitating any detailed understanding of the models provided in the first contribution, which satisfies the requirement of *usability* of a model-driven approach for business process compliance.

- **Preventive nature of business process compliance in daily operations**

A strict model-driven design of controls simplifies achieving business process compliance because it provides better support for realizing compliance in a preventive manner. The conditions that represent a control violation can be formally captured in controls by defining their relationships to controlled entities in a business process. The approach provided by this thesis detects a control violation during business process executions, even if a business process expert/technical consultant removes the control from the process, because he or she is unaware of the necessity of that control. The described approach (building on top of the models of the controls and the controlled entities) enables dynamical application of the controls during the execution phase of a business process. This is possible because there is a minimum overlap between business process design and compliance design, which supports the requirement of *reusability* and *adaptability* of the entities involved in business process compliance.

1.4 Readers Guide

The stucture of the thesis (see Figure 1):

Chapter 2 (Scenario) sets out the challenges of business process compliance in the case of a purchasing process. We describe how two different customers derive, define and realize their internal controls requirements on the purchasing process provided by a standard software provider. Based on the scenario description, a set of requirements for business process compliance are elicited and discussed.

Chapter 3 (Basic Concepts) introduces the concept of internal controls compliance and business process management. Another core objective of this chapter is to analyze several regulatory compliance requirements in the area of ERM and to work out their relationship to internal controls. This way a method is developed to build a holistic view on compliance achievement for regulations in the area of ERM by providing novel solutions to their overlapping parts. It is in this chapter that the relationship between business process management and internal controls is established.

Chapter 4 (Domain Model for Business Process Compliance) formally describes the model of business process compliance by identifying the first class entities involved in its definition. Based on the relationship between internal controls and business process management established in the previous chapter, the “verification and validation” approach is proposed for the domain of business process compliance. Verification aims to assure the correct business level design of business process models, whereas the aim of validation is to ensure that business processes work as required by controls during their executions. After this, we concentrate on developing the

model of controlled entities in a business process by providing a model of business processes and the relationship between controlled entities in this model. These modeling entities serve as the underlying conceptual framework for the design of controls in business processes, the verification of business process models and the validation of the compliant executions of business processes in later chapters.

Chapter 5 (Business Process Verification) covers the verification-part of the "verification and validation"- approach that was proposed in the previous chapter. It describes a novel verification approach to business process models. A set of business level constraints using the semantic web rule language (SWRL) are then built on top of the ontological representation of business process models in OWL-DL. The process model is then verified by checking the SWRL-expressions on the ontological representation of business process models using reasoning techniques. The implementation of the verification approach is described in detail.

Chapter 6 (Control Model for Business Process Compliance) provides a formal model of an internal control by capturing its relationship to controlled entities in a business process model (as described in chapter 4).

Chapter 7 (*Pattern Based Design of Controls in Business Processes*) presents a set of control patterns built on top of the control model presented in the previous chapter. It further provides an instantiation mechanism for the control patterns into a control that can be designed in a business process. The objective of the pattern based approach for the design of controls is to simplify the compliance design and consequently improve the usability of the approach.

Chapter 8 (Compliance Validation of Business Process Executions) describes the approach for automatic detection of control violations in the course of business process executions. The approach spans three phases and makes extensive use of the models provided in chapter 4 and 6. The implementation of the validation approach is described in detail. There follows a detailed discussion on related work including current available commercial software products.

Chapter 9 (Assessment) assesses the complexity and completeness of the modeling approach and its application during the execution phase of business processes (Compliance Validation).

Chapter 10 (Conclusions and Future Research) concludes this thesis.

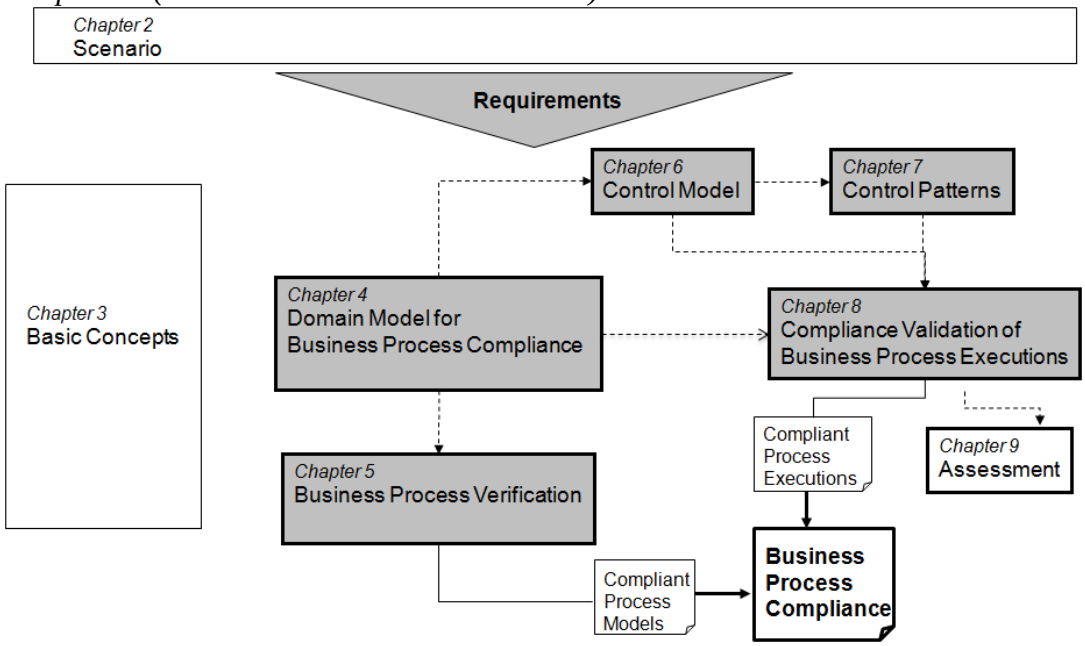

Figure 1 Overview of thesis

Some parts of this thesis are based on the following conference and workshop publications: [Namiri et al., 2008], [Namiri et al. 2007a], [Namiri et al., 2007b], [Namiri et al., 2007c] [Sadiq et al., 2007] and [Namiri et al., 2007d].

The following remarks must be taken under consideration: The formalization of the models in this thesis is written as tuples. In the formalization, a *Set* is written in capital letters, elements in a set are written in lower case letters, and relations between the sets are always in "verb"-form and written in lower case letters. Technical terms are written in *cursive* when used for the first time. The terms *control* and *internal control* are equivalently used in this thesis. Related work is discussed at the end of each chapter.

2 Scenario

We now introduce a scenario in order to expain the motivations behind, and illustrate the contributions of, this thesis. The scenario serves as the basis for understanding the problem space of internal controls compliance in enterprises. It is this scenario that will serve to exemplify the problem descriptions and the provided solutions.

The scenario is a storyboard with the following roles:

- A standard software provider called *EAVendor*. The enterprise applications provided by EAVendor are generic, which can be configured to end-to-end business processes according to the EAVendor customer's requirements.
- Two different customer enterprises (*CustomerA* and *CustomerB*), who have purchased software applications from *EAVendor*. They build their business application on top of the purchased software products.

The following is a description of the Purchase-To-Pay (P2P) -Process provided by EAVendor and the possible compliance requirements of the two customers in terms of optional enterprise-specific controls on the P2P. The P2P process enables the customer enterprises to run their procurement processes. In section 2.1, the P2P application is first mentioned and detailed in its original generic form without possible controls as it is delivered by EAVendor. In 2.2 and 2.3 the specific compliance requirements of CustomerA and CustomerB on Purchase-To-Pay (P2P) in terms of two different sets of required controls according to their specific risk assessment are introduced. In section 2.4 the two use cases are analyzed and challenges are derived from this context of business process compliance that serve to define the requirements in the conclusion of this chapter.

The process model described in section 2.1 represents a minimal best practice description of the P2P- process. The introduced control options of the two use case companies are real life requirements developed internally by SAP in cooperation with the consulting firm PriceWaterhouseCoopers (PwC).

To keep the presentation comprehensible, we omit the following supporting processes (sub-process) from our description:

- Contract negotiation as a sub-process of purchase request processing,
- goods-return-management as a sub-process of goods-receipt, and
- dunning.

2.1 Purchase-To-Pay Business Process Model

In its most general form, the P2P consists of five main sub-processes:

i) Purchase Request Processing
ii) Purchase Order Processing
iii) Goods Receipt Processing
iv) Supplier Invoice Processing
v) Payment Processing

The following is a description of the first three sub processes through the business documents involved in each sub-process. Again, in order to preserve simplicity, we do not go into the details of Supplier Invoice and Payment Processing.

The roles involved in the business process vary depending on the controls required in each customer enterprise (This will be further clarified at a later point in this thesis). Thus we restrict the roles involved in the process model description on the organizational level to those of the departments/organizational units involved. Inside each organizational unit, there will be different roles, which may exist or not according to each enterprise's specific set-up.

Example: Inside a *department X*, there may exist an employee, his manager and the manager of his manager etc. Additionally, depending on the enterprise and its required control, there may exist a controlling unit inside that department. At this stage of the business process description we simply speak about *department X*.

2.1.1 Purchase Request Processing

Purchase Request Processing (see Figure 2) is triggered when an *Operational Department (OD)* releases a *Demand* and submits it to the *Purchasing Department (PD)*. By doing this, OD signals the demand for materials internally. The PD selects a possible *Supplier* and forwards the *quotation* of the selected supplier *(SupplierQuote)* to the OD that submitted the demand. The OD can either accept or reject the quotation. Acceptance is signaled to the PD by the creation of a *Purchase Request* (PR) based on the originally created Demand and its submission to the PD. The PR then has to be approved by the PD.

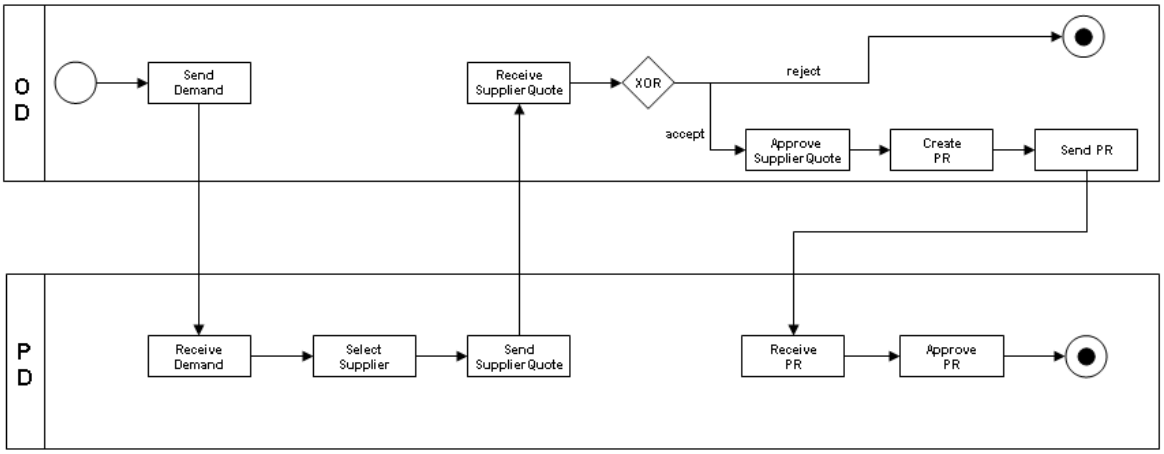

Figure 2 Purchase Request processing as a sub-process of Purchase-To-Pay-Business Process

Please consider that each process-step can be a sub-process in itself. For example, the process-step "Select Supplier" in Figure 2 itself contains process-steps such as "Select SupplierQuote", which for the sake of expedience is not included in the figure.

Business Documents involved

In the following we introduce the structural specifications of each business document in terms of the entities it contains as attributes. Bear in mind that a business document itself may also be a composition of other business documents.

Table 1 Demand Business Document

Attribute	Description
OD	The department that generated the demand
CreatedBy	The employee at OD who generated the demand
CreationDate	The date at which the demand was reported
DemandItem	A demand contains one or more entries (items) in the following structure:

Attribute	Description
Product	The product which is needed in the OD
Quantity	The quantity of the product which is needed in the OD
DeliveryDate	The required delivery date of the item in the OD
ShipToLocation	The address to which the item in the demand has to be shipped

Table 2 Supplier Business Document

Attribute	Description
Name	The name of the Supplier
Address	The postal address of the Supplier

Table 3 SupplierQuote Business Document

Attribute	Description
Supplier	The Supplier who has responded to a *Request for Quotation (RfQ)*
CreatedBy	The employee who has created the quote at the Supplier
CreationDate	The date when the quotation was submitted by the Supplier
ValidFrom	The date from which the quotation is valid
ValidTo	The date till which the quotation is valid
SupplierQuoteItem	

Attribute	Description
Product	The product specification in one quotation item
Quantity	The quantity of the specified product in the quotation item
Price	The price for the specified product and quantity in the quotation item
DeliveryDate (Optional)	The date at which the item can be delivered to the purchasing company. Only specified if a delivery date was required in the preceding Request for Quotation
ShipToLocation (Optional)	The location to which the quotation item can be delivered. Only specified if this information was required in the preceding Request for Quotation

Table 4 PurchaseRequest Business Document

<table>
<tr><th>Attribute</th><th>Description</th></tr>
<tr><td>Supplier</td><td>The supplier to which the purchase request will be sent</td></tr>
<tr><td>CreatedBy</td><td>The employee who has created the purchase request</td></tr>
<tr><td>CreationDate</td><td>The date on which the purchase request was created</td></tr>
<tr><td>ApprovedBy (Optional)</td><td>The employee(s) who have approved the purchase request</td></tr>
<tr><td>TotalAmount</td><td>The total amount of the purchase request</td></tr>
<tr><td>PurchaseRequstItem</td><td>
<table>
<tr><th>Attribute</th><th>Description</th></tr>
<tr><td>Product</td><td>See Product in DemandItem</td></tr>
<tr><td>SupplierQuote</td><td>The quote of a supplier for the product in this item</td></tr>
<tr><td>Quantity</td><td>See Quantity in DemandItem</td></tr>
<tr><td>Price</td><td>The approved price for the item</td></tr>
<tr><td>DeliveryDate (Optional)</td><td>See DeliveryDate in DemandItem</td></tr>
<tr><td>ShipToLocation (Optional)</td><td>See ShipToLocation in DemandItem</td></tr>
</table>
</td></tr>
</table>

2.1.2 Purchase Order Processing

Purchase Order Processing (see Figure 3) starts with the creation of a *Purchase Order* (PO) after a purchase request (PR) has been approved in the purchasing department PD. After the PO has been approved, it will be sent to the selected supplier (denoted with S in Figure 3). The said supplier answers with a *Purchase Order Confirmation* (POC). In the POC, the supplier signals whether or not, as well as how far, it can deliver the ordered goods. At the PD, the POC is used as a basis to decide whether

- the supplier can deliver the order as specified in the PO,
- the supplier proposes to modify the purchase order PO,
- the purchaser rejects the proposed supplier's POC, or modifications to the PO, or
- the purchaser accepts the proposed supplier's POC or the modifications to the order respectively.

Alternatively the sub-process can start when the OD asks the PD to modify an already submitted PO.

Business Documents involved

The following is a breakdown of the structural composition of the business documents involved in the above sub-process:

PurchaseOrder: has the same structural composition as PurchaseRequest (see section 2.1.1.).

PurchaseOrderConfirmation: has the same structural composition as PurchaseRequest (see Section 2.1.1.). In addition a supplier can signal in this business document to which degree it is ready to deliver. The options are: ready to delivery, not able to deliver, and partially able to deliver.

PurchaseOrderConfirmationRejection: has the same structural composition as PurchaseRequest (see Section 2.1.1.).

PurchaseOrderConfirmationModification: has the same structural composition as PurchaseRequest (see Section 2.1.1.).

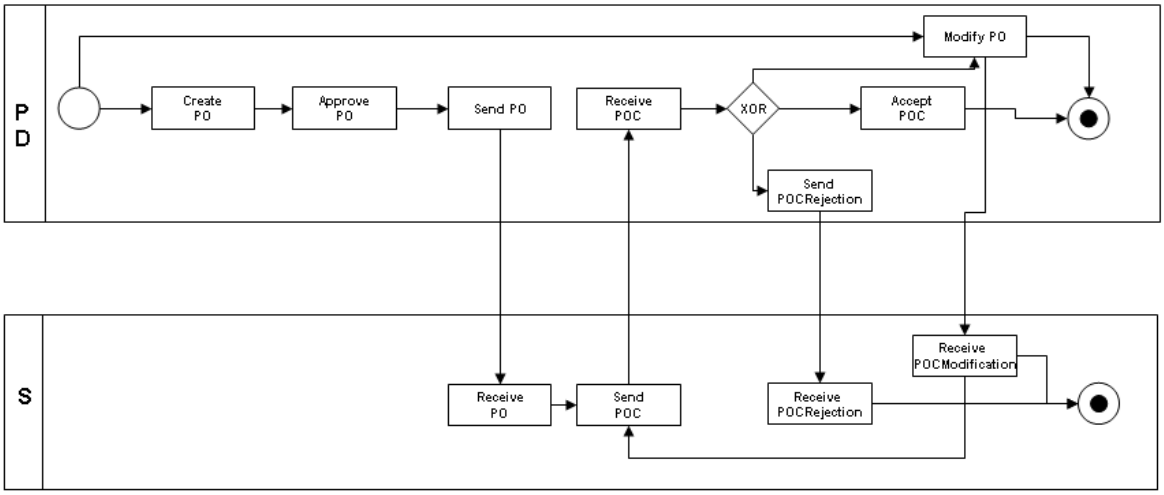

Figure 3 Purchase Order processing as a sub-process of Purchase-To-Pay Business Process

2.1.3 Goods Receipt Processing

The sub-process (see Figure 4) commences when the ordered goods have been physically received in Logistics (shown as L in Figure 4). In this case the *material and inventory* accounts (shown as MM in Figure 4) are updated with details from the *goods receipt* (GR). Further, the arrival of the goods is inputed into the corresponding sub-ledger accounts (accounts payable) for invoice receipt processing (IR) in Accounting (shown as A in Figure 4).

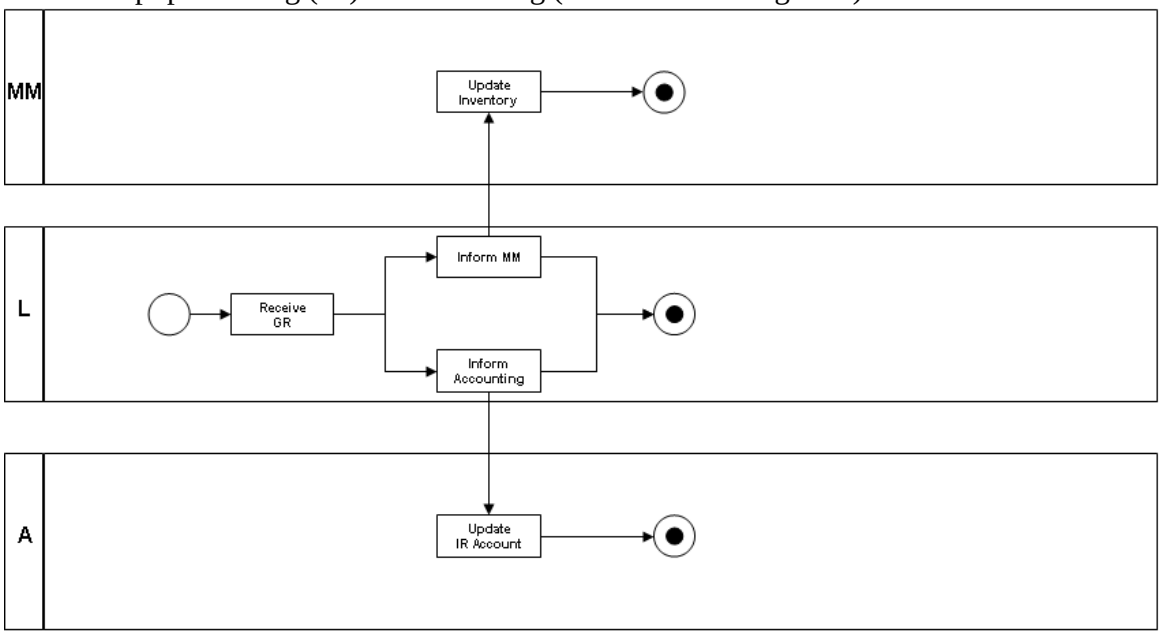

Figure 4 Goods Receipt processing as a sub-process of Purchase-To-Pay Business Process

Business Documents involved:

The structural composition of business documents involved in the goods receipt sub-process can be described in the following manner:

Table 5 Goods Receipt Business Document

Attribute	Description
Supplier	The supplier who has delivered the goods
Location	The location at which the ordered goods were received
ReceivedBy	The employee at OD who originally generated the demand acknowledges that he has received the goods
ReceivedOnBehalfBy	The employee (usually at the PD) acknowledges that he has received the goods on behalf of the employee who originally generated the demand at the OD
ReceiptDate	The date till which the quotation is valid
GoodsReceiptItem	has the same structural composition as the PurchaseOrderItem of a PurchaseOrder-business document described in Section 2.1.2)

Table 6 Inventory Business Document

<table>
<tr><th>Attribute</th><th>Description</th></tr>
<tr><td>Location</td><td>Specifies the location of the inventory</td></tr>
<tr><td>InventoryItem</td><td>The list of materials kept in the inventory
<table>
<tr><th>Attribute</th><th>Description</th></tr>
<tr><td>Product</td><td>The product in the inventory</td></tr>
<tr><td>Quantity</td><td>The quantity of the product present in the inventory</td></tr>
<tr><td>Status</td><td>The specification of availability and assigned status of the item for further usage in business processes. Possible states are “AVAILABLE”, “RESERVED”, “EMPTY”, “ORDERED” etc</td></tr>
<tr><td>StatusChangeDate</td><td>The date at which the status change took place</td></tr>
</table></td></tr>
</table>

For reasons of completeness, we briefly describe the last two remaining sub-processes.

2.1.4 Supplier Invoice Processing

Supplier invoice processing starts with the transfer of information from the preceding sub-processes to Supplier Invoice Processing. Invoice-relevant pieces of information are then reused to create and verify supplier invoices.

2.1.5 Payment Processing

Payment Processing is used to handle outgoing payments to a business process, in this case the supplier.

We conclude the process model description by visualizing the business documents and their relationship to each other. The UML class diagram representation of Purchase Request, Purchase Order and Goods Receipt Processing is depicted in Figure 5.

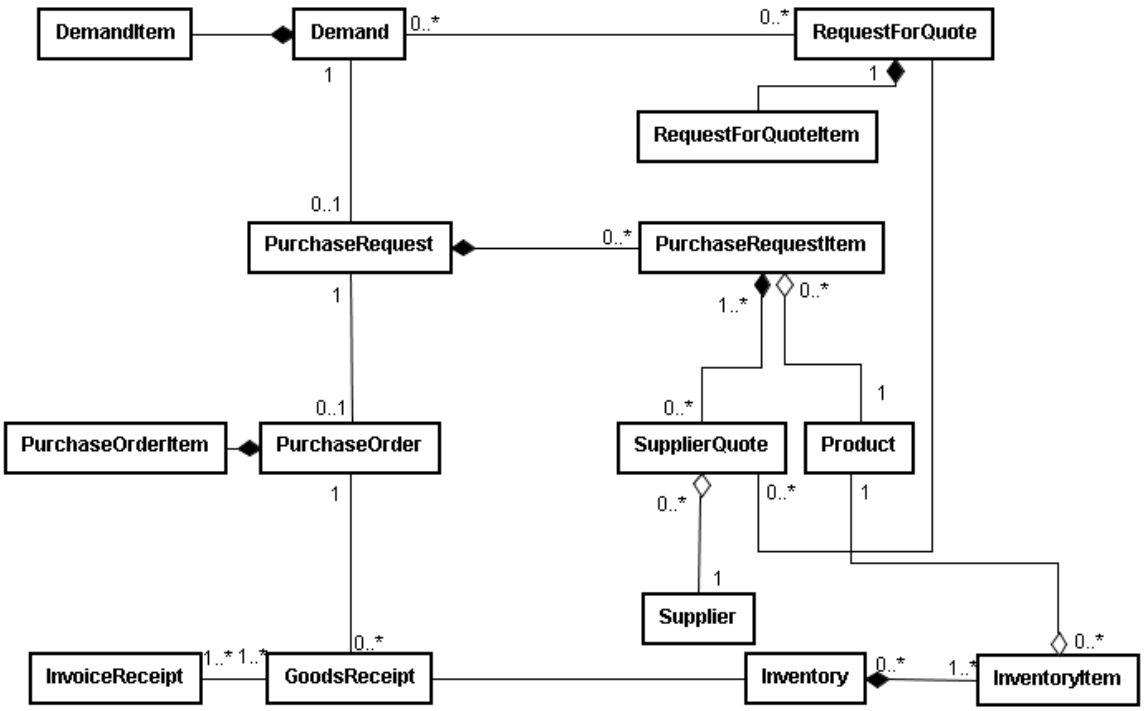

Figure 5 Overview of business documents involved in Purchase-To-Pay Business Process

2.2 Use Case 1 – Internal Controls Compliance Requirements of CustomerA on P2P

This use case describes the business of the company *CustomerA* and how it derives and applies the required controls on the purchasing process.

The company *CustomerA* receives sales orders from its customers through different channels and produces goods based on those orders. In order to fulfill the sales orders, the enterprise requires special materials which are supplied by other vendors in the market. The market situation - especially for two material types (for expedience referred to as material types 4 and 5), which are required for the production - is so highly dominated by a few big players that there exist only a few potential suppliers. CustomerA has decided to keep a certain amount of these material types in his warehouse unassigned in order to avoid production delays.

2.2.1 Identification of Relevant Accounts

Compliance experts and accounting experts at *CustomerA* identify *Inventory* to be among the most important account items in the balance sheet of the company. They decide that all related processes must be included in the risk assessment of the company. Any inventory amounts which are higher than 20% of the total value of balance sheet amount result in an unacceptably high percentage of the company's capital being locked up in material stock. This will have a negative influence on the liquidity situation of the company: Monitoring the capital lockup is a critical task for the company because it has a negotiated fixed credit limit with its house bank which naturally cannot be exceeded. The fixed credit limit is the only source of capital available for financing the long running sales orders which require extensive advance financing. This risk must

be managed, because otherwise *CustomerA* may one day be forced to reject additional customer orders due to a lack of credit.

2.2.2 Identification of Relevant Business Processes

In a second step compliance experts, together with the management of *CustomerA*, identify the business processes that have the greatest impact on the extent of the inventories. They come to the conclusion that the procurement process is a relevant business process for inventory account. Their analysis also points out that the warehousing process itself is in a close relationship with the procurement process. They contact the head of procurement (Purchasing Manager) and the Warehouse Manager and give them the task to describe these processes with the objective of identifying the inherent controls. The completion of these tasks shall ensure that the identified risk does not occur or at least that the possibility of its occurrence is minimized.

The procurement process of *CustomerA* is realized using the P2P provided by EAVendor as described in the previous sub-section. The purchasing manager, the warehouse manager, and the compliance experts meet, and they come up with the list of controls discussed in the next section.

2.2.3 Control Identification

These presented controls below are determined manually at *CustomerA* based on the specific knowledge and expertise which compliance and business process experts possess about their domain and the situation which is specifically relevant to CustomerA. Thus, these controls are specific to CustomerA and as we will see in the second use-case, the same business process in another company will require another set of controls.

Control *CA1*: Purchase Release Strategy

A purchasing guideline is created which states that employees in operational departments must issue purchase requests (PR) for order related materials of type 4 and 5 in such a way that those materials arrive in the warehouse latest one week before the start of production. To support this, every involved employee in the operational department has to create the necessary purchase requests for those material types at least two months before they are due to arrive in the warehouse. This guideline should lead to an elimination of production delays.

Control *CA2*: Check requests for unassigned materials in warehouse

To find a balanced and appropriate warehouse stock size, a minimum and maximal acceptable amound of said stock is defined. The maximum amount of stock exists to lower the warehouse costs. The minimum amount of stock exists to avoid production delays. To support this, all purchase requests must be checked in order to avoid unacceptably high warehouse costs due to an over-quantity of unused materials in stock. The control states that purchase requests containing materials of type 4 or 5 and requesting a total volume amount higher than 10,000 $ will not be approved if the available material type of 4 or 5 currently unassigned in the warehouse is two times higher than the total volume of the purchase request.

Control *CA3*: Minimum Number of Suppliers

Management ensures that a pre-defined number of suppliers have been contacted and provided with the information regarding the requested materials, depending on the volume of the potential transaction or the market situation. This control facilitates the task of supplier selection by ensuring the existence of a variety of quotations from which to choose and thereby assures *CustomerA* the ability to select the quotation with the best conditions (quality, price, etc). All the

contracts with possible suppliers must be up-to-date Concretely the compliance experts define that for purchase requests which contain materials of type 4 or 5 and a total amount higher than 10,000 $ there must exist valid contracts with at least two different possible suppliers and the according supplier quotes are not allowed to be older than six months.

Control *CA4*: Substitute Concept for Purchase Approvers

The approvals for purchase requests have to be carried out in a timely and careful manner. In the event that a person who plays a key role in the purchasing process is temporary unavailable (e.g. illness, vacation) that person has to be substituted by another person. More specifically, all approval tasks for that employee have to be re-routed to his substitute. Further if the approval task inbox of an employee with the approver role contains more than 20 tasks, all further incoming tasks will be re-routed to his deputy. This decision is based on the assumptions that an approval inbox with too many tasks may result in the approver not carefully reviewing each individual purchase request.

2.2.4 Control Effectiveness (As-Is-Situation)

After the controls have been identified, the management has to assure that they are *effective*. A control is considered to be effective if it is used in daily operations, it works as designed, and it is designed in such a way that it in fact prevents or minimizes the occurrence of the risk to mitigate. They call for technical consultants who have knowledge of the Purchase-To-Pay (P2P) provided by EAVendor. These consultants should help *CustomerA* with the implementation of the above controls in the enterprise, namely in the P2P. The consultants recommend using the reporting tools provided by the P2P. A report is basically a periodically generated representation of one or more process-steps, in one or more business processes, which contain one or more business documents based on pre-defined selection criteria. Most controls have to be tested manually by analyzing the generated reports that represent (to a greater or lesser degree) a visualization of the process execution logs. This approach only ensures violation detections after their occurence.The following is a discussion of the approach taken by *CustomerA* in order to assure the control effectiveness. This approach is what we call a control's implementation.

Implementation of control CA1 (Purchase Release Strategy)

A report exists in the P2P sold by EAVendor, which contains the necessary information about the date of the relevant purchase request creation of material types. This report has been configured specially for material types 4 and 5. We call this report *A1Report*. A new role at *CustomerA* called *Controller* in a new organizational unit called C*ontrolling* is created. The employee with this role must generate the A1Report monthly and analyze it in order to assure its adherence to the control CA1 as described in the Purchase Release Strategy. A new role called *Control Tester* is conscribed by the management team. The employee with this role has the task of checking whether the A1Report is generated as required on a monthly basis and whether the report is analyzed for control violations.

Implementation of control CA2 (Check requests for unassigned materials in warehouse)

A report exists in the P2P sold by *EAVendor*, which contains the necessary information about all materials in the warehouse and their states (Reserved, Ordered etc). We call this report *A2Report*. This report has to be generated monthly by the controlling department. The output of the monthly generated A2Reports and A1Reports are analyzed in order to find out whether there exist Purchase Request Approvals for material types 4 or 5 when unassigned material exists in

the warehouse, as designed in the control. The control tester has to verify whether the reports are generated as required and whether they have been sufficiently analyzed.

Implementation of control CA3 (Minimum Number of Suppliers)

The control tester waits until the valid supplier quotes are older than six months in such a way that the minimum number of suppliers control is violated by at least two. He then himself creates a purchase request for material types 4 and 5 as a test to see whether the purchase request gets approved or not.

Implementation of control CA4 (Substitute Concept for Purchase Approvers)

The assignment of deputies and substitutes happens at *CustomerA* in a separate department called Human Resources (HR) running their own systems covered by HR-specific business processes. There exists no technical interface between HR processes and procurement processes realized through the P2P. The management comes to the conclusion that this control can not be tested. Thus, it exists as documentation in the best practice guidelines of *CustomerA*, which has to be followed by employees in a manual way.

2.2.5 Control Test, Assessment and Correction

Based on the results of the effectiveness checks carried out by control tester, the controls will be assessed and corrected if necessary. These are the results reported by the control tester.

Test results for control CA1

For three months the ReportA1 was not generated at all. Thus assessing the effectiveness of this control is not possible for these months. Further the control tester reports that a significant number of purchase requests were not created according to the requirements specified in the control (two months before the material arrives in the warehouse). Further the control tester reports that although the controller detected control violations and informed the employees at the operational department who caused the violation about the purchase release strategy, the same employees later again ignored the purchase release strategy designed in the control.

Test results for control CA2

The defined minimum and maximum stock size for material types 4 and 5 seem to be optimal since the warehouse management costs are reduced and there were no production delays caused by missing materials of type 4 or 5. Further the control tester reports that, for those months following the creation of the A1Report by the controlling department, he was able to verify the effectiveness of CA2, but he is not able to make any statements regarding the effectiveness of CA2 for those months before the A1Report was not generated.

Test results for control CA3

The control tester reports that he generated three purchase requests. For one of them the minimum number of suppliers was 1 and for the other two requests the supplier quotes were older than six months. The latter two purchase requests were passed and approved, thus he considers the control to be ineffective. There were interviews conducted with the employees who had processed those requests. They stated that the sub-processes which are responsible for maintaining the supplier quotes are outside of the realm of their responsibility. Further they argued that they are not able to update the supplier quotes since they do not have access to the

relevant data. After the occurrence of the first purchase request requiring a newer supplier quote, they blocked that purchase request and waited until the supplier quotes got updated. The purchase request was blocked too long in the process and the supplier quotes were not updated. The approach taken to resolve this issue is described in detail in order to clearly illustrate the compliance challenge of business processes today in the industry:

After several inquiries by the operational department who originally had generated the demand about the current state of their request, the employees at the purchasing department decided to continue processing the request although they knew the control was violated. The employees who had violated the control argued that the employee who is responsible for supplier quotes did not follow the procedures in the control. The management interviewed the employee who maintains the supplier quotes. He explained that each supplier quote has a *validFrom* and *validUntil* date, during which a supplier quote is valid. He further explained that one month before the end of the supplier quote's validity (*validUntil* date), he always triggers a process called *RfQ-Processing* provided by the P2P of EAVendor, which generates a request for quotation (RfQ) for certain material types for some selected suppliers in the market. The selected suppliers answer with supplier quotes and if accepted by conditions of CustomerA, the supplier quotes data are updated by him. He argued that since he is not aware of the purchase requests, in particular the total amount of them, he does not know when to trigger the *RfQ sub-process* to update the supplier quotes. This is due to the fact that a valid supplier quote can be taken for a purchase request with a total amount lower than 10,000 $ and the same supplier quote cannot be taken for another purchase request, which has a total amount higher than 10,000 $.

Based on the testing of the controls by the control tester the management of CustomerA decides to define two additional controls:

New Control CA11: ReportA1 Execution Control

This control checks whether the controlling department has *in fact* generated the A1Report and A2Report on a monthly basis as required in CA1 and CA2.

New Control CA12: Escalation of CA1 Violation

If an employee violates the Purchase Release Strategy more than 3 times in 6 months, this has to be reported to his manager.

With regard to the deficiency discovered in the effectiveness of control CA4, *CustomerA* contacts the consultants of the P2P provided by EAVendor. The consultants recommend integrating the RfQ sub-process (Request for Quotation) into the purchase request sub-process. An integration project is carried out to implement the following adapted purchase request sub-process to assure the effectiveness of CA4. The control tester will test the CA4 again to verify its effectiveness, subsequently it will be assessed and if necessary further corrections will be made. Figure 6 shows the extended version of purchase request processing as originally delivered by EAVendor (see Figure 2), which contains the additional RfQ-sub-process. This is a business process variant of purchase request processing specific to CustomerA.

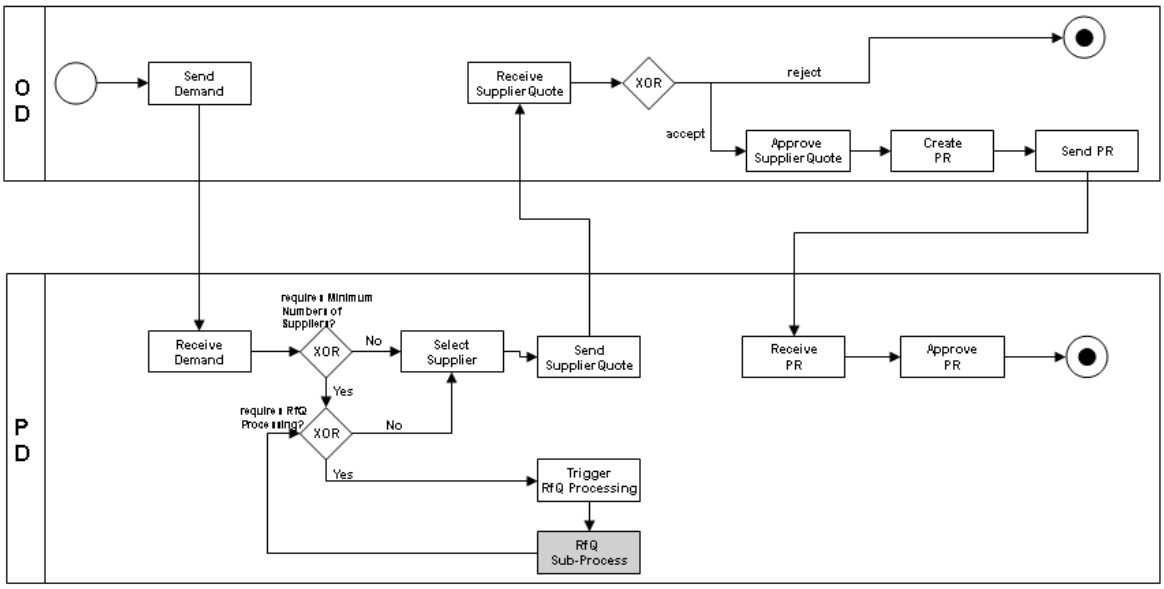

Figure 6 CustomerA's variant of Purchase Request Processing containing RfQ-sub-process

2.3 Use Case 2 – Internal Controls Compliance Requirements of CustomerB on P2P

We now introduce a second use case in order to show the differing compliance requirements of another company for the same business process delivered by standard software provider EAVendor.

CustomerB runs the P2P provided by EAVendor for the realization of its procurement process. CustomerB also buys different materials from suppliers. On the supplier market, especially for material type 2, there are many suppliers. The supplier market is highly competitive, new potential suppliers do emerge, but they disappear quickly. Thus, there exist significant price variations depending on the selected supplier. Hence, CustomerB is forced to enter into short-term business relationships with new, previously unknown suppliers in the market in order to remain competitive with regard to the price of the produced goods. In particular the enterprise will create supplier entries, in order to submit orders, which are used only a few times or even only once.

2.3.1 Identification of relevant accounts

Compliance experts and accounting experts at *CustomerB* identify *Accounts Payable* (Financial obligations against external business partners) as one of the most important account items in the balance sheet of the company. The *Accounts Payable* sub-ledger in the balance sheet of CustomerB is relevant since the accounting department has to authorize payments for several different suppliers, and also has to process a high number of supplier invoices due to the high number of potential suppliers. This situation is critical because the enterprise runs the risk of producing financial misstatements based on inappropriate and fraudulent use of financial transactions having an impact on accounts payable. The heterogeneity of the supplier landscape may result in situations in which employees of CustomerB could create their own private purchase orders for their own use. It is also possible that, due to the short term business relationships, employees at CustomerB could create non-existing suppliers in the procurement

process and create subsequently approve purchases and the according invoices for those non-existing suppliers. Furthermore, the compliance experts point out that insiders who are aware of the supplier market situation for material type 2 could just by chance submit invoices to CustomerB with their private bank information in lieu of a supplier's bank information, in the hopes of receiving the payment meant for a supplier.

2.3.2 Identification of relevant Business Processes

In a second step, the compliance experts together with the management of *CustomerB* come to the conclusion that the procurement process is one of the business processes relevant to accounts payable. The purchasing manager and the compliance experts meet, and they come up with a list of controls to avoid or at least minimize the risk of financial misstatements for accounts payable based on misuse respectively unauthorized use of the P2P. The list of controls here is determined in a fashion similar to that of CustomerA, based on the domain expertise of the purchasing manager and compliance experts and CustomerB's specific risk situation.

2.3.3 Control Identification

The following controls were identified as necessary for the company's purchasing business process:

Control *CB1*: Second Set of Eyes (SSE) on PR Approvals

According to CustomerB's financial situation, compliance experts at CustomerB know that financial transaction having a volume of approximately 10,000$ or higher represent a critical amount for the company. All purchase requests containing the material type 2 with a total amount higher than 10,000 $ have to be approved by two different employees. For this reason, a separate role called *Purchasing Clerk* is created in the purchasing department of CustomerB. The role of the two different employees necessary for approving such purchase requests has to be purchasing clerk.

Control *CB2*: Segregation of Duties (SoD) on PO Creation and Approvals

To avoid misuse of the P2P, the creation and approvals of purchase orders containing material type 2 and a total amount higher than 10,000 $ are carried out by two different employees. The person in the role of purchasing clerk is responsible for the creation of these kinds of purchase orders and the person in the role of purchasing manager is allowed to approve such purchase orders. The role of purchasing manager was pre-existant in the purchasing department (see section 2.3.2).

Notice that the description above actually does not prohibit an employee to be in the two different roles in general, but it requires that the employees creating and approving purchase orders be different and be assigned the required roles.

Control *CB3*: Check One-Time Supplier-Authorization

For immediate response to changes in the supplier market, all employees with the role purchasing clerk get the authority to create one-time-supplier entries in the P2P backend of CustomerB and to create purchase orders for them. One-Time-Suppliers are vendors, for which only general data are stored instead of maintaining bank account data and other company code data as is the case for suppliers with which a long-term relationship exists. This approach should accelerate the whole process of entering business relationships with new suppliers. In order to avoid the misuse of this special authority, the control asserts that the right of One-Time-Supplier-

Authorization will be revoked from those purchasing clerks who have not created a One-Time-Supplier in the last 3 months. This is to limit the group of persons with this right to those who in fact frequently use this type of special authority, in order to avoid its abuse.

Control *CB4*: 3-Way-Match on PR, PO, GR

The entered supplier data (such as name, address, bank account etc.) in PR, PO and GR business documents are compared with each other and must be found to match.

2.3.4 Control Effectiveness (As-Is-Situation)

After the controls have been identified, the management of CustomerB contacts some external technical consultants who should help the enterprise to implement the required controls in the P2P. These consultants have the knowledge of configuration options which is necessary in order to customize the process according to the required controls. If the software doesn't contain these configuration options, it will be necessary to implement it locally at CustomerB in accordance to CustomerB's specific situation. Further, they recommend using the report-tools delivered with the P2P to assure the control effectiveness. In the following we represent the implementation of each control in the P2P model.

Implementation of CB1 (SSE on PR Approvals)

In addition to re-engineering the Purchase Request Processing sub-process of the P2P process, the introduction of the new role purchasing clerk in the P2P requires a reorganization of the purchasing department (PD). This is achieved by the introduction of a sub-unit within the PD called *Purchasing Clerks* (PC). In PC there is a pool of employees who are assigned the role of purchasing clerk (each employee in this unit is called PC1, PC2 etc.). After the purchase request receives approval by an employee in the PC, the items contained in that purchase request business document will be checked. If the items include material 2 and the total volume amount of the purchase is higher than 10,000 $, the request is forwarded to the approval task inbox of another employee in PC using the workflow functionality provided by the P2P. In Figure 7 the customized P2P model, which is a special variant of the original sub-process as delivered by EAVendor (see Figure 7), is shown.

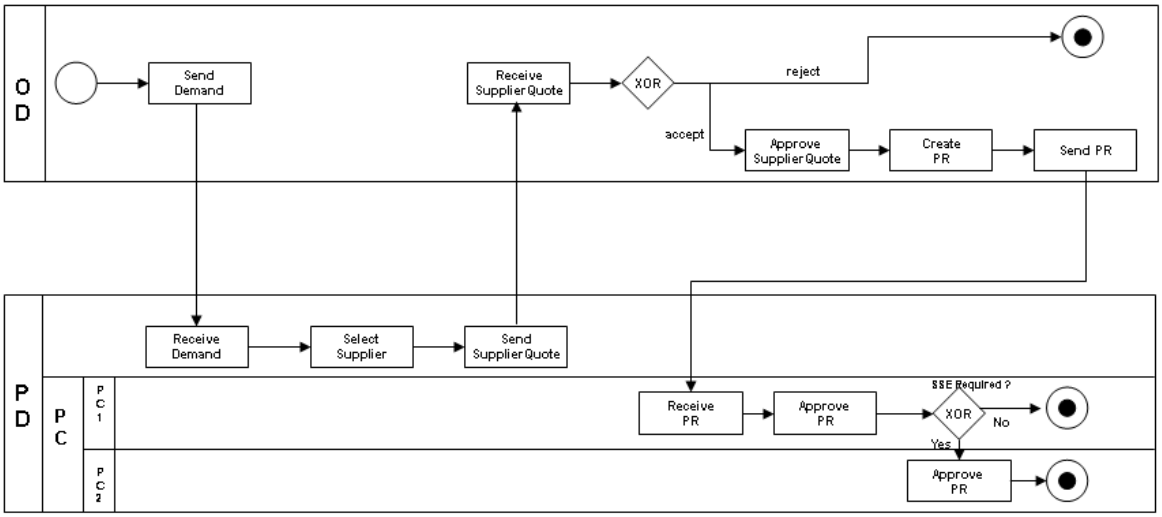

Figure 7 CutsomerB's variant of Purchase Request Processing containing a separate unit PC and two employees PC1 and PC2

Implementation of CB2 (SoD on PO Creation and Approvals)

Before a purchase order (PO) is created, previously approved purchase request is checked whether a separation of duties (SoD, see description of control CB2) on PO Creation and PO Approval is required. The default process behavior is that the POs that do not require a SoD are created and approved by an employee in the role of purchasing clerk. Please notice that this process step can be subjected to a SSE as well, but we that possibility will not be applied to the case in question. In cases for which the purchase request inspection notifies the user that a SoD is required, an "Approve PO" task is created in the task list of a purchasing clerk in PC. After the PO has been approved by this purchasing clerk, the workflow functionality creates an "Approve PO" task relating to that purchase in the task list of a purchasing manager in the PD. CustomerB's variant of Purchase Order Processing, reflecting the implementation of this control, is illustrated in Figure 8, which differs from EAVendor's original delivered process shown in Figure 3.

Implementation of CB3 (Check One-Time Supplier-Authorization)

This control is realized through the report functionality in the P2P by a customized report (let us call it ReportB3). This report contains the list of all One-Time-Supplier activities for each purchasing clerk in the P2P. It has to be generated monthly by the controlling department at CustomerB. Controlling further analyzes the output of the report manually in order to remove the One-Time-Supplier-Authorization for those purchasing clerks, who did not use this functionality as designed in the control.

Implementation of CB4 (3-Way-Match on PR, PO, GR)

The control is implemented as a variant of the Goods Receipt Processing sub-process of the P2P (see Figure 4 for the original sub-process and Figure 9 for the process model containing the control implementation at CustomerB). Upon receiving a goods receipt at a Location (L), the according purchase request and purchase order business documents are loaded from the P2P back-end that should match that goods receipt. Only if the according documents exist and their supplier data are identical to the one on the actual goods receipt the subsequent process-steps to update the inventory and the IR sub ledgers areenacted.

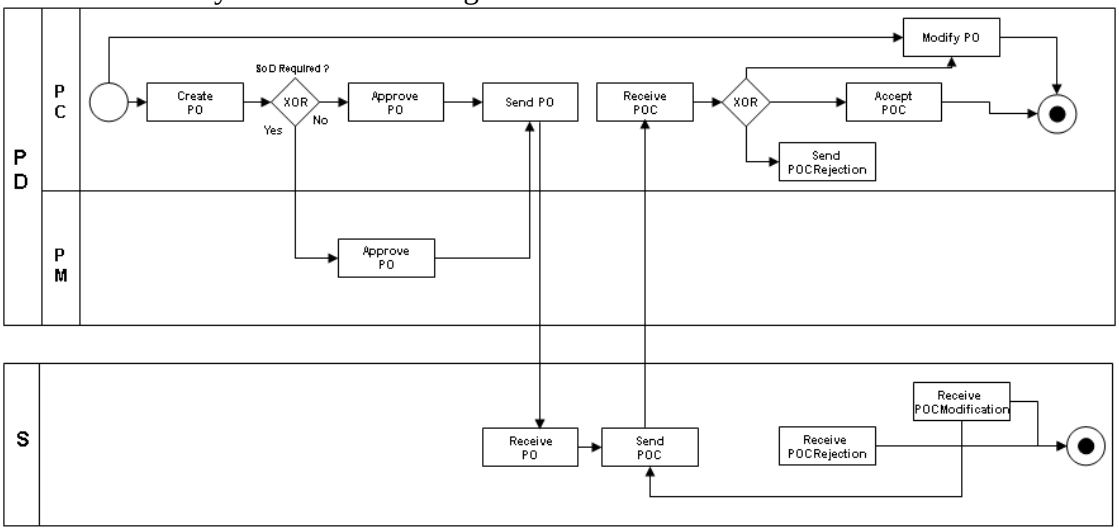

Figure 8 CustomerB's variant of Purchase Order Processing with the purpose of implementing control CB2 (SoD on PO creation and Approvals)

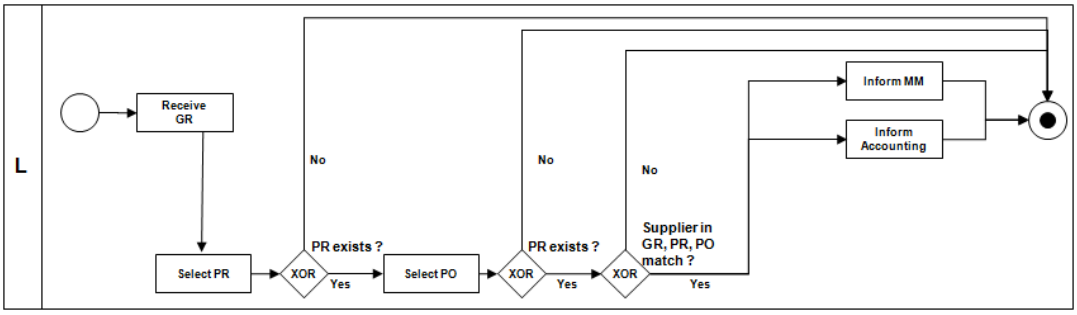

Figure 9 CustomerB's variant of Goods Receipt Processing after the implementation of the 3-Way-Match Control

2.3.5 Control Test, Assessment and Correction

The manual effectiveness checks carried out by the control tester indicate that the controls may require further assessments and corrections.Here are the test results reported by the control tester at CustomerB:

Test results for control CB1

The control tester reported that, three months after the introduction of the control, all the purchase requests requiring a SSE were approved by one single purchasing clerk, after which a purchase order was immediately created. The control tester discovered this compliance failure by randomly selecting approved purchase request documents on a weekly basis. The first assumption was that either a defect in the workflow functionality provided by the P2P or the user management system of CustomerB may have caused the problem. A bug-report was sent to EAVendor. EAVendor tried to simulate the scenario locally, but they reported that the workflow functionality was working correctly. A deeper analysis of the current business configuration of the Purchase Request Processing sub-process at CustomerB showed that it did not contain the required SSE functionality for purchase request approvals, although it had been previously implemented in the process. A subsequent investigation revealed that an employee in the operational department had complained that the purchasing process took too long. He had asked the IT department to look within the system in order to find out what had happened to his purchase request. The employee at IT told him that the purchase request was currently in the task inbox of a second purchasing clerk awaiting approval (because the Purchase had a total value amounting to more than 10,000$). Both employees were not aware of the required SSE on purchases with a total value of over 10,000 $, since it had not been properly communicated to the employees by the management. The employee in the operational department told the employee that this situation was probably a bug in the configuration of the P2P process, since his previous purchase request had been approved by only one employee (that purchase request had a total value of under 10,000$). He even sent a printed version of that purchase document to the IT employee. The IT employee agreed that this behavior must be a bug in the process configuration and removed the SSE on purchase request approvals. The P2P process was hence working correctly so far as the employees were concerned, purchase requests were now approved more quickly, but the P2P process was not working as required in the compliance requirements of CustomerB.

The management team and the auditing consultants convenced a meeting in order to discuss this situation and how they could proactively prevent the occurrence of such situations in the

future. They also invited some external consultants of the P2P in that meeting and arrived at the conclusion that, currently, there is no known technical way to prevent this situation from occurring. The management team and the auditing consultants came to the conclusion that in the future they would have to better communicate the compliance requirements and the controls to their employees. They wrote a compliance guideline for each department and distributed it throughout the enterprise. They further charged the IT manager to prepare a technical guideline, in terms of a check-list for software change management, which was distributed in the IT department of CustomerB.

Test results for control CB2

Surprisingly, the control tester reporteds that, again after three months, the required SoD on purchase creations and approvals were also not effective. All purchase orders had been created and approved by purchasing clerks in the purchasing department. The investigation discovered that after three months a new version of the P2P provided by EAVendor and including some new features was rolled out at CustomerB. The external technical consultants who introduced the new version were not aware of the required SoD on purchase order creation and approval and installed the default purchase order sub-process without the SoD at CustomerB. There were no procedures in place to recognize and avoid such situations.

In this case the management and the auditing consultants also prepared a guideline and a check-list, which had to be followed during and after rollout of new versions of the P2P.

Test results for control CB3

The control tester reporteded that this control was effective and he could not find any violations or issues. The management decided to keep this control as it was designed and there were no corrections necessary.

Test results for control CB4

In this case the control tester reported that this control was also effective, but the control design led to some payments being blocked for goods receipts to suppliers which were valid. This situation was caused by the fact that, due to the competitive market situation for material type 2, many suppliers had accepted orders from CustomerB and subsequently outsourced them and their shipment to other suppliers (sub-contractors) in the market, without informing CustomerB of this change. In this case the goods receipt document received by CustomerB was issued by a supplier-sub-contractor who had not originally received the purchase order from CustomerB. The control design led to the situation that the processing of such transactions was completely blocked and CustomerB had to go through a dunning originated by the original supplier, which did not get its order fulfillment paid by CustomerB. This situation resulted in additional costs for CustomerB.

Based on testing the controls by the control tester, the management of CustomerB decided to add 2 new controls to the guidelines and check lists that were then generated as corrections to the control tester assessments on CB1 and CB2:

New control CB41: Modified 3-Way-Match on PR, PO, and GR

The goods receipt business document and purchase request will contain the *order-number* from PO. Further all goods receipts that are without this number will not be processed and will be sent back to the supplier.

New control CB42: Resolve Blocked Payments

A report provided by the P2P will be generated monthly by the controlling department. It will contain all blocked payments. The grounds on which those payments were blocked must be investigated manually by the controlling department.

2.4 Discussion and Elicitation of the Challenges

We conclude this chapter with an analysis of the described use cases. The discussion should help us to determine the challenges inherent in the compliance management of enterprises, in particular in situations where a standard software provider selling a configurable business process application, in our case EAVendor, aims to provide business solutions like P2P for procurement which can be customized to different customers' compliance requirements.

The challenges identified are the following:

1. Identifying significant accounts, risks and relevant business processes
2. Serving enterprise-specific business process variants
3. Business objectives vs. control objectives
4. Identification and design of controls
5. Maintenance of compliance
6. Heterogeneity and gaps between the roles involved

Each of these challenges is discussed in more depth in the following sub-sections. We describe for each challenge the problem space, the current solution approach, and how it can be improved.

2.4.1 Identifying significant Accounts, Risks and relevant Processes

We'ye seen that, according to enterprise situations and the business environment in which the companies are operating, different accounts are considered as significant for each enterprise. In use case 1 it was the inventory account, and in use case 2 it was the accounts payable. Identifying those relevant accounts is a very sensitive task, which requires domain specific knowledge and especially accounting know-how.

We have further seen that a business process subject to risk assessment that impacts an account is not necessarily the criterion that makes an account a relevant account. For example, at CustomerA the relationship between its negotiated fixed credit limit with its house bank and its long-running sales orders made inventory accounts a relevant account, although the procurement process and the sales order process were at first glance completely independent and autonomous processes. But this situation made the procurement process a relevant process for internal controls compliance of CustomerA.

Further, we can see that although for both enterprises the same business process, procurement, was identified as a relevant business process, the respective risks faced by each enterprise regarding the procurement process were quite different: While CustomerA was facing the risk of delays in production and too-high warehouse costs, CustomerB was facing the risk of fraud.

The task of identifying the above entities is carried out, at present and for the most part, manually, based on the expertise of compliance and accounting experts together with business process experts. The completion of this task requires a lot of time, and therefore it is very costly. A solution to the manual approach taken today would be to capture the expertise of the different roles involved in business process compliance as precise formal descriptions of the entities and how the roles and the entities interplay in the domain of compliance. Such a formal description would need to include possible relationships between these entities and the formal description of

situations for which and in which an account can be considered as significant, a business process can be considered as a relevant for a significant account etc.

The task of automating the identification the relevant accounts, risks, and the relevant processes is beyond the scope of this thesis. Our contribution begins after their identification by accounting and compliance experts.

2.4.2 Serving enterprise-specific business process variants

We saw that according to the enterprise specific risk assessment, the required controls on identical business processes can be quite different. At the end, therefore, the way a business process works, i.e. the way it is configured, will be enterprise-specific. For instance if we take a look at the purchase request processing of each customer enterprise after the realization of the compliance requirements, they each represent two different variants of the same sub-process, but with identical *business objectives,* namely "receive demands and create and approve their relevant purchase requests".

A standard software provider like EAVendor faces the challenge of having to produce its software in such a way that it can be adapted to each customer's enterprise-specific requirements, in this case its compliance requirements. But at the same time, the software provided must not be too generic, because then the introduction and maintenance of the software on the customer‘s end becomes too complicated. This is due to the fact that those missing features (in our case the controls) come at the cost of generality in the software. A model-based description and deployment of the involved entities in business process compliance can support EAVendor in providing a mechanism for flexible introduction of enterprise-specific business process variants within customer companies: the existence o precise formalized models allows an automated approach in this regard. The solution to these challenges, as developed in this thesis, must satisfy the following requirements:

1. Representation of the controls and business processes each as separate modeling entities
2. Capturing, on the model-level, the relationships that exist between controls and business processes
3. A deployment mechanism for integrating the separately modeled controls on business process models using the relationships that exist between controls and business processes

Satisfying the requirements above enables the business processes controls to be independent from each other at a modeling level. This independence of control and business process models can support the automated detection of any control violations, because the conditions that describe a control are captured separately and can be evaluated.

A Software provider like EAVendor could then provide a repository of business process models and a repository of control models for its customers. This way the controls would not be too tightly integrated in business processes delivered but would be present and ready to be deployed on the customer's end. This approach would assure that business processes and controls are both reusable, in different business-level contexts at different customer companies having different compliance requirements.

2.4.3 Business Objectives vs. Control Objectives

As we've seen, in none of the cases for which the control tester discovered a deficiency in control effectiveness was the business objective of the business process violated. In all cases, the business process was working "correctly" insofar as it fulfilled the business objectives for which it was originally designed and implemented. In our use-cases the business objective was simply to "order and receive goods". For this reason each customer company was forced to define a

separate role “control tester”, who had to check that controls were working correctly (Control Objective) but also that the business processes were still working correctly despite the existence of controls (Business Objective). If an enterprise failed to define such a role, then many of the controls would not work as designed in daily operation, even if at first glance each control design seemed correct. The same can apply to business processes: if a control for a business process is working correctly but the business process itself is not working correctly because of or in spite of the existence of that control caused by problems in its design.

Today, the task of testing the controls and finding deficiencies in their design is usually a time consuming and expensive one, mostly carried out manually. Enterprises require mechanisms to achieve a higher level of automation in assuring the effectiveness of controls. They make this requirement known to the software vendors. Although it is basically the task and responsibility of customer enterprises to be compliant, the software vendors serving those enterprises face the challenge of providing solutions for their customers which allow those customers to fulfill their control objectives.

We suggest a solution to this challenge through a modeling approach for controls and business processes, by separating their respective designs. A monitoring mechanism during execution of business processes, in addition to business process and control design, assures that the business and control objectives of a business process are satisfied.

2.4.4 Identification and Design of the Controls

Controls have to be identified and designed very carefully. The two crucial aspects here are the set of controls in an enterprise and the relationships that exist between them:

Control Set

The defined controls may inhibit the efficient execution of business processes. Basically, the challenge is to prevent business processes from becoming too complicated and to make sure they do not require too much time and resources, as well as knowledge that is usually outside of the scope of the user’s business knowledge, to enact. In such a situation users of business processes may not accept the procedures necessary to the fulfillment of control objectives. It is for this reason that management and compliance experts cannot simply define as many controls as possible on a business process to assure the enterprise’s compliance. This would result in the compliance itself representing a risk for the enterprise in that the enterprise might face the risk of not fulfilling its business objectives. Such results were certainly not intended by compliance requirements such as the Sarbanes-Oxley Act.

Control Inter-Dependency vs. Control-Contradiction

Closely related to the above challenge are the relationships which obviously exist between controls. As we saw in the scenario, in certain cases, a single control may not be autonomously effective as designed. For instance, discovering a violation of a control through comparison of certain attributes of two reports only works if both required reports have indeed been generated. In this context, the crucial problem met in practice is that many controls are currently being manually assured through periodically generated reports that have to be compared to each other. If an employee in charge of generating and comparing different reports to each other fails to fulfill his duties, the control cannot be effective, because the necessary reports responsible for detecting any control violations are not compared to each other or do not even exist. In such cases a separate control is required, one which ensures that a certain report has in fact been generated. To relate this situation to the scenario, in case of CustomerA the effectiveness of CA1 and CA2

could only be achieved when, after the test and assessment phase for that control, additional controls CA11 and CA12 were added.

We provide a solution to the above issue by providing a precise model of the necessary artifacts involved in a control. For instance a model of a *report* can be modeled as an activity that may be invoked in the system. Consequently the generation of a report or its comparison to other reports can automatically be monitored. By using this approach a higher degree of control automation can be achieved.

On a similar note, two different controls can contradict each other or can have an impact on the business process, when occurring in a certain combination where they block the coninuation of the process. Recall for instance in the case of CustomerB the control CB1, which required a SSE on purchase request approvals by two different purchasing clerks. If in such a case a compliance expert defined a SoD on “Receive PR” and “Approve PR” where the second process step had to be done by a purchasing manager, the process would fall into a “blocked” state after the purchase request has been received by the purchasing clerk.

Thus, the controls must be carefully designed in conjunction with a detailed study of the current setup of the business processes and the other controls. At present, this is done manually in most cases and involves different roles in an enterprise. This is a costly and time consuming approach. Enterprises require support in order to achieve a higher level of automation in the effective design of compliance for their business processes. We contribute an improvement to the described situation by providing a model-driven approach for the design of controls and their effectiveness in business process executions.

2.4.5 Maintenance of Compliance

Once compliant does not mean forever compliant. In addition to regulation bodies that periodically audit the enterprises to assess their compliance, purely technical circumstances have an impact on the compliance level of enterprises.

Software Change Management

As the existing technical environment of enterprises change, the business processes that were compliant before may no longer be compliant. This ever-changing context means that a control which is effective in a business may be eliminated or contain some issues after updates in the technical environment relevant to the business process. We saw this in the scenario case of CustomerB when a new version of the P2P provided by EAVendor was rolled out (Overwriting the SoD -CB2 by the original default process model provided by EAVendor). The challenge specific to this context is that such situations may be detected very late, if at all. Manual procedures are in place in today’s business world which are supposed to avoid the occurence of such mistakes, as is the case with most manually controlled tasks they are costly and take a trial and error approach. Consider that standardized approaches such as IT Infrastructure Library (ITIL) represent best practice descriptions for management of IT-Systems, which can be applied on a voluntary basis. One major problem is that a control violation is hard to detect, as in most cases the business processes continue to work properly insofar as they continue to fulfill their business objectives even if a control is violated. There are approaches required to automatically detect such control violations. A certain level of automation can be achieved by static verification of the current configuration of process models: one can verify whether they contain the required controls. However, such a static approach only insures that the business process model contains the control and cannot serve as a proof of compliance, i.e. of control effectiveness.

The approach suggested by this thesis automatically recognizes a control violation during business process execution by using the model of that control and describes how a business process must behave in daily operation under existent of that control in the business process.

Autonomous Business Processes

Enterprises run many different business processes, often each are designed and deployed independently from each other. Moreover the business processes are subject to the supervision of different stakeholders and may be maintained by different teams in an enterprise. This approach is in many regards advantageous, but when it comes to compliance this situation represents a large drawback to assuring the effectiveness of controls. As we saw in the case of CustomerA, the CA3-control defined in Purchase Request Processing was not effective because the sub-process RFQ-Processing was not integrated into the P2P. Even the integration of the business process does not guarantee the long-term effectiveness of CA3, because CA3 will only be effective so long as RfQ-Processing is working correctly.

We simplify the business processes of enterprises by providing a flexible instantiation approach for a business process (in the above example RfQ-Processing) that is required for compliance of another business process (in the above example Purchase Request Processing) during execution time of the relevant business processes. The possibility for declaring the necessity of instantiation of the required business processes is provided in the model of control.

2.4.6 Heterogeneity and Gap between roles involved

We saw in the scenario that there are different roles involved in the compliance management lifecycle of business processes. Accounting experts identify relevant accounts and balance sheet items, compliance experts identify and design the controls, business process experts, like the purchasing managers in our scenarios, have detailed knowledge of relevant business processes which comes into the play when identifying relevant business processes, and on a technical level there are IT experts who are responsible for implementing and maintaining the controls on the business process and software applications which realize those processes. Control testers have the task of checking the effectiveness of controls, which is subject to later assessment etc. Additionally there are external auditing firms who certify the compliance of an enterprise.

Within an enterprise, each person in each of these roles has to cooperate in order for the company to achieve compliance. Each of these roles has its own interest, and expertise, and each uses different terminology. The alignment of these roles is one of the important reasons why achieving and retaining compliance requires a special treatment beyond the technical challenges discussed so far.

Enterprises require a shared cooperative environment for these different roles, a need which they address mainly to theirstandard software providers such as EAVendor. We provide a cooperative environment for designing and deploying controls on business processes, which can be used by compliance experts in conjunction with business process experts. Further we provide a set of control patterns, which provide the language and terminology for compliance and business process experts to communicate in the domain of business process compliance.

2.5 Conclusion

We have introduced two use cases for compliance. Their discussion revealed that compliance of business processes, in particular their control requirement, is *orthogonal* to usual known

artifacts from business process management, namely business process design and execution. This is mainly due to the crucial difference between control objectives and business objectives in business processes. Business process compliance can be considered as a separate layer of business process management activities, which is enterprise-specific. In today's business world, the response to this is the definition and introduction of separate teams, organizations, and roles in enterprises that have the task of interpreting and achieving compliance for an enterprise. But the question of how best to support such a layer on a technical level is not yet answered.

The crucial questions are: how to design such a layer and what are the relationships between a separate layer for compliance and the existing business processes. The requirements on this separate layer inherent in compliance were elicited in section 2.4 : the scenario using the two different compliance approaches taken by the two use case companies for the purchase-to-pay-process. Here is a summary of the resulting list of requirements:

i) Formal model of accounts, risks and business processes including their relationships to each other
ii) Semantic description of *significance* of an account, *relevance* for a business process and possible *risks* on a *relevant* business processes
iii) Identification of necessary controls for a business process
iv) Model of a control and its relationship to business processes
v) An approach for separated design of controls and business processes
vi) An approach for deploying separately designed controls on business process models
vii) Monitoring of control effectiveness during business process executions
viii) A mechanism for handling possible control violations
ix) A cooperative environment for compliance and business process experts for design and management of controls
x) A common terminology of the domain in which the involved roles communicate

In the rest of this thesis we will introduce and discuss methods and architectures for the design and application of such a separate layer for compliance of business processes. As it was previously mentioned, the requirements i-iii will not be addressed.

3 Basic Concepts

Obviously, in order to understand business process compliance, one must first understand the underlying concepts of business processes and of compliance. It is important to understand these concepts in order to develop their intersection in precise models that can be used to describe a compliant business process and its execution. In this thesis we define compliance as 'adhering to regulations in the area of enterprise risk management' (ERM). As was stated in the introduction, these regulations are in place in order to assure effective internal controls within companies. The main objective of this chapter is to introduce the core concepts of internal controls and business process management which are used throughout this thesis.

In section 3.1 internal controls are introduced, along with an internal controls framework proposed by the Committee of Sponsoring Organization (COSO). The description of the compliance domain included in this thesis is highly influenced by the COSO recommendations. The provided models and the approach for achieving business process compliance are in part a result of our analysis of the COSO framework. Throughout this section we will refer to SOX as 'a popular regulation requiring internal controls compliance'. This introduction to internal controls concludes with an analysis of some other regulations in the area of ERM in order to expose their relationship to internal controls.

In section 3.2 the key life-cycle phases of a business process are introduced. After their introduction, they are put in relation to the internal controls by discussing the application time of internal controls in business processes (Detective controls vs. Preventive controls). Based on the relationship exposed, a detailed argumentation in favor of the model-driven approach presented in this thesis as being the core requirement for realizing a preventive nature of business process compliance is given (section 3.3).

3.1 Internal Controls

There are several definitions and interpretations of the term *Internal Control*, as it affects so many different parts of an organization as well as their responsible stakeholders. It also impacts an organization on different levels, from strategic management level to the way that IT systems are managed. Thus the way internal controls are designed and assured may depend on the organization, its risk and business environment and the organizational level at which internal controls are applied.

The committee of Sponsoring Organization (COSO) proposed in [COSO92] a popular and mature framework for setting up internal controls in enterprises. COSO defines internal controls as:

"a process, effected by an entity's board of directors, management, and other personnel, designed to provide reasonable assurance regarding the achievement of objectives in the following categories: a) Effectiveness and efficiency of operations; b) Reliability of financial reporting; and c) Compliance with laws and regulations."

Effective internal control is supposed to assure that an organization generates reliable financial reporting and complies with the laws and regulations to which it is subjected. Naturally, internal control cannot absolutely insure that the objectives of an organization will be met. This often depends on exterior factors, such as competition or technological innovation. These factors are outside of the scope of internal control; therefore, effective internal control provides only timely

information or feedback about any progress which has been made towards the achievement of operational and strategic objectives, but cannot guarantee their achievement.

Internal control plays an important role in the prevention and detection of fraud under the Sarbanes-Oxley Act (SOX) [SOX02]. The SOX requires that companies perform a risk assessment and assess related controls. This typically involves identifying scenarios in which theft or loss could occur and determining whether existing control procedures effectively manage the risk level (see the use case example presented in chapter 2). The SOX comprises 69 different sections organized in 11 titles ranging from additional corporate board responsibilities to criminal penalties and describes specific mandates and requirements for financial reporting. Here is an overview of those SOX sections affecting software and IT-systems, which are outlined in Title 3 (Corporate responsibility) and Title 4 (Enhanced financial disclosures):

- Title 3, section 301 (Public company audit committees): The auditing committee shall establish procedures for confidential and anonymous reports by employees of an organization regarding questionable accounting or auditing methods and issues.
- Title 3, section 302 (Corporate responsibility for financial reports): Management is responsible for effective disclosure of controls and procedures regarding financial reporting, operations and compliance, and disclosure of significant deficiencies in internal control to audit committee and external auditors.
- Title 4, section 401 (Disclosure reports): All material corrections (corrective actions) must be included in the financial reports, which have been identified by external auditors. Further, investors must be provided with a clear understanding of the company's balance sheet situation and the way the balance sheet and its items are affected by the activities in an organization.
- Title 4, section 404 (Management assessment of internal controls): Periodic reports should include a report from management on the effectiveness of internal controls over financial reporting. This report should contain documentation on the control designs and effectiveness, their tests, disclosure of any material weaknesses, and their attestation by external auditors.
- Title 4, section 409 (Real time issuer disclosure): Timely information on material changes in the financial conditions and operations of the company must be provided.

The most important part of the SOX for IT-systems is section 404, which requires the realization of effective internal controls in companies. Section 302 defines the requirements defined in section 404 as 'management responsibility'. In this thesis our most important goal is to provide novel methods and solutions for the implementation of **section 404**, which will implicitly support management in substantiating the section 302 requirements of the Sarbanes Oxley Act.

The internal controls compliance must be reported periodically to external partners. The level of possible deficiencies in the internal controls of a company will be assessed based on these reports. The internal controls compliance of a company will then be decided based on these deficiencies.

COSO defines deficiency as "*a condition within an internal control system worthy of attention. A deficiency, therefore, may represent a perceived, potential or real shortcoming, or an opportunity to strengthen the internal control system to provide a greater likelihood that the entity's objectives will be achieved.*" There are 3 levels of deficiency [PCAOB04]: control deficiency, significant deficiency and material weakness. [PCAOB04] gives the following definitions for different levels of deficiency:

- *"A control deficiency exists when the design or operation of a control does not allow management or employees, in the normal course of performing their assigned functions, to prevent or detect misstatements on a timely basis"* [PCAOB04, section 8]

- *"A significant deficiency is a control deficiency, or combination of control deficiencies, that adversely affects the company's ability to initiate, authorize, record, process, or report external financial data reliably in accordance with generally accepted accounting principles such that there is more than a remote likelihood that a misstatement of the company's annual or interim financial statements that is more than inconsequential will not be prevented or detected"* [PCAOB04, section 9]
- *"A material weakness is a significant deficiency, or combination of significant deficiencies, that results in more than a remote likelihood that a material misstatement of the annual or interim financial statements will not be prevented or detected"* [PCAOB04 section 10].

Although material weakness is the relevant issue for public reporting on internal controls, there exists no automatable process for identifying a material weakness in the internal controls of a company. This is because the making of such a determination cannot be expressed in only quantitative terms. Material weakness can include several concepts: the level of risk, materiality in relation to the entity's financial statements, and the timeliness of error detection. A shortcoming in the internal control system of an enterprise may actually be considered as a material weakness, whereas the same shortcoming (situation) in another enterprise may only be considered as a control deficiency by the auditing experts. The reason for this is that the impact of a loss caused by a deficiency is relative to the size and to the financial situation of a company.

3.1.1 Roles and their Responsibilities in Internal Controls Compliance

We introduce the roles and their responsibilities that a publicly traded company must enact in order to comply with SOX 404 and 302. This model will then serve as a generalized pattern for compliance with other regulations in the area of ERM (they will be described in sub-section 3.1.3) which are associated with internal controls regarding necessary roles and their responsibilities.

SOX 404 requires the assignation of the following distinct roles:

- **The Management Team**: embodied in the person of Chief Financial Officer (CFO) who takes personal responsibility for an effective implementation of internal controls.
- **Compliance Experts**: The management team is usually supported by a special team or even a department dedicated to SOX compliance and internal audit experts. The task of identification, design, and effectiveness checks of the controls is delegated by the CFO to these experts, whom we will call compliance experts. Compliance experts act in conjunction with requests from the CFO. They have detailed knowledge of regulatory requirements. They have little or no knowledge of the realization of business processes in an enterprise. Their main task is rather to define and monitor the necessary controls according to the risk assessment, and to notify other entities in the enterprise in the case of control violations. They do not define how to bring a process into a compliant state because this is the task of the business process experts.
- **Business Process Experts**: These groups of people are responsible for relevant business processes residing in operational departments, where the business processes are run. Their task is to implement the identified controls according to their design. Typical characteristics of business process experts include having the knowledge of configuration and maintenance of processes while keeping business objectives (goals) in mind. For example, the business objective of a purchasing process is simply to set up a process in which internal orders can be processed and sent to suppliers so that the ordered goods can be received and supplier invoices paid. It is obvious that in large scale ERP systems this role is carried out by different persons, or even by different organizational units. This

group of persons in an enterprise usually has little or no knowledge of regulations and compliance requirements, but very detailed knowledge of how a process is implemented.

- **Business Users**: Business users are the employees in a company who actually fulfill the business tasks (using business processes). They must behave compliantly by using the business processes in accordance with the controls. Compliant behavior may or may not be system supported.
- **External Auditors**: These are official bodies or external firms who assess and certify the effective design, documentation, and implementation of internal controls. For most large public companies, external auditors will be from one of these four firms: Deloitte, Ernst & Young, Price Waterhouse Coopers, and KPMG. This work will however not touch upon their role in business process compliance.

3.1.2 COSO - A Framework for Internal Controls

Section 404 of SOX requires that management of the public companies implement and document a system for internal controls compliance. But SOX does not give any guidelines on how to realize an effective internal controls system. It rather recommends using well-known frameworks as a best practice to set up an effective internal control process. COSO, which was mentioned earlier, is mostly recognized by regulation bodies and auditors as a de facto standard for the realization of the internal controls system. In the internal SAP project ICCOMP (see section 1.2) COSO was selected as the guideline framework for defining internal controls on business processes.

COSO emphasizes an internal control which is **not** specific to one event or circumstance, but rather to a series of actions that permeate an entity's activities. These actions are pervasive and are inherent to the way management runs the business. This implies that being SOX 404 compliant is not a one-time task. It is a continuous process due to two facts: 1) Internal control is itself a process and 2) SOX 404 compliance must be periodically reported.

COSO introduces two key terms for setting up an effective internal control process: *Control Objectives* and *Control Components*.

3.1.2.1 Control Objectives

To provide a context for the implementation of its integrated framework, COSO sets out three types of objectives, referred to as control objective types, for management and auditors. Control objectives of a control provide the measurable targets in view of which the company can define controls. COSO differentiates between the following types of control objectives [COSO92]:

- **Operations**: Controls of this type should ensure that the company is operating effectively with respect to safeguarding resources against loss.
- **Financial Reporting**: Controls of this type should ensure the preparation of reliable published financial statements (within the US in in accordance with Generally Accepted Accounting Principles – GAAP). To support this objective a series of assertions underlying an entity's financial statements must be made regarding the following aspects:
  - *Existence or Occurrence:* Assets, liabilities, and ownership exist at a specific date and recorded transactions represent events that actually occurred during a certain period.
  - *Completeness:* All transactions and other events and circumstances that occurred during a specific period, and should have been recognized in that period, have, in fact, been recorded.

- *Rights and Obligations*: Assets are the rights, and liabilities are the obligations, of the entity at a given date.
- *Valuation or Allocation*: Asset, liability, revenue, and expense components are recorded in appropriate amounts in conformity with relevant and appropriate accounting principles. Transactions are mathematically correct and appropriately summarized, and recorded in the entity's books and records.
- *Presentation and Disclosure*: Items in the financial statements are properly described, sorted, and classified.

- **Compliance**: Controls of this type assure that the company adheres to all industry- and environmental-specific laws and regulations to which the company is subjected. They are dependent on factors such as the industry in which the complany operates (food, health and medicine, transport and logistic etc), the country the company is located in, etc.

3.1.2.2 Control Components

COSO recognizes that industries, companies, and management practices all differ. Therefore, COSO recommends evaluating the optimal application of the framework in the subjective context of the specific company concerned with realizing the internal control process. The fulfillment of the control objectives introduced above is achieved along five essential control components which should be acknowledged by the company and auditors. The description of control components is narrowly adhered to by COSO in order to meet the dynamic process of internal control and to control for its subjective nature. The five components are interrelated and are derived according to the way the management runs the business of a company. These components are:

- Control Environment
- Risk Assessment
- Control Activities
- Information and Communication
- Monitoring

3.1.2.2.1 Control Environment

This component refers to the overall tone of the organization. . It includes integrity and ethical values. It also applies to the competence of the entity's people, the management's philosophy and operating style, the way management assigns authority and responsibility, and organizes and develops its employees, as well as the attention and direction received from the board of directors. The control environment is reviewed by external auditors who pay attention to the following aspects:

- Existence and implementation of codes of conduct and other policies regarding acceptable business practice, conflicts of interest, or expected standards of ethical and moral behavior. This is directly related to section 301 of SOX.
- Dealings with business partners including suppliers, customers, investors, creditors, insurers, competitors, and auditors, etc.
- Pressure to meet unrealistic performance targets – particularly for short-term results – and the extent to which compensation is based on achieving those performance targets.

Control Environment is the most subjective component in COSO. It is not discussed in this thesis.

3.1.2.2.2 Risk Assessment

Risk assessment is a methodological approach to the continuous identification, analysis, control, and monitoring of critical situations and events by proactively using adequate processes, methods, and tools in order to balance the effort of managing events and the impact of these events.

The COSO risk assessment process calls for risk identification and analysis as a company generates revenue and manages expenses. The risks which are due to internal or external factors and which affect the company's achieving its defined control objectives are those upon which the most focus is placed. The risk assessment should follow the specific company's value chain of activities [Green, 2002]. The value chain of activities is directly reflected in the balance sheet of a company (See as examples for risk assessment the use cases in 2.2 and 2.3). Obviously the value chain is different for each company. Risk assessment is an iterative process and should be tightly integrated into the planning process of an enterprise.

Further, according to COSO, risks have to be separated into two different levels: the *entity-wide level* and the *activity level*.

Entity-wide risks influence the company as a whole. External factors influencing *entity wide* risks can include: Technological developments, changing customer needs or expectations, changing competition situations, new legislation or regulation, natural disasters, and global economic change. Internal factors influencing entity-wide risks can include: Disruption in information systems, quality of personnel, methods of training, and motivation; change in management responsibilities, nature of the entity's activities and employee accessibility to assets, and unassertive or ineffectual board or audit committees.

Dealing with risks at the *activity level*, according to COSO, should help the company to focus on major business units and existing functions therein, such as sales, production, marketing, etc. This is done by identifying control objectives for business processes that affect significant accounts. To avoid overlooking relevant risks, it is best to identify potential risks while ignoring the likelihood of risk occurrence. We will focus on risks associated with the activity level of a company.

A further COSO requirement is to *manage the change* in an enterprise. Every entity needs to have a formal or informal process, whose purpose is to identify conditions that can significantly alter its ability to achieve objectives. Some of the changing circumstances that require special attention are:

- Changed operating environment
- New personnel
- New or revamped information systems
- Rapid growth
- New technology
- New lines, products, and activities
- Corporate restructuring

Basically, managing the change states that business process reengineering has to be followed by an updated risk assessment in an enterprise. This emphasizes the nature of risk assessment as an iterative process.

3.1.2.2.3 Control Activities

A control activity is a procedure instigated in order to mitigate the risk that a control objective may not be achieved. It is very important to emphasize the nature of a control activity (in the following, a control) the way the compliance and auditors understand it, since it may easily be

misunderstood as the control objective itself. Two essential elements of a control are its *policy* and its *procedure*: the policy is in close relationship with the control objective, which was the original motivation for defining the control. The procedure of a control assures that an action, or a set of actions, will be carried out in order to ensure the policy of a control. For example if an approval or authorization for a certain business level activity is required (the policy), then the control exists in order to check the approval or authorization procedure for that business level activity is in fact carried out as required. The controls are by no means bounded to the information processing and IT level operations of an enterprise. For example, the policy of a control in an enterprise which holds precious goods (for example gold) in stock is to have a door to the stock. Having a door does not assure the control effectiveness in this case, since the door could always be open and everyone could enter. The procedure of the control is then to check whether i) the door exists ii) it is closed and iii) only authorized personal may enter.

For SOX 404 compliance, a company may identify several risk areas that require mitigation by control procedures. It is for this reason that the result of a risk assessment, following the definition and design of the according control activities, is a matched set of control objectives, risks, and controls. Keep in mind that a control definition is usually described in terms of its policy and its procedure only. It is not separately mentioned in such a matched set of control objectives, risks, and controls.

There are several categorizations of control types, which are relevant to different levels and aspects of an enterprise. Theey can be located on high levels, as top level reviews of management activities, system software acquisitions, the system development methodology an enterprise uses to build its IT Systems etc. For a comprehensive list and discussion of control types, refer to COSO [COSO92].

A control can be *preventive* or *detective*:

Preventive Controls

Preventive controls are designed in order to avoid an unintended event or result. They exist to avoid the occurrence of potential problems. They can monitor operations and/or provide input to operations. One example of a preventive control relevant to the activity level risks of a company (as described in 3.1.2.2.2) is an inventory control system that predicts out-of-stock items or a credit authorization system that checks credit worthiness before goods are shipped. An example of a preventive control which addresses the entity-wide level risks of a company (see section 3.1.2.2.2) is a control which prevents a company from having unqualified personnel by requiring a certain qualification level in a job description. Further examples of preventive controls are:

- Segregate Duties (An example of this control type is provided in section 2.3)
- control access to physical facilities and information systems
- use well-designed documents (prevent errors)
- a cash budgeting system which monitors cash flows and forecasts of future cash flows
- an inventory control system that predicts out-of-stock items
- a credit authorization system that checks credit worthiness before goods are shipped.

Detective Controls

COSO defines a detective control as a control designed to discover an unintended event or result. It exists to uncover problems which could impede the achievement of control objectives soon after they arise, by measuring the relevant parameters of a process and indicating if they deviate from a given plan. Some typical examples of detective controls are:

- periodic performance reporting with variances
- standard costing and variances
- reconcile receivables, i.e. to check on a periodic basic that a company has received the expected money from its business partners
- periodic credit history review etc.

3.1.2.2.4 Information and Communication

A properly designed internal control should provide the management with information about its performance. For this reason COSO requires that a company establish separate procedures that will generate information about the state of internal controls and communicate this information to the appropriate managers.

This can be achieved with the help of information systems that produce reports containing operational, financial, and compliance-related information. They deal with internally generated data and information about external events, activities and conditions necessary for business decision-making and external reporting. Communication in a broader sense must also occur throughout the organization. Communication must occur from the top-down in order to inform employees about the existence of internal controls and their respective responsibilities. A bottom-up communication must occur to inform management about the state and effectiveness of the internal controls, in order to help management to take follow-up actions in case of deficiencies. Effective communication with external partners, such as customers, suppliers, regulators, and shareholders, is also necessary.

3.1.2.2.5 Monitoring

Internal controls change over time, and effective procedures may eventually become non effective. This may happen due to new personnel, changed business practices or a changed control environment. For this reasons, internal controls systems need to be monitored. This is a process which, over time, will help the assessment of the system's effectiveness and performance. This is accomplished by ongoing monitoring activities and separate evaluations of the internal controls system.

3.1.3 Relations between Internal Controls and Regulatory Compliance Requirements

We have introduced internal controls, with COSO as the relevant framework for realizing an internal controls system in an organization. The existence of an effective and effectively documented and implemented internal controls system is required by SOX 404. The reader may ask how regulatory compliance requirements in general are related to the internal controls system. What does having an effective internal controls system mean with respect to other regulatory requirements with which an organization must be compliant? Internal control is about enterprise risk management (ERM), thus only those regulatory requirements associated with ERM should be considered. In this sub-section, we identify the existing redundancy among some key regulations associated with ERM. With the identification of this overlap, the cost of

compliance can be reduced by simply eliminating the redundancy in the related compliance processes.

We will first briefly introduce those regulatory requirements associated to the ERM (other than SOX), which were selected for the analysis. The following list of regulations does not represent a complete list of regulations associated with ERM. These were selected according to the amount of attention they receive from the compliance experts:

- Basel II in Europe and its US counterpart: Inter-agency Operational Risk Supervisory Guidance on Operational Risk Advanced Measurement Approaches (AMA) for Regulatory Capital
- The Federal Deposit Insurance Corporation Improvement Act of 1991 (FDICIA)
- The Gramm-Leach-Bliley Act of 1999 (GLBA)
- Solvency II
- Grundsätze ordnungsmäßiger DV-gestützter Buchführungssysteme (GoBS)

We will then compare these partly overlapping regulatory requirements and discuss their common elements and requirements. Based on this, we will identify opportunities for organizations to construct a holistic approach to realizing enterprise-wide risk management, which reduces the cost and efforts necessary in order to become ERM compliant.

Basel II

Basel II [BaselII08] is intended by international bank regulatory authorities to promote enhanced risk management practices and to better align minimum regulatory capital requirements with the risk profile of a banking institution. Institutions qualified for Basel II will be allowed to use their own internal models for quantifying operational risks. Quantified operational risks will help to determine minimum regulatory capital. Qualification will be subject to regulatory review on a qualitative and quantitative level. Basel II is about applying state-of-the-art risk management of operational risk and internal controls systems and setting minimum capital requirements for banks.

FDICIA

The Federal Deposit Insurance Corporation Improvement Act of 1991(FDICIA) [FDICIA91] was one of the earliest efforts to promote a formal internal control discipline, with the process of attestation of the adequacy of internal controls. Under the annual audit and reporting requirements specified in FDICIA all insured depository institutions with $500 million or more in total assets are required to submit annual management assessments of their internal control structure and to obtain attestations of those assessments from their independent external auditor.

Banking institutions are the main focus of FDICIA, which mandated that federal banking and thrift supervisors pass specific regulations to establish standards for the safe and sound operation of a banking organization. As a result, specific regulatory definitions of effective internal control structures, and requirements for annual management review and board reporting were established. It made executive management, through their assertions, personally responsible for the internal control structure of their organizations.

Solvency II

Solvency II [Romeike et. al, 2006] is relevant for insurance companies that operate in the EU. Its purpose is to create an international standard and framework that insurance regulators can use when creating regulations about how much capital an insurance firm needs to put aside for unforeseen events. Solvency II prescribes three areas of requirements for insurance companies: i)

quantitative requirements, mainly regarding the amount of capital an insurer should hold, ii) requirements for the governance and risk management of insurers as well as for the effective supervision of insurers, and iii) disclosure and transparency of processes.

GoBS

Grundsätze ordnungsmäßiger DV-gestützter Buchführungssysteme (GoBS) [Philipp, 1998] is a German law and regulates the electronic processing of accounting and their underlying IT-systems. It can be considered as equivalent to SOX in Germany. It basically prescribes an internal controls system as a "component of process documentation" within a framework of IT-based accounting systems. It further regulates the retention requirements of the data and documents that are produced and consumed by electronic accounting systems and the rules applicable to security in accounting systems.

GLBA

The Gramm-Leach-Bliley Act of 1999 (GLBA) [GLBA99] imposes standards on how a financial institution's customer's private financial information may be shared among commonly owned businesses and with third parties. GLBA requires that banking regulators write safety and soundness standards for the safeguarding of customer information. It was used as an expansion of the original FDICIA safety and soundness regulation. According to GLBA, the (U.S.) banking agencies are responsible for establishing compatible regulations, and enforcing these in the banks they regulate. One main focus is the need for the ongoing assessment of the adequacy of internal risk management processes. GLBA auditors and regulators assess GLBA compliance on an enterprise-wide basis based on the management's responsibility for risk assessment processes, risk management systems, and risk controls.

3.1.3.1 Comparison and Discussion

Often an organization has to be compliant with more than one of these regulatory requirements. It is difficult to quantify and segregate costs according to each regulation. A close look at FDICIA, GLBA, GoBS, SOX, Solvency II, and Basel II / AMA requirements reveals certain common principles. They are [BITS05]:

- A greater emphasis on internal control systems and processes and their impact on operational risk;
- Extended requirements for risk assessment and its documentation and supporting evidence of sound systems of controls;
- The need for clearly defined roles and responsibilities regarding senior management's overseeing of internal control systems, with specific accountability and penalties for non-compliance directed at responsible individuals and entities;
- Concern for the accuracy and transparency of financial reporting and their related controls;
- An increased need for operational risk data collection and quantitative processes; and
- Better alignment of minimum regulatory capital requirements with the risk profiles of supervised institutions, specifically with regard to operational risk (and internal control systems).

The overlaps between these various sets of regulatory requirements can be drawn to show the convergence of the internal control certification and attestation processes. The main challenges are:

- Determining which business processes should be included in the evaluation of risks;
- Determining which controls are relevant;
- Documenting the design of the controls;
- Evaluating the design effectiveness of controls;
- Evaluating the operating effectiveness of controls;
- Determining which control deficiencies are of a magnitude great enough to constitute a material weakness;
- Documenting the results of the evaluation; and
- Communicating the findings to the auditory.

Thus it is obvious that bringing a higher level of support and automation to the above aspects will improve the introduction and maintenance of an internal controls system and consequently will improve regulatory compliance in area of ERM.

3.2 Business Process Management

Business process management (BPM) can be considered as a set of management activities related to business processes along a timeline. Thus business process management can be described by the set of these management activities that describe the life cycle of a business process. Business process management life cycle models have been proposed in among others, [van der Aalst et al, 2002a] [zur Muehlen, 2004] [Dumas et al, 2005]. In this thesis we follow a slightly extended life cycle model compared to the model proposed by [zur Muehlen, 2004]. In our case we see business process verification and mining (which will be described in sections 3.2.6 and 3.2.7) as separate stages. The reasons for this are: i) that our model, based on [zur Muehlen, 2004], reflects a consolidated view on business process management proposed by [Heilmann, 2005][Neumann et al, 2003][Galler et al, 1995][Striemer et al, 1995] ii) that in addition to activities in a business process model it also considers the resources that are consumed and the outputs that are produced by a business process and iii) that it is the way that most commercial and open source BPM solution providers realize their business processes with respect to the infrastructures for using those business processes. The life cycle comprises the management activities of analysis, design, implementation, execution, monitoring, and verification and validation. [zur Muehlen, 2004], in contrast, combines the two latter phases together and calls it evaluation.

3.2.1 Business Process Analysis

This phase is the first life cycle phase of business process management. In this phase, not only does one analyze how a process should work, one also analyzes and defines the roles involved in the operation of the process on an organizational structure level. The result of this phase is a set of business level requirements on the business process, which serves as input to the next BPM phase.

3.2.2 Business Process Design

This phase, also called Business Process Modeling, is about capturing the business level knowledge of a domain produced during business process analysis in one or more models. In our

understanding of business process design, we assume that the produced models are *executable*. Therefore a minimum requirement of the resulting description of the model is that it must be expressed in clear syntax in order to be interpretable and executable by an execution infrastructure.

3.2.3 Business Process Implementation

The proposed business process model produced during the previous phase is the input for the technical realization of the business process. This is related either to software development, if no pre-built business process infrastructure is used, or to business configuration (see section 1.2), if the software is deliverd by a standard software provider. In the latter case, the process model is used as a blueprint for the adaptation of an existing system in order to reflect the requirements formulated in the analysis phase and technically modeled in the design phase.

3.2.4 Business Process Execution

This phase, also called enactment, relates to the daily operations of the business process. Based on the implemented business process, business users enact the business process to fulfill their assigned function, be it to order goods, sell goods etc. In this phase, the business process infrastructure is used to handle individual cases covered by the business process. We call such a case, which is a grounded member instance of the business process model, a *business process instance*. This instance reflects individual information related only to that instance. We call the entity containing all individual information for a business process instance a *business process context*. A business process instance can unambiguously be identified by its context.

3.2.5 Business Process Monitoring

This phase is also called Business Activity Monitoring. Monitoring is a continuous activity that is performed with respect to individual business process instances. During this phase, each instance is tracked according to defined metrics. Business process monitoring can be used to obtain information about the current state of a business process instance. This can be used as basis for communication with business partners (e.g. a customer calls and requires information about his order) or it can be used to detect problems with a certain business process instance (e.g. a delivery has not reached the customer).

3.2.6 Business Process Mining

Business process mining is a relatively young discipline in business process management, which relies on business process monitoring. The aim of process mining is to analyze event logs extracted through business process monitoring. The extracted information can then be seen in comparison with the previously designed model. This allows process analysts to detect discrepancies, bottlenecks, or contradictions between the currently executed business process instance and its model. Another usage of business process mining is to discover process models based on event logs that were produced by information systems. The motivation behind the latter usage is that often in companies business processes exist without being formally designed or communicated. They exist implicitly because employees informally do their daily work supported by various information systems which then build the business processes. Mining techniques applied on event logs produced by information systems enables us to discover and determine explicit (formal) process models.

3.2.7 Business Process Verification

Verification addresses the correctness of a business process model. It focusses on two aspects: 1) that a model satisfies a set of properties given by a formula and 2) checking general properties of a model regarding its "syntactical" correctness. The first aspect is subject to model checking. The second aspect is related to determining issues which can exist in a process model design such as determining deadlocks or constellations in which a process execution of the given model would never terminate. We consider business process verification as a management activity on a process model during the design phase useful for determining in advance whether a process model exhibits (or does not exhibit) certain desirable behaviors. By performing this verification at design time the model can, based on the potential problems identified, be modified before it is executed. Technical verification of a process model greatly depends on the language, i.e. the formalism used to express the business process model during the business process design phase.

3.2.8 Business Process Validation

The task of business process validation is to check whether a process model works as designed. Compared to business process verification, it can be considered as a "higher level" check of a business process. The validition of a business process is a more difficult process than its verification. This is because verification requires a logical analysis of a process model, while the validation requires matching the behavior of a business process execution with the requirements formulated during the business process analysis phase. As a basis for validation of a business process, both business process monitoring and business process mining can be used. The core difference between verification and validation is that while verification results let the user know whether a business process model is designed correctly (its design accords to a specification), validation determines whether a business process works as designed (its execution is in accordance with its specification and design).

3.3 Interrelationship between Business Process Management and Internal Controls Compliance

Business process management carried out in the above phases is well recognized as a means to enforce corporate policies. Regulatory mandates also define policies and guidelines for business practice. One may question why a separate modeling facility in addition to those available in BPM is required to capture and enforce compliance requirements for business processes. We identify the following reasons:

- Firstly, the source of the control objectives and business objectives will be distinct with regard to ownership and governance as well to timeline. Whereas business objectives for business processes within businesses can be expected to have some similarities, control objectives will more often be dictated by external sources and at different times.
- Secondly, they have different concerns: business objectives and control objectives. Thus the use of business process techniques and languages to model control objectives may not provide a conceptually faithful separation of the two domains. Compliance is in essence a normative notion, and thus control objectives are fundamentally descriptive, i.e. indicating *what* needs to be done (in order to comply). There is evidence of some developments towards descriptive approaches for BPM, but these works were predominantly focused on achieving flexibility in business process execution (see e.g. [Hagerty, 2007] [Sartor, 2005]).

- Thirdly, there is likelihood of conflicts, inconsistencies, and redundancies within the two specifications. The intersection of the requirements and the modeling perspective needs to be carefully studied.

And why not model the controls during the business process design phase in accordance with the business process model produced with the help of the selected business process modeling language? Reducing business process compliance to the business process design phase has several drawbacks, from both a legal and a technical perspective:

A compliance project requires a certain approach from an auditing perspective, which defines a set of requirements on the IT system landscape of an enterprise. Actually, in the context of regulatory requirements such as SOX, the law requires that the internal controls on different entities in enterprises be *effectively* applied during the execution time of business processes. This means that the enterprises have to prove that their processes and internal controls work as planned in daily operations.

Therefore a pure design time based approach does not fully satisfy the requirements set by regulations such as SOX. Further, the manual embedding of all compliance requirements into the process models has several technical drawbacks:

- Firstly, the process models become too complicated, not readable and manageable when they are directly, i.e. manually enriched with the compliance controls.
- Secondly, since the compliance requirements on business processes are usually defined and implemented by different stakeholders in enterprises and they have different life cycles, they have to be separated from the original business logic of a process. This requirement becomes obvious when considering the fact that if a required control option of a business process is reset due to some operational reasons, such as faster transactional response time or similar, the business process will still function properly in terms of fulfilling its "business objectives", namely "purchasing goods" or "selling goods", but will no longer fulfill its "control objectives". For an example of such a situation, please refer to the different use cases introduced in section 2.2 and 2.3.
- Thirdly, from the perspective of a standard software provider, the shipped process models become less reusable for different customers if the compliance requirements are "hard coded" in the original process models. The customers act in different environments and have different compliance requirements for equivalent business processes (equivalent business objectives). Therefore, the compliance requirements have to be designed and provided separately.

Further, from a compliance auditing perspective, the compliance requirements are tested and certified as late as possible in the management life cycle of a business process by external auditors, namely during the execution phase of the business process – possibly by processing logs collected through monitoring techniques.

All of which serves to illustrate that a novel approach is required to 1) decouple the internal controls from the business process and 2) reduce the effort (especially the technical skills) necessary for setting/managing new control objectives according to their controls. The necessity for such an approach can also be practically clarified by referring again to the compliance use cases described in sections 2.2 and 2.3, where we saw the *orthogonal* nature of compliance compared to the business process (as concluded in section 2.5). As shown in those cases, the reasons we've taken such a compliance approach can be found in the nature of internal controls compliance. It forces us to define separate roles in a company (see section 3.1.1) and consequently separate technical layers for compliance.

However, the challenge inherent in such a technical layer is that on a representation level (namely in the business process design), we aim to conceptually separate the controls from the

business process, and on the level of their execution we aim to tightly integrate the controls with business process execution in order to support the control effectiveness at runtime. We must therefore strive to effectively separate and then coherently reintegrate each control, out of and then back into, its respective business process.

Detective Controls vs. Preventive Controls

There are several approaches which put the above BPM phases to some use, either alone or in combinations, in order to handle the internal controls compliance of business processes. This leads to different relationships between internal controls management and business process management, especially with regard to the life cycle phases. The core difference between the different approaches is whether they aim to support **detective** or **preventive** controls (see COSO, section 3.1.2).

One can use business process verification to check whether, during business process design, a process model that satisfies certain controls, (contains these controls) is produced. As discussed before, this is not sufficient from the auditing perspective. The reason is that with this strategy an enterprise only documents that it has designed the controls. It must also prove that the controls are effective while executing process instances. Thus the business process execution phase has to be taken into account as well.

In the area of detective controls there are currently two main approaches towards achieving post compliance. The first is *retrospective reporting*, wherein traditional audits are conducted for "after-the-fact" detection, often through manual checks by expensive consultants. A second and more recent approach is to provide some level of automation through *automated detection*. This approach may be supported by usage of process mining technologies [van der Aalst et al.,2005a]. The bulk of existing software solutions for compliance follows this approach. The proposed solutions hook into a variety of enterprise system components (e.g. SAP HR, LDAP Directory, Groupware etc.) and generate audit reports against hard-coded checks performed on the target system. These solutions often specialize in a certain class of checks, for example the widely supported checks that relate to Segregation of Duty violations in role management systems. However, this approach remains "after-the-fact" detection, which we call assuring "Post Compliance".

A major issue with the above approaches (in varying degrees of impact) is the lack of sustainability. Even with automated detection facilities the hard coded check repositories can quickly grow out of control making it extremely difficult to evolve and maintain them for changing legislatures and compliance requirements. In addition to external pressures, there are often internal pressures within a company towards quality of service initiatives for process improvement, which have similar requirements. The complexity of the situation is increased by the presence of dynamically changing collaborative processes shared with business partners. The diversity, scale, and complexity of compliance requirements warrant a highly systematic and well-grounded approach.

Figure 10 visually summarizes preventive controls vs. detective controls and puts them in relationship to the according stages in business process management life cycle as described in section 3.2.

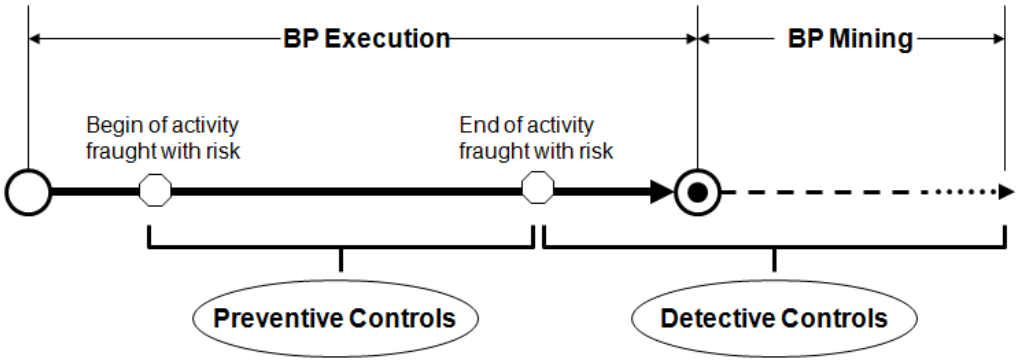

Figure 10 The position of preventive and detective controls in business processes

Based on the discussion above, we believe that a sustainable approach for achieving business process compliance should fundamentally have a **preventive** focus. As such, we envision an approach that provides the capacity to capture compliance requirements through a generic requirements modeling and enforcement framework, and subsequently facilitates the propagation of these requirements into business process models and enterprise applications, thus achieving *compliance by design*. The ideal is to have as many preventive controls as possible. The usage of formal approaches, beyond all others, would fulfill the role of capturing the conditions in which a detective control may be violated and to prevent a control violation from occurring. Of course one has keep in mind that the occurrence of a detective control violation may not necessarily represent the occurrence of a risk. For this reason a flexible strategy has to be developed in order to react to a control violation in a suitable way. This is important, since defining a general *blocking* of all business process instances causing a control violation may not be adequate.

Example 3.1:

Let us assume a detective control such as: *Check whether the bank account of a supplier has changed*.

Such a situation may or may not be a fraud situation, (an existing supplier may in fact have changed his bank account due to perfectly valid and legal reasons). Currently, the above detective control is realized in the following manner:

1. An accounting expert periodically collects a set of all suppliers from whom goods are ordered.
2. He subsequently checks whether the bank account information of each supplier has been changed.
3. He then investigates whether or not each supplier determined in step 2 exists.

Realizing such an approach is very inefficient and expensive due steps 1 and 2, for which all order entries and supplier entries must periodically be manually visited by the accounting expert. And in cases of fraud, the order has already been submitted and must be retroactively retrieved.

One can design such a detective control as a preventive control and execute the control automatically *before* an order has been submitted to a supplier. In order to do this, the following requirements must be realized:

A. Define the triggering event of a control as a model that can be automatically recognized by the system. In the example above such an event would be "*before the order is sent to that supplier (or even before it is approved)*".
B. Capture the conditions that could potentially represent a fraud in that model. The occurrence of such a condition could then be automatically recognized. In the case of the

above example such a condition would be the situation where "*the banking account of a supplier has changed"*

Providing the above-described requirements enables the original detective control to be realized as a preventive control. The efforts required for assuring internal controls compliance are reduced by eliminating the manual steps 1 and 2, which were necessary in the original detective approach.

In cases where the conditions of the preventive controls (see B) become true during the event-part of a preventive control (see A), the current business process instance could be blocked from continuing and a notification message for the accounting expert could be generated. The accounting expert would then continue with step 3.

3.4 Conclusion

In this section we introduced the basic concepts of internal controls and business process management. By looking at other relevant regulatory requirements in the area of enterprise risk management (ERM), we showed the common requirements for the existence of an effective, up and running, internal controls system.

As a consequence of those requirements, we provide novel methods and solutions for realizing effective internal controls on operative business processes. These methods rely on the fulfillment of the following requirements:

- a functioning "Control Environment" of COSO in an enterprise
- a guaranteed top-down "Information and Communication", meaning that the necessity and responsibility for internal controls is communicated by management to the employees
- a set of selected significant accounts, relevant business processes for those accounts, and assessed risks and controls proposed by compliance experts.

Given this, the methods and solutions provided in this thesis then address the following COSO components:

- Control activities: Design of controls on business processes
- Bottom-up Information and Communications: Issues and shortcomings are communicated to management and responsible roles in an enterprise
- Monitoring: The operations of business processes are monitored for effectiveness of the prescribed controls in daily operations.

We call a business process in such an environment compliant (Business Process Compliance) and mean: explicitly compliant to regulations associated with ERM.

We further discussed relevant phases of business process management with regard to internal controls. We discussed the difference between detective and preventive controls and how they are related to the business process life cycle. We argued that an effective and efficient strategy for achieving business process compliance should have a preventive nature, which supports compliance by design in a business process, whereas detective controls represent a post-compliance. The requirement for realizing a preventive nature of compliance is to capture the entities involved in internal controls in precise formal models.

4 Domain Model for Business Process Compliance

In the previous chapter, we introduced basic concepts that are used in business process compliance, namely internal controls and business process management. In this chapter we move from a rather informal discussion towards more precise modeling of business process compliance and the necessary artifacts for defining and achieving it. This chapter aims to provide a formal framework in the form of a set of modeling entities, based on which the controls on business processes will be define. The business process executions will then be validated according to those controls.

For this endeavor a top down approach is used. First an upper model for business process compliance is formally described by introducing some basic sets and the relations between them. In the next step we concentrate on a part of the upper model, namely the relationship between controls and business processes. We introduce the reference model that we propose in this thesis to ensure the effectiveness of controls in business processes. The reference model sets different phases in business process and internal controls management in relation to each other. These phases should support the effectiveness of controls in business processes. The reference model proposes to use a verification and validation approach to ensure the effectiveness of controls in business processes: Business process verification ensures that a business process model is deigned as required. Validation ensures that business processes work in daily operations as required by the controls through validating the business process executions.

In order to define controls on business processes, the notion of controlled entities will be introduced. They are the entities in a business process that are controlled by a control during execution time of business processes. The role and position of the controlled entities in a business process must be precisely captured. Controlled entities are further used as artifacts serving the modeling of a control. Chapter 6 will present a control model that can be designed on business processes building on top of the controlled entities described in this chapter. To support the readability of the formal descriptions and their relationships, an object-oriented approach is used in the endeavor to describe models: With the unified modeling language (UML) providing a modeling set of constructs there exists a language adequate for capturing the structural complexity of the models represented on a static level. We use basically the structural features of UML in terms of class diagrams.

In summary, the objectives of this chapter are three-fold:

1. To establish a reference model for supporting controls in business processes
2. To construct a formalized repository of business process models building on top of the controlled entities, on which a control will be designed.
3. To construct a formalized repository of business process instances building on top of the controlled entities, which must behave as required by the controls.

In section 4.1 we give a formal definition for the upper model of business process compliance. The definition is exemplified by the situation of the use case companies presented in chapter 2. In this section we introduce the interconnection of business process and internal controls management in terms ofthe reference model for supporting the effectiveness of controls. Section 4.2 introduces the controlled entities (CEs) in a business process. Here a model of business process, consisting of the controlled entities and their relationships to each other, is motivated and formally described. In section 4.2 we further develop the necessary artifacts that are required to assure the effectiveness of a control during execution of business processes (business process instance). We conclude this chapter by discussing the related work (section 4.3). Here we

compare the related research in the area of integrating risks in business processes (section 4.3.1). In sub-sections 4.3.2 and 4.3.3 a detailed discussion of current business process modeling approaches is provided, along with a comparison of these to our proposed model of business process capturing the controlled entities.

4.1 Formal Definition of Business Process Compliance

In the following, the logical relationships between the first class entities identified and exemplified by the scenario (see chapter 2) are captured. The core elements of the formal definition of the upper model for business process compliance are: a set of significant accounts (*ACCOUNTS*), a set of risks (*RISKS*), the set of relevant business processes (*BPS*) in a company and a set of controls (*CTLS*) on the business processes (see Figure 11). A system responsible for business process compliance must contain the given sets and implement the relationships between them as shown in figure 11.

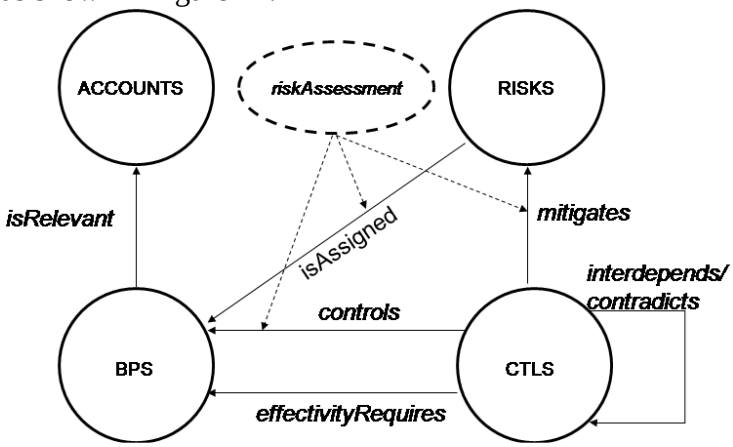

Figure 11 The involved entities and their relationships in business process compliance

Definition 4.1: Business Process Compliance Definition (BPCD)

A tuple *BPCD = (ACCOUNTS, BPS, RISKS, CTLS, isRelevant, controls, effectivityRequires, mitigates, isAssigned, riskAssessment, interdepends, contradicts)* is called the *Business Process Compliance Definition*, in which:

- *ACCOUNTS* is a set of significant accounts
- *BPS* is set of business processes
- *RISKS* is a set of risks
- *CTLS* is a set of controls
- $isRelevant \subseteq BPS \times ACCOUNTS$ is a relation that maps relevant business processes on siginificant accounts.
- *controls* is a total function $CTLS \rightarrow BPS$.
- $effectivityRequires \subseteq CTLS \times BPS$ is a relation between the set of controls and relevant business processes.
- $mitigates \subseteq CTLS \times RISKS$ is a total function
- $isAssigned \subseteq RISKS \times BPS$ is a relation between risks and relevant business processes.
- $riskAssessment \subseteq (BPS, 2^{RISKS}, CTLS)$ is a set of tuples of type *(bps, rk, ctl)*, where $bps \in BPS$, $rks \in 2^{RISKS}$ and $ctl \in CTLS$.

- *interdepends* $\subseteq CTLS \times CTLS$ is a relation, which identifies those controls depending on each other.
- *contradicts* $\subseteq CTLS \times CTLS$ is a relation, which identifies those controls contradicting each other.

The description of each of the relations in the above definition is as follows and they will be exemplified in next sub-section:

- *controls* delivers the set of controls that are required for a business processes
- *isRelevant* delivers relevant business processes on significant accounts
- *effectivityRequires* delivers business processes that are necessary for the effectiveness of a control
- *mitigates* delivers the risks for which a control exists.
- *isAssigned* shows the risks that have been identified for a business process
- *riskAssessment* is a relation that represents for each business process its risks and the controls mitigating them. This relation reflects the result of achieving COSO's Risk Assessment component (see section 3.1.2.2.2)
- *interdepends* is a relation between those controls that must be be defined together on a business process, because they depend on each other
- *contradicts* is a relation between those controls that are not allowed to be defined together on a business process, because they contradict each other.

The domain-specific-knowledge required for the internal controls compliance for business process reflected in BPCD (Definition 4.1) is determined by following sources:

- Analysis of non IT-related COSO framework as a de-facto standard for realizing the internal controls compliance recognized by regulation bodies and compliance/auditing experts

- Analysis of Accounting Standards of the Public Company Accounting Oversight Board (PCAOB) [PCAOB04], which has also ratified COSO.

- Participation in internal controls compliance projects.

4.1.1 Scenario Revisited

In the following each relation that occurred in BPCD (see Definition 4.1) is shown graphically using the two use cases of the scenario (see chapter 2).

The relation *isRelevant* in the case of CustomerA (see section 2.3) is shown in Figure 12. The procurement (in the case of CustomerA, the purchasing business process) and its sales business process are relevant for significant accounts *Inventory* and *Receivables*.

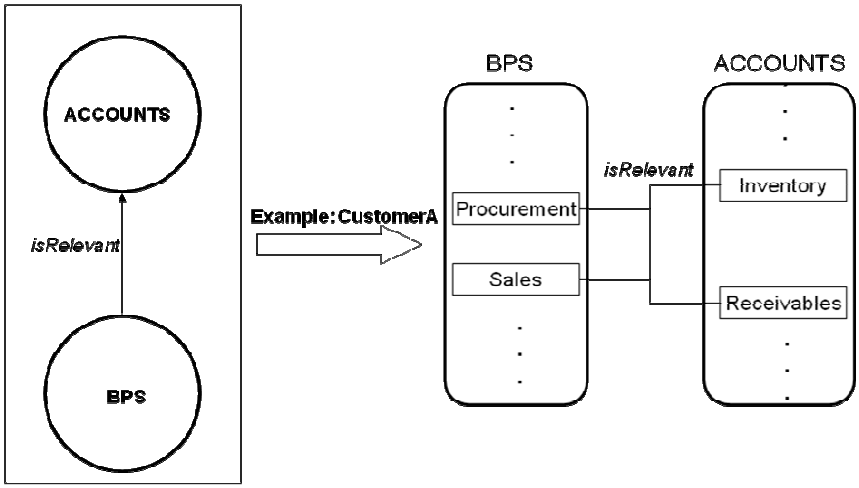

Figure 12 Example of Relation *isRelevant* in case of CutsomerA

Figure 13 exemplifies the function *controls* and the relation *effectivityRequires* in the case of CustomerA. Controls CA1, CA2, CA3 and CA4 (see section 2.3) are defined for the procurement of CustomerA (its purchasing business process), whereas the control CA4 requires RfQ-sub-process for its effectiveness (see section 2.2.5).

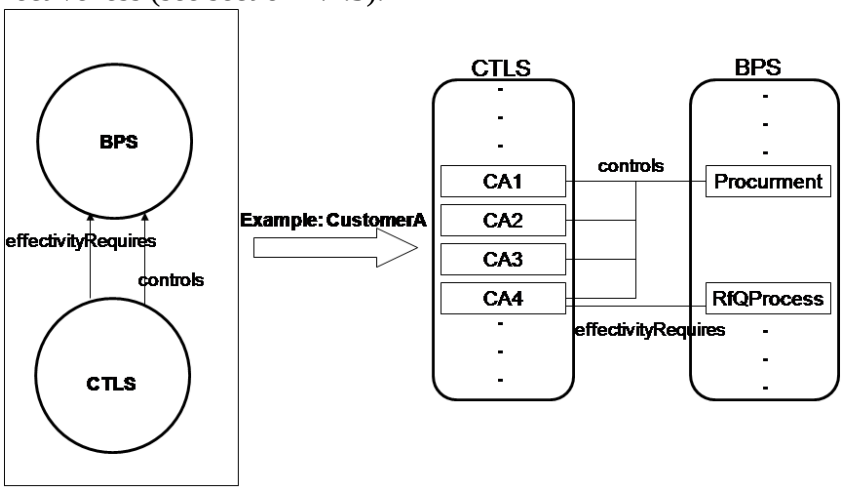

Figure 13 Function *controls* and relation *effectivityRequires* in case of CustomerA

Relation *interdepends* is exemplified in Figure 14 using the use case of CustomerA. Controls CA1 and CA2 depends on CA11 to be effective and CA11 depends on CA12.

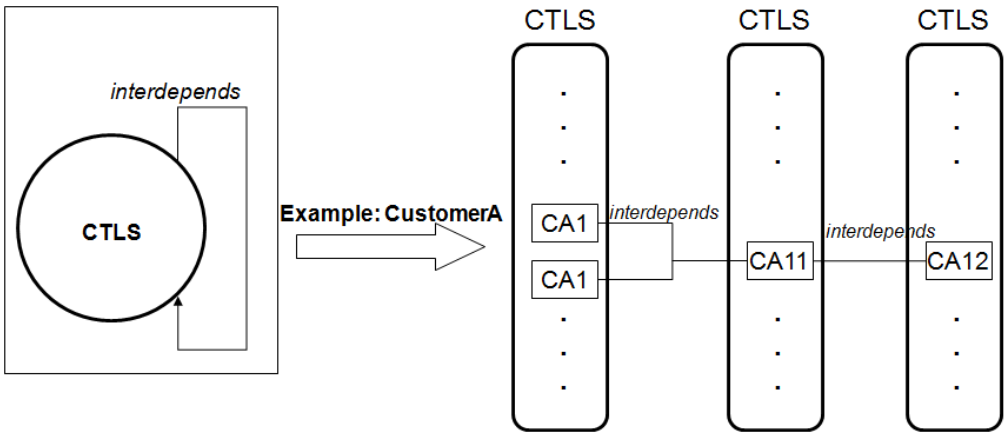

Figure 14 Relation *interdepends* for CustomerA

Figure 15 shows the risks that were identified in the purchasing business process of CustomerA.

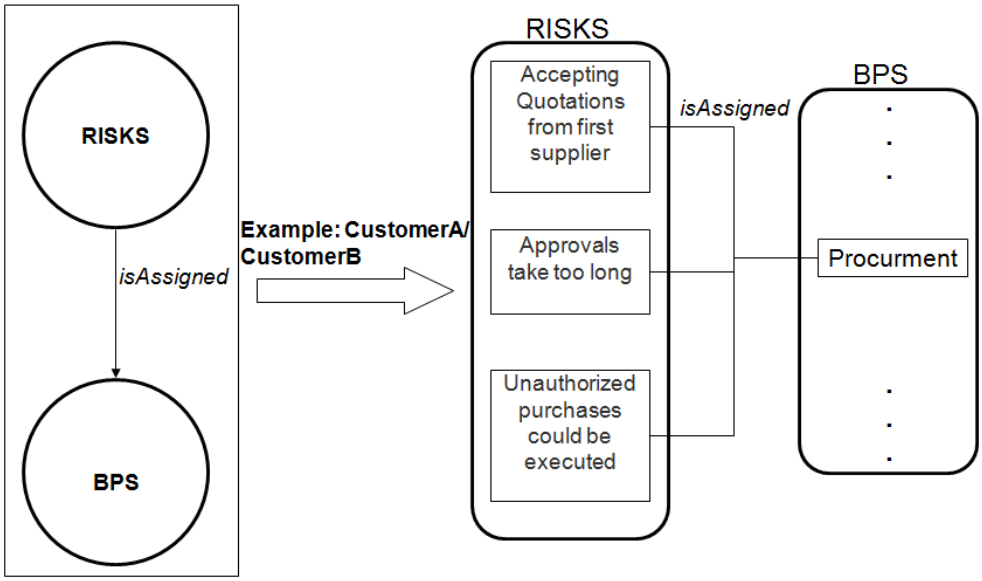

Figure 15 Relation *isAssigned*: Risks that are identified in the Procurement of CustomerA and CustomerB

Figure 16 exemplifies the relation *mitigates* for some of the controls that were defined for purchasing business process in case of CustomerA and CustomerB: In the case of CustomerA control CA3 mitigates the risk of "Accepting Quotations from first supplier" and CA4 mitigates the risk that "Approvals take too long". In the case of CustomerB controls CB1, CB2 and CB3 are all required to mitigate the risk that "Unauthorized purchases could be executed".

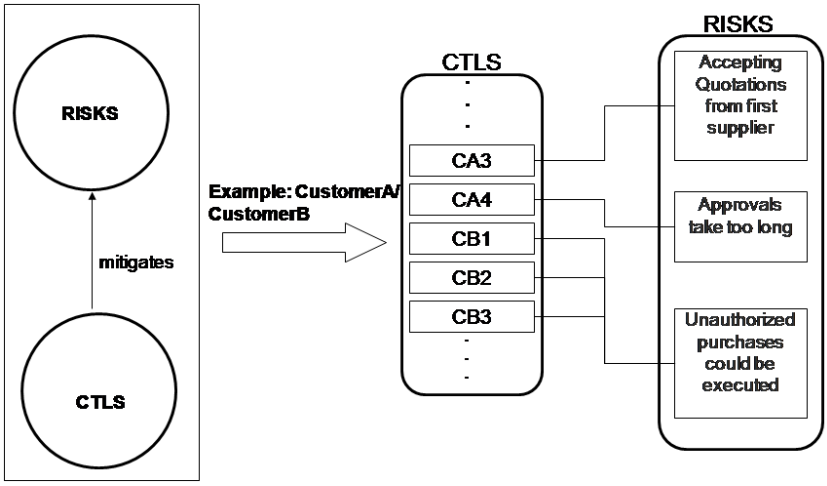

Figure 16 Relation *mitigates* in case of CustomerA and CustomerB

The exemplification of relation *riskAssessment* in the case of CustomerA is illustrated for reason of better readability in the form of a table instead of a figure, as was the case for previous relations. The table has the following form: The first column contains a list of relevant business processes, the second column contains for each entry in the first column a list of risks, and the third column contains a control for each entry in the first column. One row of the table is illustrated as an example in Table 7. A company must present such a table (with rows for all of its relevant business processes) to external auditors.

Table 7 Example of a Risk Assessment entry for CustomerA (*riskAssessment* relation)

Business Process	**Risks**	**Control**
Procurement	Accepting quotation from first supplier	CA3
	Risk of selecting low-quality goods	
	Risk of having only one supplier	

4.1.2 Selected Method for supporting the relation *controls*

We concentrate on the realization of the relation *controls* in BPCD (Definition 4.1) between business processes and the existing controls in a company. We propose a hybrid approach based on business process verification and validation.

Business process verification (see section 3.2.7) is used to check the process model according to a given business level specification. Since a pure design-time approach – as discussed in section 3.3 – is not sufficient, our approach supports the compliance validation of business processes executions. This is achieved by monitoring business process executions in order to check whether business process instances violate any previously designed controls.

We require the existence of the set *CTLS* (introduced in Definition 4.1), which means that the task of enterprise-specific interpretation of a regulation in the area of enterprise risk management (ERM) remains that of the compliance experts. Their domain-specific knowledge is necessary to interpret a regulation for an enterprise. Based on that interpretation (Determining of significant accounts, risk Assessment, etc.), the compliance expert proposes a set of necessary controls to mitigate the enterprise-specific risks. Based on the proposed set of controls by the compliance

expert, the business process instances will be visited and checked during execution time (see Figure 17).

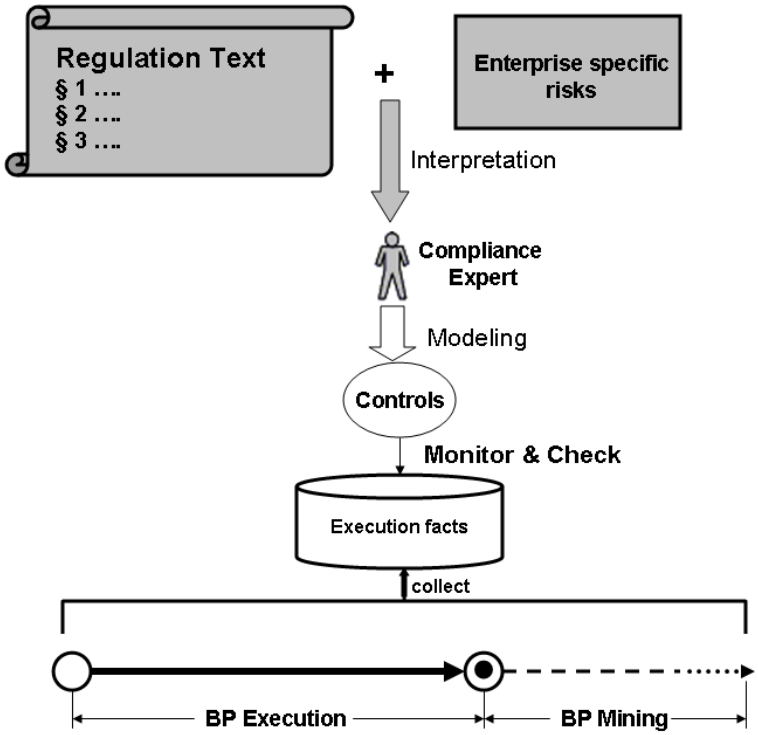

Figure 17 Compliance expert identifies and provides a set of controls

In Figure 18 our overall method as the interconnection between business process and internal controls management is presented. The bold arrows show the flow between different phases in business process and controls management. The dashed arrows represent the way the roles involved in business processes compliance are involved in different phases of business process and controls management. The two domains are formulated by different stakeholders and have different lifecycle phases. On the one hand the design of controls will impact the way a business process is executed. The test of controls carried out by control testers (under the supervision of compliance experts) may lead to (re)design of existing controls or to the definition of new controls. On the other hand, a (re)design of a business process causes an update of the risk assessment, which may lead to a new/updated set of controls. Additionally, business process monitoring will support the validation of a business process, which assesses the effectiveness of internal controls on business process executions. The result of the validation will serve as an input to internal controls certification.

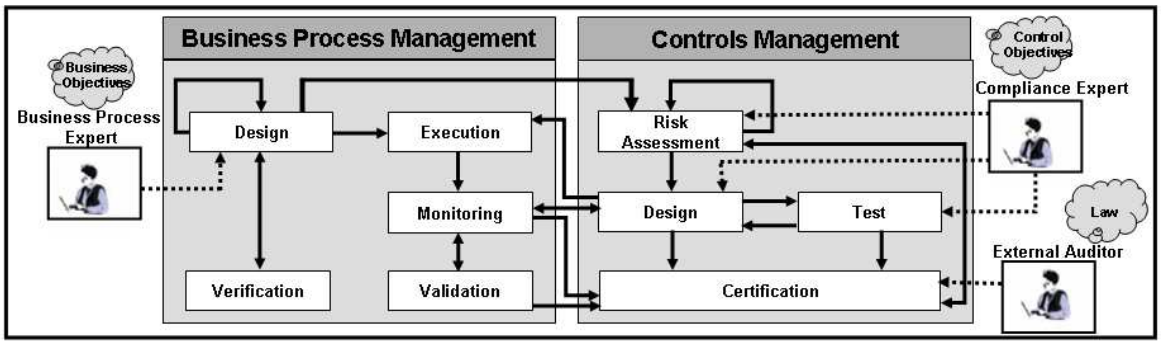

Figure 18 Selected Method: Verification and Validation of Business Processes

4.2 Controlled Entities in Business Processes

In this section we introduce the concept of controlled entities in a business process, formally define them and present their position in a business process that is subjected by a control.

A control influences different dimensions of the way business processes are enacted, namely:

- The execution order and occurrence of its activities
- The *Business Documents* involved (including their attributes such as amount etc.) and
- The *Users* including their *roles* performing any action in a business process.

Each of these dimensions contains entities that are subjected to controls. We call such an entity a *Controlled Entity (CE)*. In order to develop a clear understanding of the compliance requirements on business processes, we must create a precise model for the business processes and identify the position of the controlled entities within them. In focus are operative business processes such as Purchasing, Sales, Human Resource Management etc, which can be IT supported using workflow technology. In the following sub-sections such a model for business process is introduced and formally defined, based on its controlled entities a control will be designed.

4.2.1 First Class Entities in Business Processes

In order to provide a modeling approach for controls in business processes, a precise model of a business process, including the entities that can be used to design a business process together with the relationship between them, has to be developed. A control can then be modeled on top of such a model. Current approaches in support of business processes (their design, implementation and execution, see section 3.2) can be viewed from one of the following perspectives, upon which they focus:

- Activity-based perspective: tends to emphasize the activities in a business process as the dominant dimension. The activities produce, consume or transform information according to a set of rules
- Information-based perspective: emphasizes the information dimension by considering an activity in a business process as an operation that is triggered by a change of information

Most business process systems [Georgakopoulos et al, 1995] support a modeling approach with focuses on the first perspective, namely actions taken to achieve a certain "business objective" (also referred to as "activity-oriented" or "verb-centric"). Thus, in these cases a business process model must be described as a flow of activities. In real life business scenarios

the process model descriptions should also capture what is enacted on during enactment of a business process by describing the models of the *business documents* that matter to the business (for instance a Purchase Order document, an Insurance policy document etc).

Describing the process models purely by the business documents involved is called "document-oriented" or "noun-centric" and is related to the information-based perspective of workflow systems. This approach is mainly concerned with dependencies between data used by activities and deriving process flow based on such dependencies. The work in [Müller et al., 2006] showed that the available dependency information is usually insufficient for the generation of process models. It also showed that it could be difficult to determine the dependencies of a large number of data objects.

We believe that in order to be able to reflect real-life business scenarios as precise models, for which compliance requirements in terms of controls as formal models can be defined, a sensitive mix of both the document- and activity-oriented approaches is required. This can also be ascertained when studying the Hammer's Framework [Hammer, 2004], which employs *seven* dimensions to describe how work is coordinated to achieve operational and strategic business objectives. The notion of "work" mentioned in that work is in our point of view the operational business processes in enterprises. As we can see in Table 8 each dimension can clearly be assigned to the scope of *activities* or *business documents*. We have assigned the fourth dimension "who performs the work" to the scope of activities: the question raised in this dimension motivates us to ask which role the users play in a business process model and how their interrelationship to business documents and activities can be described. Our interpretation is as follows: Users enact activities on business documents. We will detail this relationship later in section 4.2.2.3.

Table 8 Hammer's View on work achieved reflected in business processes

Dimensions of Business Process	**Reflection**
Which results does the work deliver?	Business Document
Which information does the work require?	Business Document
How thoroughly is the work performed?	Business Document
Who performs the work?	Activity
Where is the work performed?	Activity
When is the work performed?	Activity
Which work is not performed?	Activity

Based on the mapped dimensions on business documents and activities proposed by Hammer we can see that neither a pure activity-flow nor document-centric approach for designing business processes is sufficient to describe the models of operative business processes. This is due the fact that both activities and business documents are reflected in different dimensions of work as presented by Hammer. Referring back to the scenario of purchasing business process (see section 2.1), the same fact can be recognized: business documents (such as Purchase Requests and Orders, Goods Receipts) together with activities (such as approving or rejecting them) were the entities that constituted the business process.

At the same time compliance requirements in business processes, in terms of internal controls, constrain the behavior of such entities in business processes. For example the presence of control CB2 at CustomerB (Segregation of Duties on Purchase Order Creations and Approvals with an

amount higher than 10000 $ for certain material types, see section 2.3.3) forced the business document *PurchaseOrder* and the activities related to its creation and approval in the purchasing business process behave in a special way at CustomerB, while the non-existence of the control CB2 at CustomerA leads to another behavior of those business documents and activities.

In the following sub-section a model satisfying the above discussion, called *Business Process Definition*, is presented. The controls will be formally defined in chapter 6 based on the entities in the business process definition.

4.2.2 Business Process Definition

In the following we introduce the *Business Process Definition (BPD)*. It represents the defined business model of an operative process in an enterprise. First we use the class-diagram of UML notation to present a model of BPD, then we successively detail its controlled entities relevant for business process compliance.

The model of *BPD* is represented in Figure 19. In our meta-model the first class entities in a BPD are *Activity, Role, Business Document (BD), and Transition.*

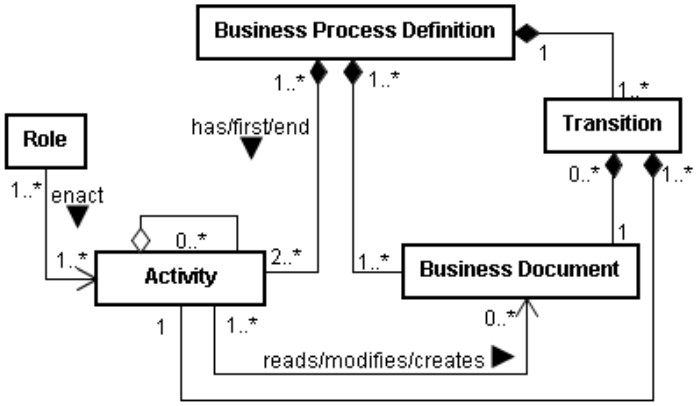

Figure 19 Business Process Definition (UML Notation)

A business process designed according to business process definition *BPD* consists of at least two activities (*Begin* and *End* denoting the beginning and the end of a business process) and one BD. Each activity and BD can be reused in different BPDs. An Activity has access to a certain set of BDs, where a BD must occur in the context of at least one Activity. Furthermore each Activity has a transition and a BD may also have a Transition. Each transition is unique to a BD or Activity. In a BPD there exist one or several roles. A *Role* enacts a set of activities inside a BPD. The high level description of the entities is as follows:

Business Document

A BD is a self-contained type representation of a unique business level entity, which is used in daily operations. BDs are processed in a business process in order to achieve a certain business goal. Beyond an internal structural model and a set of methods, which expose the functionality of the BD to the outside world (clients), a BD has a life cycle, which is described by its according transitions. As will be later described, the life cycle of a BD in terms of its transitions is unique inside a customer enterprise. From the point of view of a standard software provider, a BD can have different sets of transitions, which is due to the fact that the business models of customer

enterprises differ one from the other. Examples for typical BDs are “Purchase Order”, “Sales Order”, “Insurance claim” etc.

Further, a description of how these documents are processed by the transitions, i.e. which *states* they go through to achieve a specific “business objective”, is necessary. For instance, the roles involved in purchase order processing will update the purchase order business document in the course of the purchasing operations. The purchase order business document thereby has a life-cycle starting from its creation in some operational department, receipt by the purchasing department, possible its validation by the controlling department and upon delivery of goods, its closing and archiving. As we saw in our scenario, the design of such a business document can be too coarse-grained to be used on daily operations, which leads to dividing such a business document into several business documents with their own life cycles and attributes (such as Demand, Purchase Request, Purchase Order, Goods Receipt etc.).

Activity

An Activity represents significant business-level progress during the execution of a business process. One could also refer to an activity as a business process step. As mentioned before, BDs are processed, which means that an activity reads a set of BDs, may modify those BDs, and may create new instances of other BDs. This is achieved by invoking several methods of BDs inside an Activity. The relations between activities is designed by the transitions between them. Further, an Activity may be aggregated by other activities. In such a case an activity itself is a “sub-process”.

Role

A Role is a job function within the context of an enterprise which is designed into a business process.A Role is a job function within the context of an enterprise in which the business process (its definition) is designed. It has some associated knowledge captured about the authority and responsibility conferred on a group of persons assigned to the role. Being assigned to a Role in an enterprise means an approval to perform an activity that can affect one or more BDs.

Transition

A Transition is a modeling entity that describes the state change for a BD or the progress between activities in a business process. Given a set of activities and BDs in a BPD, there exist special coordinating transitions specifying which activities are to be executed on which BDs. The concept of transition is required for two reasons: i) it is used to describe the notion of “activity-flow” in a business process (also called “control-flow” in activity-oriented workflows) ii) it is used to capture the state change of a business document in a business process, i.e. the state-flow of a business document in a business process. An example of the state-flow of a business document captured by a transition would be a purchase order business document that is in *approved* state and later becomes *ordered*. In our conceptual model for business process, the flow of activities and business documents states are captured by transitions.

Based on the above entities a formal definition for BPD is given:

Definition 4.2: Business Process Definition (BPD)
A Business Process Definition is a tuple *BPD = (BDS, ACTIVITIES, ROLES, TRS, start, end, enacts reads, modifies, creates)* in which:

- *BDS* is a set of Business Documents
- ACTIVITIES is a set of activities over BDS

- *TRS* is a set of Transitions over *BDS* and *ACTIVITIES*
- *ROLES* is a set of roles
- *start* and *end* are each a total function from *BPS* (see Definition 4.1) to *ACTIVITIES: BPS* $\rightarrow$ *ACTIVITIES)*
- *enacts* $\subseteq$ *ROLES* $\times$ *ACTVITIES* is a relation between roles and activities
- Relations *reads, modifies, and creates* are each a subset of *ACTIVITIES* $\times$ *BDS.*

The definition of business process models according to BPD allows us to capture real life business process scenarios as they are found in industrial operative environments, because it contains several dimensions of work as described in Hammer's Framework (see section 0). The concept of BPD will be compared in detail to the related-work in section 4.3.2.

The controls will be defined on business process models and designed according to BPD (see Definition 4.2). Each of the entities occuring in Figure 20 is a *Controlled Entity* (CE) in a business process, where the relation between a *User* and the other relevant entities in a business process will be described in section 4.2.2.3.

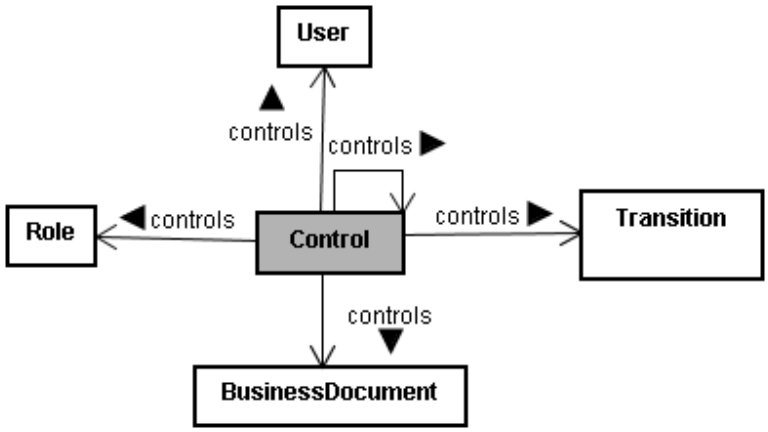

Figure 20 Controlled Entities

Definition 4.3: Controlled Entity (CE)
A Controlled Entity for a business process can be one of the following entities in a business process:

- *Transition*
- *Business Document*
- *User*
- *Role*
- *Control*

Thus it is important to develop a detailed understanding of each of the above entities that are subject to business process compliance. In the following sub-sections we are concerned with controlled entities of *transition, business document, role* and *user* types in BPD. A model of the controlled entity of type *Control* will be presented in detail in chapter 6. A *control* itself is considered as a controlled entity in a business process because if a control is not effective in a business process, i.e. its violation does not affect the business process executions, the enterprise runs the risk of being non-compliant. Thus the main task of compliance experts is not only to design the controls but also to assure their effectiveness. This situation was discussed in section

3.1.1 and recognized in our selected method for achieving business process compliance by separating the roles in compliance from that of business process expert (see Figure 18, section 4.1.2).

The required controls for business process compliance will be defined on business processes that are contained in a repository of business process models at a company. The definition of such a repository is straightforward:

Definition 4.4: Business Process Repository (BPR)
A business process repository *BPR* is a set of *BPDs*.

We conclude the introduction of BPD by a visualized example for the BPD of *Goods Receipt (GR) Processing*, as introduced in the scenario-chapter (see section 2.1.3).

Example 4.1: Goods Receipt Processing according to BPD
Goods Receipt Processing designed according to BPD is visualized in Figure 21. The figure contains 3 sets: *ROLES*, *ACTIVITIES* and *BDS* (see Definition 4.2). The relation between the elements of the sets (enacts, reads, etc. as they were introduced in Definition 4.2) is shown as arrows between the elements. In the figure, the roles are shown only at an organizational-unit level (Logistics *L*, Material Management *M and* Accounting *A*). A role can itself represent a role hierarchy, which represents the different roles inside an organizational unit. Further, only the relations: *enacts, creates, reads, and modifies* are represented. The relations: *start* and *end* are obvious and the interplay of *Transitions* will be formally defined later in sub-section 4.2.2.1.)

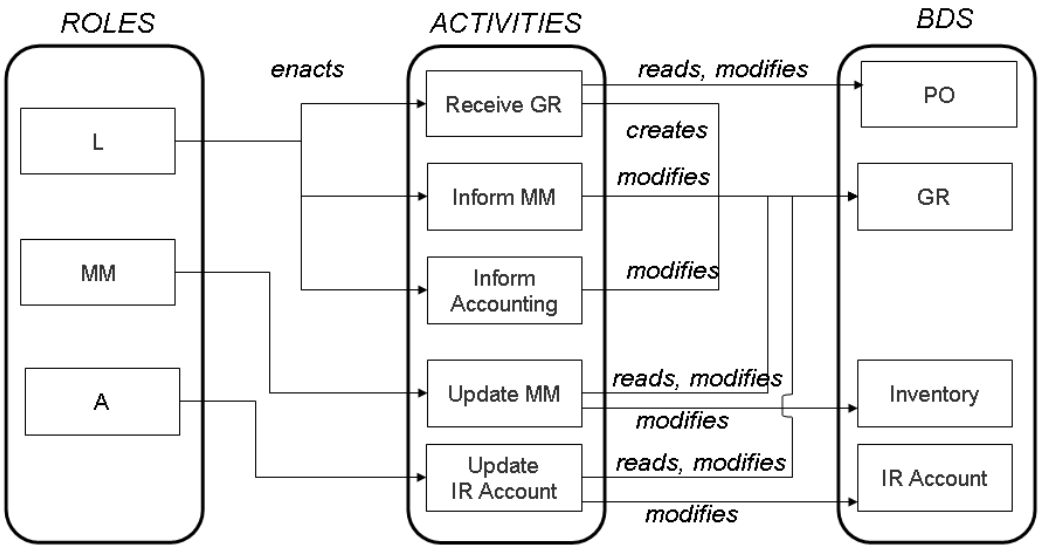

Figure 21 Visualization of the business process definition for Goods Receipt Processing

4.2.2.1 Business Documents, Activities and their Transitions

Below we discuss in detail the interrelationship between business documents and activities in our model, which together with their transitions build the key notions of a business process definition. An overview of the definitions in this sub-section is given in Figure 22. These entities and their interrelationships will be used in chapter 6 for the modeling of compliance requirements, i.e. control modeling.

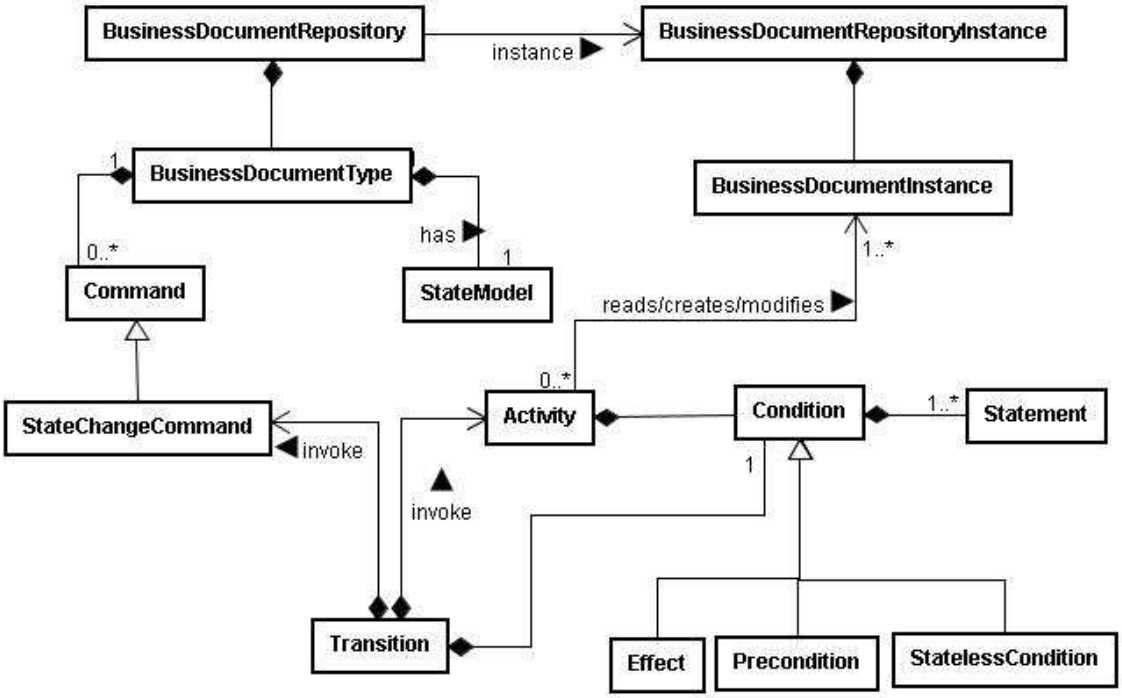

Figure 22 Model of Business Document, Activity and Transition (UML Notation)

We assume the existence of the following pair wise disjoint sets:

- *PCD* a set of primitive core data types
- *ACTVITIES* a set of activity names
- *BDSN* a set of business document type names
- *ATTRIBUTES* a set of attribute names
- *STATES* a set of business document states names
- *STATEVALUES, a* set of state values
- *IDS* is a set of unambigious identifiers of core data type *ID.*

The requirements for having pair wise disjoint sets listed above basically force the unambiguity of the business document names and activity names in a system, i.e. they assure that there exist no two business documents in which activities have the same name. The same applies to the attributes, states, their values, and the identifier of a business document.

In the following we define the state model of a business document:

Definition 4.5: State Model (SM)
A *State Model* is a tuple *SM = (S, SV, initialStateValue, finalStateValues, assignStateValues)*, with

- $S \subseteq STATES$
- $SV \subseteq 2^{STATEVALUES}$, is a finite pair wise disjoint set of *STATEVALUES* sets
- $initialStateValue \subseteq S \times SV$ is a relation that specifies for each $s \in S$ an unique initial state value
- $finalStateValues \subseteq S \times SV$ is a relation that specifies for each $s \in S$ a set of possible final state values

- *assignStateValues* $\subseteq S \times 2^{SV}$ maps each state name on a finite set of possible state values.

Example 4.2: An excerpt of the state model (SM) for the business document of type Purchase Order (PO) is visualized in Figure 23. It represents only an excerpt of the possible status names (state variables) and their values. A PO can be (among others) in states of *APPROVAL, ORDERING, POCNFM* (Purchase Order Conformation) etc. Its *APPROVAL* state for example may be not decided (*Not Approved or Awaiting*), *Rejected, Approved* etc. We will not attempt to provide a complete state model for PO, i.e. all its possible state names and their possible values.

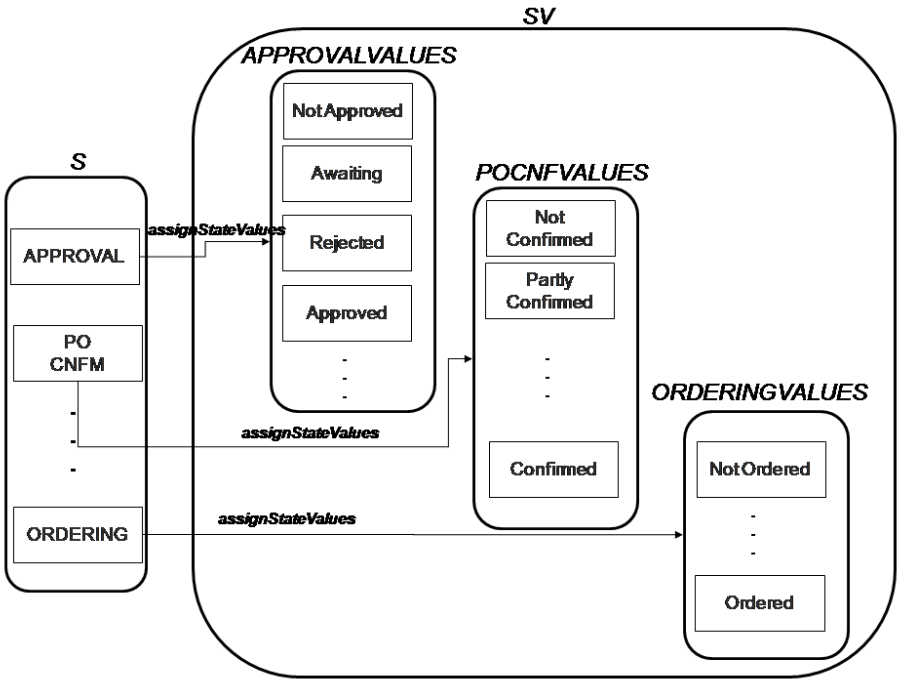

Figure 23 An excerpt of state model for purchase order business document (PO)

The controlled entity of business document type necessary for business process compliance is defined in following, where *ran (r)* represents the *range* of a relation *r* between sets *X* and *Y* with *ran (r) = {y: Y | ∃x: X ∧ r (x, y)}.*

Definition 4.6: Business Document Type (BDT)
A Business Document Type is a tuple *BDT = (bd, A, header, items, COMMANDS, type, SM)*, with

- *bd* $\in$ *BDSN*
- *A* $\subseteq$ *ATTRIBUTESRecovery ACTION*
- *header, items* $\subseteq$ *A*, where *header* $\neq \varnothing$
- *COMMANDS*, is a set of action names that can be invoked in the business document
- *type: A* $\times$ *PCD* $\cup$ *BDSN*, where *PCD* is the set of primitive core data types and *BDSN* is a set of business document type names. We say that a business document type *bdt'* is referenced in business document type *bdt*, if *bdt'* $\in$ *ran (type)* $\wedge$ *type(a)* $\wedge$ *a* $\in$ *A.*
- *SM* is a state model

The definition above specifies a business document type with an unambiguous name *bd*, a set of attributes, a state model *SM*, and a set of actions (*COMMANDS*) that a business document provides. A command belonging to *COMMANDS* of a business document type represents a possible life-cycle action specific to that business document type that can be performed on that business document. An example of a command possible for a business document of type *Invoice* would be *Confirm Invoice* or a command *Check PO* for a business document of type PO. The *header* and *items* attributes refer to two special sets of attributes, which are integrated in the definition to emphasize the necessity of their being a first class entity for business document types. A *header* contains additional meta-data information about the business document that further specifies its business document type. *Items*, if existent, is a collection of attributes which *reference* other business documents and core data types in the system. This is achieved through the introduction of the relation *type*, which relates each attribute of the business document (including its *header* and possible *items*) to any data types in the system. By doing so, the specification allows to flexibly build new business documents based on an already existing set of core data types and other business document types in the system. This is basically following the paradigm of Object-Oriented-Design (OOD) and the composition and aggregation mechanisms therein. The life cycle of a business document is specified by the state model *SM* it may go through. The *Transitions* of a business document will build on top of this state model, which we will elaborate upon later. As we will see, the states of a business document reflect the steps a business document may undergo (in an enterprise) during a business process, (eg. a purchase order may be: created, approved, declined etc.).

Instances of business document types can be created during execution of business processes, which are defined as follows:

Definition 4.7: Business Document Instance (BDI)
A Business Document Instance (*BDI)* of a *BDT* (as defined in Definition 4.6) is a triple $BDI = (id, CURRENTSTATES, value)$, with

- $id \in IDS$
- $CURRENTSTATES \subseteq STATEVALUES$ is the set of the current values for each state $s \in S$ of the according business document type
- $value: (A \times IDS) \times RI$ is a partial function, which assigns each attribute in A (including the *header* and *items*) to an element of the possible values (instances) in $BDSN \cup PCD$ (*RI* is defined in Definition 4.9*)*

The definition of a business document instance captures the notion of "object instantiation" as known from the OO-Paradigm. An instantiation of a business document type generates an unambiguous instance of that business document (given through its $id \in IDS$). The attributes in the business document types may have a value (*value* is a partial function) and a business document instance has a set of states and a certain progress of the business document life cycle in a business process.

Notation: By $value(id, a) = b$ we denote the value b of an attribute $a \in A$ referenced in the according business document type with an instance identification *id.*

Discussion: The definition above intentionally leaves open the question of how exactly to instantiate a certain business document type. It is purposefully not specified which attributes must have a value after the instantiation. In this respect we must remain abstract because we consider

the instantiation procedure of a business document type to depend on the type of business being discussed. We leave it to the implementation level of a business process (see section 3.2.3), because of two reasons: Firstly, the instantiation of a business document type (for instance a purchase order) will be different from the instantiation of a business document type goods receipt (GR) especially with regard to which attributes of those business documents must have a value after the instantiation. Secondly, and closely related, is the fact that the instantiation (the set of required assigned values) of one same business document type is enterprise-specific. It is for this reason that a standard software provider (Deliverer of Business Document type repository) will include as many attributes as possible/known into the specification of a concrete business document type without further specifying its concrete implementation with regard to instantiation. Instantiation will then happen at the customer enterprise: this is called Customization or Business Configuration. Such a strategy enables a standard software provider to serve different customers with the same set of business document types. As an example, remember the modified Control CB41 of CustomerB from the scenario chapter: the requirement for the technical realization of this control was that the *GR* Business document type must contain an additional attribute *order-number* copied from the according *PO*. This attribute was not necessary for CustomerA's model of the business document type GR since the control was not necessary for CustomerA. The EAVendor, aware of the different customer requirements, would include this attribute in the business document type GR without specifying it as mandatory, in order to keep the Business Document type as flexible and reusable as possible at different (customer) enterprises. One could argue that the introduction of identifiers "optional" and "mandatory" in the specification of the partial function *value* would be necessary, but we made the decision to stay generic at this stage of the specification in order to remain as reusable as possible on the conceptual level.

Example 4.3: Business Document Instance of Purchase Order (PO)
Figure 24 partly illustrates an instance of the business document type *PurchaseOrder*.

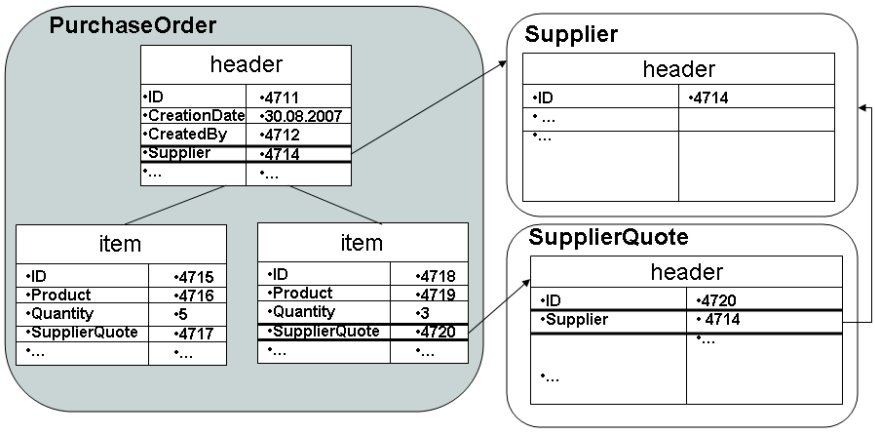

Figure 24 An instantiation of business document type purchase order

The notion of business document repository has been mentioned. Such a repository is normally provided by a standard software provider (such as EAVendor) to its customers. The customers can select business document types from such a repository and implement their enterprise-specific instantiation of the business document types in their business processes. The formal definition of such a repository is as follows:

Definition 4.8: Business Document Repository (R)
A *business document repository* is a finite set *R* of business document types, such that

$\forall bdt, bdt': BDT, att \in A \wedge type(att, bdt') \Rightarrow bdt \in R \wedge bdt' \in R$

The definition should assure that every business document type referenced in *R* also occurs in *R*. This property assures that a customer implements its business processes on top of a predefined set of business document types. During the execution of business processes, a set of business document instances are instantiated and available to be processed. They are contained in a business document repository instance, which is defined below.

Definition 4.9: Business Document Repository Instance (RI)
A business Document repository instance *RI* of a repository *R* is a mapping *instance* that assigns each business document type in *R* a finite set of business document instances.

An instance RI of repository R represents the set of business document instances which currently exist (already created, modified, or can be read by activities).

Example 4.4: Business Document Repository and its instance
Consider a simplified set of business documents involved in the purchasing process (see Figure 5 in chapter 2). The *R* (see Definition 4.8), *RI* and *instance* (see Definition 4.9), the function *type* of the business documents (see Definition 4.6) and *value*-function of the business document instances (see Definition 4.7) of such a model are visualized in Figure 25.

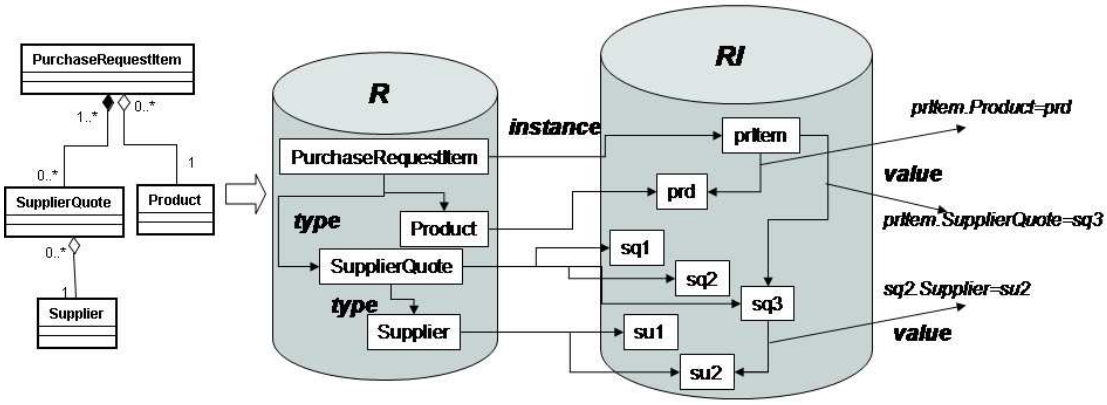

Figure 25 Example of a Business Document Repository (R) and its instance(RI)

The notion of statements defined below is introduced in order to enable us to define an activity. An activity is not only specified through the business documents it consumes (creates, reads, or modifies), but an activity definition also specifies which conditions have to be satisfied before an activity can be enacted. Further with the help of statements we design possible effects of an activity (on a set of business document instances). In this way it becomes possible to specify the flow of activities in a business process.

Definition 4.10: Statement
A *statement* on a *repository instance RI* of a *repository R* is one of the following:

- a predicate *ASSIGNED(bdi, a)*, which returns *TRUE* if for the business document instance *bdi* the *value* of its attribute *a* returns a business document instance in *RI*. Formally:

$\forall bdt: BDT, bdi:BDI, a \in A \subseteq ATTRIBUTES \wedge instance(bdt, bdi)$
$\wedge ASSIGNED(bdi, a)$
$\Rightarrow value(bdi, a) \in RI \wedge a \in RI$

- A predicate *EQUALS (bdi, a, b)*, which returns *TRUE*, if *ASSIGNED (bdi, a)* is *TRUE* and *value (bdi, a) = b*.
- *GREATER, GREATER_EQUALS, SMALLER, SMALLER_EQUALS* are each predicates which take following parameters as input *(bdi, a, b)* and return *TRUE*, if the predicate *ASSIGNED (bdi, a)* is *TRUE* and *a* is greater (or equals) respectively smaller (or equals) than *b*.
- predicate *NEW(bdi,a)*, which returns *TRUE* if for the business document instance *bdi* the value of its attribute *a* returns a business document instance not in *RI*. Formally:
  $\forall bdt: BDT, bdi:BDI, a \in A \subseteq ATTRIBUTES \wedge instance(bdt, bdi)$
  $\wedge ASSIGNED(bdi, a)$
  $\Rightarrow value(bdi, a) \notin RI \wedge a \in RI$

- predicate *STATE(bdi sv)* (state statement) returns *TRUE*, if the current state of a business document instance *bdi* has the state value *sv*, where $STATE \in S$ *and* $sv \in STATEVALUE$. *S* is a set of states belonging to a business document type $bdt \in R$.

The business document instance *bdi* is *initial* if for all state values *CURRENTSTATES* of that instance it is *TRUE* they are in an initial state and there exists some attribute *att* of a *bdi* for which the predicate *NEW(bdi, att)* returns true.

Definition 4.11: Condition
A *condition* is a conjunction or disjunction of *statements* (Definition 4.10) and *negated statements.* An *effect* is a conjunction of *stateless statements*, i.e. it contains no statement of the form *STATE (bdi, sv)*. A *precondition* is a conjunction of stateless statements and without any statement of the form *NEW(bdi, att)*.

We now introduce the formal definition of an activity:

Definition 4.12: Activity
An activity over a repository instance RI is a tuple activity = (name, BDI_{read}, $BDI_{\mathrm{mod}\,ify}$, P, E_{create}, $E_{\mathrm{mod}\,ify}$), such that:

- *name* $\in$ *ACTIVITIES* is the activity name
- $BDI_{read} \subseteq RI$ is a set of business document instances to be read
- $BDI_{\mathrm{mod}\,ify} \subseteq RI$ is a set of business document instances to be modified, with $BDI_{read} \subseteq BDI_{\mathrm{mod}\,ify}$ and $BDI_{\mathrm{mod}\,ify} = \emptyset$ allowed
- P is a precondition
- E_{create} is an effect, which contains only non-negated NEW-statements
- $E_{\mathrm{mod}\,ify}$ is an effect, which contains negated or non-negated ASSIGNED-statements.

An *activity modifies RI*, if

- $BDI_{\mathrm{mod}\,ify} \neq \emptyset$ and

- there exists a set *m* of attributes of a business document instance *bdi*, for which the attribute *att* the statement *ASSIGNED(bdi,a)* occurs in $E_{\mathrm{mod}\,ify}$ of the *activity* and
- for those attributes *m* the values of some attributes *att* in *m* are modified after the activity is enacted.

An *activity creates* a business document instance *bdi* in *RI*, if

- $bdi \notin BDI_{read}$ and
- *NEW (bdi',a)* is a statement in E_{create} with *value (bdi', a) = bdi* and
- *ASSIGNED(bdi, a)* is allowed in $E_{\mathrm{mod}\,ify}$.

The introduction of two different effects (*create* and *modify*) allows an activity to create a business document instance and at the same time to modify it.

Example 4.5: Activity *Receive GR*
(Notation: *varName: att* stores the current instance of *att* in attVar. attVar can then be reused in another place in the activity specification.)

***Activity** Receive GR*

- *name = "Receive GR"*
- BDI_{read} *={po:PO}*
- $BDI_{\mathrm{mod}\,ify}$ *= {po:PO}*
- *P = ¬ASSIGNED(po,GR) ∧ ASSIGNED (po, Supplier)*
- E_{create} *= NEW(po,GR)*
- $E_{\mathrm{mod}\,ify}$ *= ASSIGNED (po,GR)*

Description: The activity has the name "*Receive GR*". It reads the business document instance *po* of type *PO*. It reads this business document to determine the necessary information regarding the order. The activity modifies that *po* (PO is element of *BDI modify set*). There are two *statements* (simplified) that have to be fulfilled as precondition *P* of this activity: 1) No "Goods receipts" exist already for that order (the attribute *GR* of *po* is not assigned and 2) a supplier does in fact exist for that order (the attribute *Supplier* of *po* is assigned). The activity generates an *initial GR* business document instance for the order by specifying *NEW (po, GR)* in the create effect. Further the activity modifies the order in such a way that the order will contain an according Goods Receipt (*GR* Attribute of it is assigned). The specification leaves open the implementation of the business document instance creation (in this case GR), i.e. which attributes may be assigned (See definition of *initial* business document instance in Definition 4.7).

Definition 4.13: State Change Command (SCC)
A *State Change Command* is a command belonging to the set *COMMANDS* of a business document type (according to Definition 4.6) which changes the current value of a state name belonging to a business document type. Formally, state change command for a business document type is a tuple *scc = (bdi, cn, F)* with

- *bdi* being the business document instance on which the command has to be invoked,
- *cn* ∈ *COMMANDS* specifying the command name, and
- *F* being a non-negated state statement (as specified in Definition 4.10).

Based on the state change commands and activities we now define the transitions of business documents and activities that were used in Definition 4.2.

Definition 4.14: Transition
Given a set of activities *ACTIVITIES* over a repository instance *RI*, a *transition* is an expression with one of the following forms:

- *If c then* ***invoke*** *act*
- *If c then* ***invoke*** *scc*

where *c* is a condition of statements over *RI*, *act* $\in$ *ACTIVITIES* is an activity over *RI*, and *scc* is a state change command for a business document instance in *RI*.

Using the transitions it is possible to describe i) the progress of a business process in terms of the activities which have to be executed and ii) the life cycle of the business documents involved in a business process.

Example 4.6: Transitions in Purchase Order Processing
Below (see Figure 26) we give two examples of the different types of transitions that occur in the purchasing business process, namely in its Purchase Order Processing (see section 2.1.2).

In the first example (*trs*), the transition specifies that the activity *Send PO* will be invoked if the *approval* status of an instance of the purchase order business document (*po*) is "*Approved*" and an according purchase request for that *po* exists (the attribute *PurchaseRequest* is *assigned* in the *po* instance).

In the second transition (*trs'*) it is specified that if a purchase order instance *po* has already been sent to a supplier (the activity *Send PO* has *assigned* the attribute *Supplier* of the *po*) and the *APPROVAL* state of the *po* is already *Approved*, then the *ORDERING* state of the *po* will be set to *Ordered*. The transitions *trs* and *trs'* are as followed:

trs: if *APPROVAL(po, Approved)* $\wedge$ *ASSIGNED(po, PurchaseRequest)* ***then invoke*** *"Send PO" activity*

trs': if *ASSIGNED(po,Supplier)* $\wedge$ *APPROVAL(po, Approved)* ***then invoke*** *scc = (po,release, F),*
where *release* is the *name* of the state change command and *F = ORDERING (po, Ordered);*

The two transitions in the example above are visualized in Figure 26.

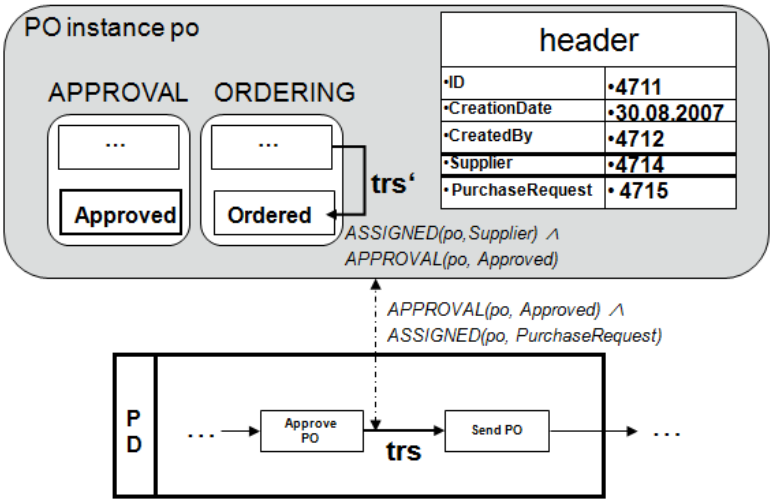

Figure 26 Two Examples for Transitions (trs and trs') in the Purchasing Business Process

The *activities* and *commands* that can be performed on a business document are herein described as transitions between the activities and the status changes of business documents, instead of a pure activity-flow-oriented definition of the process description as is the case in most workflow approaches. The modeling approach selected in this thesis was chosen due to the fact that with this model, the user has a variety of possible alternative activities at every stage in the process's progress. Upon choosing an activity to perform, the status of the business document may be changed and another set of activities may become available, depending on the preconditions of the activities available and the new state values of the business document states. Modeling such a behavior in a pure activity-flow-oriented approach would lead to a process model graph so vast as to be impossible to handle in most real life business processes. Modeling the state model of business documents in UML State diagrams has two disadvantages: UML State diagrams are not executable and the activity-flow aspect cannot be expressed via UML State diagrams.

The *activities* and *state change command (scc)* invoked in a *transition* are mostly performed by users in a business process. This is related to the notion of business tasks that appear in the task inbox of a business user, and that have to be processed by that user. For instance a user can accept a task, reject it, or assign it to another user etc. One could consider the transition as having an execution life cycle in terms of a state diagram as well. We see the status model for workflow steps done in [Casati et al., 1999] as suitable to be reused in our transition model.

4.2.2.2 An Execution of Business Process Definition: Business Process Instance

In this section we are concerned with constructing a model of business process instance. A business process instance represents an execution of a business process model designed according to BPD (see Definition 4.2). The business process instances are collected in a repository called business process repository instance (BPRI). The validation of business process compliance, i.e. the effectiveness of controls in business processes, will be checked on the process instances stored in a BPRI. The controls have been previously defined for business process definitions contained in a BPR (see section 4.2.2, Definition 4.4).

In the following we introduce the definition of the entities required to build the repository of business process instances. Figure 27 gives an overview of these entities.

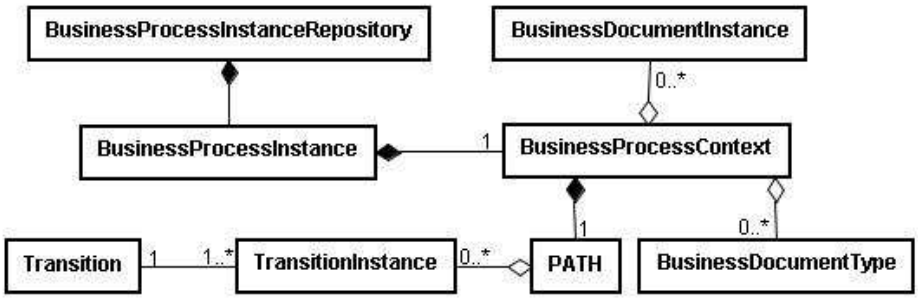

Figure 27 Entities related to Business Process Instance

The description and specification of each of the entities occurring in Figure 27 is as follows:

Upon the invocation of an *activity* or a *scc* in a *transition* (see Definition 4.14) a *transition instance* will be produced. A transition instance is an already *invoked* transition. All transition instances have some common attributes, which help to determine the current state of a process instance. These common attributes are specified for a *transition instance* by the following definition:

Definition 4.15: Transition Instance
A *transition instance* is a tuple *trs = (id, usr, rle, start, end, ref)* in which:

- *id* $\in$ *IDS* identifies the current instance of a transition
- *usr* $\in$ *USERS* is the user, who has caused the invocation of the transition
- *rle* $\in$ *ROLES* is the role of that user
- *start* and *end* are each of type *DATE* $\in$ *PCD* (primitive core data types), which is a core data type. Respectively, they identify the beginning and the end of the time that the current transition instance took to complete
- *ref* $\in$ *IDS* refers to the transition type of the current transition instance.

All that remains is to specify the execution of a BPD itself, a business process instance. Based on a business process definition according instances can be created, started and executed. The execution then relates to fulfilling the business objective of a business process. Although a business process instance adheres to a defined business process definition, different business process instances can vary according to the different transitions taken during their execution and the actors involved in fulfilling the business objective for which the business process exists. As explained in section 3.2.4, the state of a business process instance is reflected in the business process context. The context of a business process instance acts as a container for all consumed and produced information (in terms of business document instances) and the execution path of a process instance. Before introducing the formal definition of a business process instance, we formally introduce its context:

Definition 4.16: Business Process Context

The *Context* of a business process instance *bpi* is tuple *ctx* = (*id*, $2^{(BDSN \times RI)}$, *PATH*) where:

- *id* $\in$ *IDS* identifies the instance of a business process, to which the current context is assigned

- $2^{(BDSN \times RI)}$ is a set of business documents type names and their current instantiation in *bpi*
- *PATH* is a set *of transition instances*, which have so far been invoked during the execution of the *bpi*.

The second parameter in the tuple of Definition 4.16 acts as a set of key-value-pairs consisting of business document type names $bd \in BDSN$ and the according business documents instance *bdi* in *bpi*. The current value of a certain business document instance can be acquired through its key-name from the *context* (its second parameter) of a business process instance. Using the context a business process instance can now be defined as the following:

Definition 4.17: Business Process Instance
A *business process instance* of a business process definition *bpd* is a *tuple bpi = (id, start, end, bpdId, ctrs, ctx)*, in which

- $id \in IDS$ identifies the current instance of a business process
- *start* and *end* are each of type $Date \in PCD$. They respectively identify the beginning and, in the case that the instance has been completed, the end time of the current instance.
- $bpdId \in IDS$ identifies the corresponding business process definition of the current business process instance
- $ctrs \in IDS$ refers to a transition. It marks the current position of the business process instance.
- *ctx* is the context of business process instance.

It should be noted that a *ctrs* refers to a *transition* and *not* a *transition instance*. This implies that the according *activity* or *scc* in the transition is still not invoked. Further, using the information provided in a business process definition (including its transitions), it is possible to calculate the possible set of transitions that a business process instance could take as the next transition (next progress step in a business process instance which is at the position marked by *ctrs)*. In addition, the information provided by context *ctx* of a business process instance *bpi* enables the complete determination of the current business document instances (i.e. their current states and the values of their attributes) involved (produced and consumed) in the course of the execution of a business process.

A collection of business process instances is a *Business Process Instance Repository*. Its definition is now straightforward:

Definition 5.18: Business Process Instance Repository (BPRI)
A business process instance repository *BPRI* is a set of business process instances.

Controls will be designed on the elements in BPR (see Definition 4.4) and the effectiveness of these controls will be assured in daily operations based on the elements in BPRI according to Definition 4.17.

4.2.2.3 Role-Based Access to Business Documents and Activities

As we saw in the scenario, some of the defined controls (controls CA4, see section 2.2.3, CB1, CB2 and CB3, see section 2.3.3) were related to users (employees) and their roles in the use case companies. We have also identified (in section 4.2.2) the entities *User* and *Role* as controlled

entity (see Definition 4.3) types in business processes. In the current section we are concerned with exposing these types of controlled entities occurring in business process compliance, and exposing their relationship to business documents and activities. The model proposed serves as an underlying meta-model to define controls on business processes which involve users and their roles in their control definition.

In Figure 19, we only sketched the accessing of the activities in a business process, which is represented through an *enacts*-relation between a *Role* in a business process definition and an *Activity*. In the following we detail this relation in the business process definition (BPD) and the Role-Based-Access to business processes in our domain model for business process compliance.

We basically apply the concept of role-based-access-control (RBAC) in an adapted form of the RBAC-Model proposed by [Sandhu et al, 1996]. Below we first give a brief general introduction of the RBAC Core Model, and then we apply it to our model.

Role-Based Access Control (RBAC) has become a widely accepted and well-known approach for managing the authorization and controlling the access to modern systems. It regulates the access of individual persons in an enterprise to the information and systems containing the information on the basis of activities the users execute in the system. The core difference between this and the user-based access systems is that instead of specifying all the activities each user is allowed to enact, the access to activities is defined based on the authorizations specified in roles. A role is usually a function used to categorize users within the organization, and is assigned to an appropriated set of permissions by an administrator; a permission being an authority to perform an operation/activity on one of the objects in the system. So when a user attempts to execute an activity on a target object, the access control system only allows it to proceed if this user is assigned the role(s) that includes the necessary permissions for that operation. The main benefit of such an approach is the ease of administration and its scalability; if a user moves to a new organizational unit with a new function within the organization, there is no need to revoke the authorizations that the user had in the previous function and grant the authorizations that the user needs in the new organizational unit; the administrator simply needs to revoke and grant the appropriate role membership for the user.

According to the description above, the core RBAC model includes sets of five basic data elements called users, roles, objects, operations, and permissions. The RBAC model as a whole is fundamentally defined in terms of individual users being assigned to roles and permissions being assigned to those roles. As such, a role is a means of naming many-to-many relationships among individual users and permissions. In addition, the core RBAC model includes a set of sessions where each session is a mapping between a user and an activated subset of roles that are assigned to the user.

A user is defined as a human being. Although the concept of a user can be extended to include machines, networks, or intelligent autonomous agents /services, the definition is limited to a person for reasons of simplicity. A role is a job function within the context of an organization with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role. Permission is an approval to perform an operation on one or more RBAC protected objects. An operation executes some function for the user upon its invocation. The types of operations and objects that RBAC controls are dependent on the type of system in which it will be implemented. In our case, the RBAC objects are business documents produced and consumed in a business process.

Below we apply the RBAC model to our business process definition model as introduced in section 4.2.2, which is visualized in Figure 28. We call it RBAC4BPD.

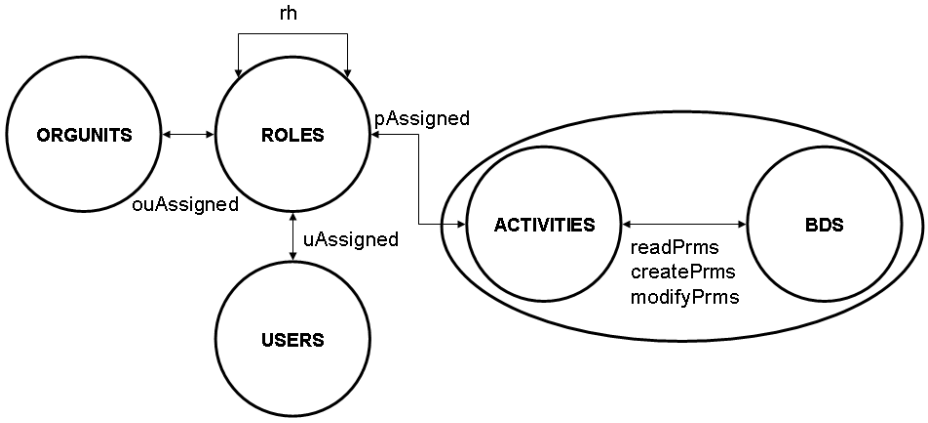

Figure 28 RBAC4BPD: Role-Based Access to Activities and Business Documents in Business Process Definition

In RBAC4BPD a set of roles are assigned to an Organizational Unit. The general role hierarchies taken from RBAC allow the user to map the structure inside an organizational unit (like a purchasing department) and to map the existing roles into the model. Hierarchies are commonly used for structuring roles to reflect an organization's line of authority and responsibility. Role hierarchies define an inheritance relation among roles. Inheritance has been described in terms of permissions; i.e., r1 "inherits" role r2 if all privileges of r2 are also privileges of r1. Below is the formal definition of RBAC4BPD:

Definition 4.19: RBAC4BPD
The *RBAC for Business Process Definition* is a tuple *RBAC4BPD = (ORGUNITS, ROLES, USERS, ACTIVITIES, BDS, uAssigned, ouAssigned, readPrms, createPrms, modifyPrms ,pAssigned ,rh)*, with

- the sets *ORGUNITS, ROLES,USERS,ACTIVITIES,BDS*
- $uAssigned \subseteq ROLES \times USERS$, is a many-to-many relation, which assigns users to roles
- $ouAssigned \subseteq ROLES \times ORGUNITS$ is a total function $ROLES \rightarrow ORGUNITS$, which assigns each role to a single organizational unit
- *readPrms, createPrms and modifyPrms* = $2^{(ACTIVITIES \times BDS)}$, the set of read, create, and modify permissions for activities on business documents
- $pAssigned \subseteq ROLES \times ACTIVITIES$, a many-to-many role -activity assignment relation
- $rh \subseteq ROLES \times ROLES$ is partial.

The RBAC-permissions in our case are the union of *read, modify, and create* permissions on business documents, upon which an activity will be allowed to enact. Thus the semantic between the *enacts*-relation in Definition 4.2 and the *pAssign*-relation in *RBAC4BPD* (Definition 4.19) is as follows: if a role enacts an activity, then a role assignment to that activity must exist and if the activity creates a business document, the activity must have the permission to create that business document etc.

4.3 Related Work

Discussion of related work in this section is divided into 3 distinct parts:

In section 4.3.1 we discuss the research done in area of risks in business processes, which we consider as related to section 4.1. In section 4.3.2 we compare related work for business process models, which we consider as related to the model of business processes proposed in this chapter. In section 4.3.3 our RBAC4BPD is compared with previous research in the area of role-based access to business processes.

4.3.1 On Risk Management for Business Processes

The concept of risk is extensively discussed in literature and is subject to various definitions depending on the domain of study.

From the financial perspective as discussed in [Markowitz, 1952], risk can be considered as "variance of return". From the project management perspective, risk is defined as a "measure of probability and consequence of not achieving a defined project goal".

Obviously, there are several types of risk, whereas in this work we consider the taxonomy of enterprise risks as presented in Figure 29. The taxonomy is derived by the work done in [ERM06].

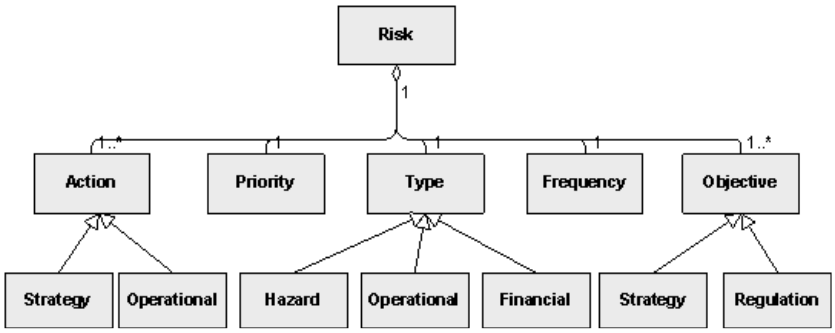

Figure 29 A taxonomy of enterprise risks [ERM06]

One interesting type withing the domain of business process compliance is the domain having to do with operational risks. A commonly used definition of operational risk is "the loss resulting from inadequate or failed internal processes, people or systems or from external events" [BaselII08].

A risk may occur in the context of the internal strategy of an enterprise, can be defined due to external factors prescribed by regulations, or can have elements of both. While controls monitor the business processes and report the occurrence of a risk for a business process, an action can be defined for each risk as a reaction to its occurrence. An action can adjust the strategic management of an enterprise or may be operational in terms of causing the reengineering of the business process. For a discussion on measurement of priority and frequency of enterprise risks please refer to [ERM06].

[Bernard et al, 2002] have proposed a conceptual model of risk and define it as a probable event and its impact on an entity: Given an initial state of an entity, probable events may affect that entity during its evolution and cause deviations from the expected future states. Risk factors may be concrete or abstract (endogenous or exogenous) and they are able to affect the likelihood or the impact of events. The impact itself is defined as the effect of the event related to the entity supporting the risk. An effect can be positive or negative.

In our case, the entities in the above conceptual model of risks are operative business processes affecting a significant account in an enterprise. A risk on such a business process is any probable event which can cause some deviations on expected outcomes, in our case financial statements asserted by an enterprise or the expected operative results of that business process, which are reflected in the significant accounts of the enterprise.

Previous research by [Sienou et al., 2006a] has shown that an integration of risks and business processes entails the following challenges:

- Domain-Specific-Knowledge: Risks for business processes are events which occur in a given business context requiring domain-specific knowledge of that domain to identify and manage. Thus it is difficult to transfer and reuse the results of risk management of a certain business process on another business process or even on another organization using the same business process.
- Risk treatment: It is expensive to identify the risk during operation of business processes. In most cases it is required to review and reengineer the business process *models*.
- Heterogeneity of Risks: Risks and events causing them are complex structures with multiple interdependencies.

Thus it is desirable to find a smooth approach to integrate risks and their treatment into business process models.

The work of [zur Muehlen et al., 2005] provides an appealing method for integrating risks into business processes. The proposed technique for "risk-aware" business process models is developed for EPCs (Event-driven Process Chains) using an extended notation. However, their notation as introduced in that work is not able to capture *all* types of process-related risks. In particular, it is not possible to capture risks related to process elements other than functions (see [zur Muehlen et al., 2005]). As a more comprehensive model, which captures different types of risks in the context of a process model, they propose a column-based notation, in which each risk type is captured in a separate column next to the process model. However we consider the proposed model for business process compliance in section 4.1.2 as a completion to the approach proposed in [zur Muehlen et al., 2005], since that work does not explicitly state how a risk is positioned inside the business process compliance domain and leaves the semantic link between risks, business processes, accounts, and controls, open.

Similarly [Goedertier et al., 2006] present a logical language PENELOPE that provides the ability to verify temporal constraints arising from compliance requirements on effected business processes. Distinct from that work, the contribution of the definition of business process compliance (Definition 4.1) in our work provides a precise model for business process compliance that can be used in a model-driven approach to develop a system for managing the business process compliance in enterprises.

[Sienou et al., 2006b] proposes a vertical and horizontal integration of risk management into process management: Horizontal integration is concerned with applying the risk management in a given process management phase in order to manage uncertainties or opportunities specific to the current context. Vertical integration is about managing the information of risk management while moving down in the process management lifecycle. Again, while we argue that the approach proposed is valid, we consider it as orthogonal to the model of business process compliance and the formal definition proposed in our work. We are concerned with formally capturing the model of business process compliance in order to provide a formal specification of a system for business process compliance. While we see the four elements (Account, Business Process, Risk, and Control) as essential first class entities in such a model, the works introduced above only capture

the interrelationship between risks and business processes including their management life cycles.

4.3.2 On Modeling the Behavior in Business Processes

A significant amount of work exists which proposes different approaches for modeling the relationship between different artifacts (control flows and data) in business process models.

The most prominent models are the traditional activity-centric workflows such as [van der Aalst, 1997][Basu et al., 2002][Georgakopoulos et al, 1995][Cubera et al., 2007] focusing on control flow. Recently data-flow-driven workflows such as [Müller et al., 2006][Sun et al., 2005] have received increasing attention. [Wang et al, 2005] propose document-driven workflow systems where data dependencies and control flows are combined in the process design. In their framework, activities (called business tasks) are defined using input and output documents as well as constraints similar to business rules and policies. We extend the concepts presented in the above works by the introduction of the business documents and by modeling their state life cycles in terms of their *transitions*, which provide a general framework to group "business level" data logically into a set of unique entities (Business Documents). In this way we show how the activities that operate on those business documents are related to each other.

Very close to the concept of "Business Documents" is the entity "Business artifacts", which were originally introduced in [Nigam et al., 2003]. They define an artifact as a "concrete, identifiable, self-describing chunk of information that can be used by a business person to actually run a business". Further "Artifacts are taken to be the only explicit information contained in the business; that is, the set of business records represents the information content of the business". The key properties of business artifacts are:

- A business artifact consists of two parts: an enterprise-wide unique identity and self-describing content.
- The content may be represented as nested name-value pairs.
- The identity of a business artifact cannot be changed.
- Consequently, an artifact cannot be split into two or more pieces, each of which has the same identity (although a different artifact with the same content but different identity can be created).
- The content of a business artifact can be modified arbitrarily; that is, values can be modified and new name-value pairs can be added.
- Content can be copied from one artifact to another.
- New information from computation, external input, or any other source can be added to an artifact.

We strongly agree with this definition and our formal model is widely aligned with the above definitions and properties. The significant difference between our model and the above definition is the way we see and accordingly designed the life cycle of a business document in terms of the transitions on the state names and their values. While [Nigam et al., 2003] designs the states of business artifacts as a fixed list of states (names), we design the life cycle of a business document as a list of states where each state can have different values (Example: *APPROVAL* state can have the values *Not Approved*, *Rejected*, *Approved* etc). This allows for a more flexible way of defining business process models.

Another related thread of work is the product-driven case handling approach [van der Aalst et al., 2005b], which addresses many concerns of workflows similar to ours, especially with respect to the treatment of process context or data. The central concept for case handling is the *case* and not the activities or the routing of the activities. The case is the "product", which is manufactured,

and at any time workers should be aware of the product. Examples of cases are the evaluation of a job application, the outcome of a tax assessment or the ruling for an insurance claim. Central to the concept of case handling are activities and data objects. States can be modeled on both activities and data. Our process definition and a case are similar in many respects. Case-handling, however, details the structure of the case using data objects that can be managed and updated independently in various activities in the context of the case. We treat data as unique entities that are updated within each activity. To maintain the proper granularity of business level operations, we do not detail the business documents and the activities. Case handling is more concerned with the case execution details by providing different states of activities, while we argue that the activities and their states proposed there can only be applied to "tasks", which mainly appear in the work list of a business user. Our activities and their interrelationships with the business documents and the transition model describe the behavior model of a system. Further the state model of data in the case handling paradigm has a different semantic than our state life cycle of business documents, which allows for a more flexible and practice-oriented (from perspective of business users) modeling approach of business documents, their life cycle and interplay with activities in a business process.

Another thread of related research significant to the concept of activities in our process model is the "services" in Service Oriented Architecture (SOA). In particular we consider *Web Ontology Language for Web Services* (OWL-S) [Martin et al. 2004] as a relevant development in this area. OWL-S provides an ontology to describe Web services semantically in order to compose them together into business processes. In our model an activity is roughly described by the business documents it reads and the business documents it modifies or creates. It is then expressed by preconditions and effects. Similarly, a "service" in OWL-S has an input, an output, a precondition, and conditional effects. Here we discuss the OWL-S model and compare it to our approach:

OWL-S is an ontology-based approach for the semantic description of Web services. It encompasses efforts to populate the web with content and services having formal semantics. The ultimate goal of OWL-S is to provide an ontology that allows software agents to discover, execute, and compose web services to business processes in an automated manner. The structure of the ontology of services is motivated by the need to answer three essential questions about a service:

- *What does the service provide to the potential clients?* This is answered by a profile, which is used to advertise the service. To capture this perspective, each instance of the class Service presents a *ServiceProfile*.
- *How is it used?* The answer to this question is given in the "process model" captured by the *ServiceModel* class. Instances of the class Service use the property *describedBy* to refer to the service's *ServiceModel*.
- *How does one interact with it?* This is a rather technical issue and an answer to this question is given in the "grounding", which provides the needed details about transport protocols. Instances of the class *Service* have a *supports* property referring to a *ServiceGrounding*.

The class *Service* provides an organizational point of reference for declaring Web services; one instance of Service will exist for each distinct published service. The properties *presents*, *describedBy*, and *supports* are *properties* (relations) of a Service. The classes ServiceProfile, ServiceModel, and ServiceGrounding are the respective ranges of those properties. The details of profiles, models, and groundings may vary widely from one type of service to another.

OWL-S suffers from problems, which haven been discussed in detail in [Balzer et al., 2004]. OWL does not give constructs that are sufficiently rich to express the data flow in OWL-S.

OWL-S therefore can be thought of as an extended OWL language, requiring specialized reasoning methods in the most general case. When it comes to modeling with OWL-S concepts, it often becomes hard to get an overview of how the different parts connect to each other. For example, the same parameter (such as an input to a process) may be referenced in several places, and the control flow of composite processes may be of significant complexity. Especially for the domain of business process compliance we see the following two shortcomings of OWL-S:

- Conditional Model: Although OWL-S Profile designates elements for pre- and post conditions (effects), it still does not specify how to declare those conditions precisely. This limitation of OWL-S Profile is directly related to the problem of describing relations between input and output parameters. The main problem is that conditions often refer to concrete parameter instances which are not known before execution. Thus, a declaration of these instances must be present in the definition of a precondition via variables. The problem is that OWL as the underlying formalism of OWL-S does not support any straightforward mechanism to declare variables. [Balzer et al., 2004] propose as a solution to this problem to extend OWL by reification of additional concepts in a similar way as it is done by OWL-S to define the data flow in a process model. As a consequence, special algorithms are required to verify such definitions and to derive knowledge from them. [Balzer et al., 2004] then come to the conclusion that subsumption reasoning is not sufficient anymore to tackle this issue. Comparing to our model, we consider our model as a formal specification on a conceptual level and we do not bound it to a certain logical formalism as OWL-S does with OWL respectively Description Logics.
- Business level Underlying Data Model of Services: As said before, a crucial requirement for modeling operative business processes for applying the compliance requirements on them is to define business level data entities that are consumed and produced by a business process respectively the services/activities in a business process. For this reason we have integrated the concept of "Business Documents" and their pertinent state model respectively the transitions on them as first class entities in our process model definition. OWL-S completely suffers to provide this aspect in its specification and it also lacks to propose how to treat data in processes and their services.

However in many respects we also follow the spirit of OWL-S, especially when it comes to the definition of an activity by providing a mechanism to specify the "effect" of its enactment. In our case, the effect will be described in terms of whether business document attributes become assigned or not, and whether new business document instances are created. On the same note, we also allow non-determinism in the execution of an activity resulting in many possible effects given by the fact that our model is kept mostly abstract without specifying any business-specific details about the underlying business documents and their state life cycle.

In context of web services and service oriented architecture *Web Services Model Ontology (WSMO)* [Roman et al., 2005] provides a conceptual framework for semantic description of web services in order to facilitate the automation of their discovery and invocation over the web. The combination of such web services can potentially result in business process models, thus we consider WSMO as related work on modeling the bahaviour in business processes. The conceptual model WSMO consists of four main elements: *Ontologies*, *Goals*, *Web Services* and *Mediators*. *Ontologies* provide the terminology used by other WSMO elements to describe the relevant aspects of a domain. Web services describe the computational entity providing some functionality in a domain. Goals represent an objective upon thir fulllfillment a web service is executed. Mediators support overcoming interoperability problems between different WSMO elements. One of the key differences between OWL-S and WSMO is the existence of the concept

Goals in WSMO for describing web services. Web Services Modeling Language (WSML) is the language used to formally describe different elements in WSMO that support different formal logic variants and therefore different expressivity levels. According to [Agarwal, 2007], WSML does not provide any modeling entity for orchestrating web services. Therefore it is not made clear in WSMO how to model business processes based on web services.

We conclude the related work discussion with the statement that, to the best of our knowledge, none of the works introduced above shows a concrete, applicable model which recognizes all the aspects required in the domain model of business process compliance, namely Role, Business Documents, Activity and the Transitions applicable to those Documents and Activities. We have developed a concrete, novel, conceptual design for that domain, which explicitly contains all the necessary semantic relationships between the aspects mentioned above.

4.3.3 On the RBAC and Business Processes

Many extensions to RBAC models have been proposed, such as task-based access control (TBAC) [Thomas et al., 1997] or web service-based access control (WS-RBAC4BP) [Peng et al., 2004]: TBAC models permissions to enact operations from a more task-oriented perspective than the traditional subject-object one which is proposed by the core RBAC model. It is an active security model that makes access decisions based not only on the operations the subject (user/role) owns but also on the current execution context of the business process, in which an activity may be accessed and enacted. It also takes temporal constraints into account where access is permitted based on a just-in-time fashion for the tasks related to the sessions introduced in the core RBAC model.

Further, in the context of Service Oriented Architecture (SOA), where Web services may be used to realize cross-organizational business processes, [Peng et al., 2004] proposes an extended RBAC model for web services. The most significant difference from traditional RBAC models is that their model takes companies as subjects and Web services as protected objects. There are two kinds of constraints in their RBAC-extension: supplement constraints and authorization constraints. These constraints must be enforced when the relations are constructed and access decisions are made.

Many Access Control Models including the ones above based on RBAC have limitations regarding the representation of the relationship between user and roles, the business level operation(s) they enact during the execution of a business process, and their organizational embedding in an enterprise [Chandramouli, 2003]. Thus we adapt the RBAC model in a very limited and cautious way in order to use it in our approach and bridge the gaps mentioned above. We see our work rather as a completion to the RBAC-extensions mentioned above since they are concerned with different facets of modeling the user's and the system's access to resources.

4.4 Conclusion

In this chapter we have developed a novel model and formal specification for the domain of business process compliance and the entities controlled within it. We started by introducing an upper-level model containing entities relevant to the domain, namely: account, business process, control, and risk. A part of the model was tackled, namely the model of a business process and the controlled entities. We discussed the relevant first class entities of a business process model from the angle from which they are observed in operative business processes in the industry. We argued that the traditional, purely activity-oriented view of business processes is not enough to

describe a business process the way the business users understand it. We argued that the "business level data", which are consumed along the execution phase of a business process, are crucial and have to be considered as first class entities in a business process model. We introduced the concept of "Business Documents". We recognized the question of "who" is involved in an operative business process by introducing the entity "Role" as a first class citizen in a business process model. As a framework to determine the concepts in our model we used the Hammer's Framework, which introduces seven dimensions of the work done in enterprises. We put the framework in the context of business processes and derived the motivation of three entities: activity, business document, and role.

The notion of controlled entities for business process compliance that are subject to controls was introduced, i.e. a control will constrain the behavior of a certain controlled entity in a business process. The controlled entities are: activity, business documents, user, role and a control itself.

Step by step, we showed the position of each controlled entity type in a business process model and showed the semantic interrelationships amongst them. An exception is the model of the controlled entity control that will be separately introduced in chapter 5.

As the next refinement step on the model, we introduced the formal model of business documents. We designed the lifecycle of a business document instance in a business process as a set of states, where each state can potentially have different values. Based on the business documents and the concept of conditions, we then formally designed the behavior of an activity. It consists of: a tuple of the business documents it reads and modifies, the preconditions of the activity, and the effects on the business documents newly created and those which were consumed by the activity. The "flow" of a business process can then be described in terms of "transitions", which may happen between the activities and the states of business documents.

Further, the model of business process instance and its depending artifacts transition instance and business process context were introduced. The effectiveness of a defined control on a business process model will be checked during the execution time of business processes using a business process instance.

For the design of the way users enact the activities we tightly followed the RBAC model, with some necessary conceptual adaptations. The extension aims to integrate the concept of business documents in the model. In order to reflect the organizational structure in which the operative business processes happen, we introduced the entity "OrgUnit", which builds on the relation of "role hierarchies" proposed in RBAC.

Throughout this chapter we frequently referred to the scenario use cases provided in chapter 2 and exemplified the concepts and the relationships developed by different facets of the scenario.

5 Business Process Verification

In the previous chapter, in section 4.1.2, we introduced the proposed method for realizing effective internal controls on business processes. It consists of two steps: the *verification* of business process models in order to verify that they are designed as required, and the *validation* of business process executions in order to validate the fact that they work as designed.

Based on the proposed method, in this chapter we introduce the solution provided by this thesis for verification of business process models produced during design or a reengineering phase of business processes (see section 3.2).

Note that this chapter is not directly related to the formalizations we gave in the previous chapter, since for the implementation of the business process design verification presented here we used a tool (given in scope of a project that will be described shortly) that does not support all the modeling entities developed in previous chapter. In section 5.1.1.2 we give detailed explantations of the relationship between used models.

Generally verification of process models spans over two aspects:

1. Checking that a model satisfies a set of properties given by a formula
2. Checking general properties of a model regarding its "syntactical" correctness.

In this chapter, the proposed solution for business process verification is related to the first aspect, i.e. the properties represent some business level correctness requirements on a business process model.

The benefits of a method for verification of business processes in the context of business process compliance are threefold:

1. Through an automated verification of a set of business process models that exist in companies, the risk of designing, implementing and consequently executing noncompliant business processes can be decreased
2. By automating the task of verification achieved for the most part manually in today's business world, the cost of manual inspection, analysis and testing of business process models for compliance can be reduced.
3. Completely new-designed and implemented business processes, once verified, can be considered as compliant, while it should be noted that they cannot be considered as remaining always compliant (see section 3.1.2.2.2).

The verification of a business process model is realized using formal methods. These methods seek to establish a logical proof that a system will work correctly, i.e. that it is correctly designed. A formal approach requires:

- a modeling language to describe the system, in our case a business process;
- a specification language to describe the business level correctness requirements on the system and
- an analysis technique to verify that the system meets its specifications.

The model describes the possible behavior of the system, and the specification describes the desired behaviors of the system. The statement that a model satisfies the specification is now a logical statement, to be proved or disproved using the analysis technique.

The modeling language to describe the system in our case is provided by the business process definition BPD (Definition 4.2) introduced in section 4.2.2. In the current chapter we introduce a specification language using Web Ontology Language (OWL) to semantically describe the business processes and their business level correctness requirements together with the analysis technique used to verify a business process design.

In this chapter, we start by outlining the basics required for understanding the approach as well as the context of the project ATHENA, throughout which the research in this chapter was achieved. In project ATHENA, a business process modeling approach was developed called *Cross Organizational Business Processes* (CBP). CBP provided the underlying models of the tools implemented in the project. These tools serve as the underlying business process modeling infrastructure in this chapter, on which the verification approach is applied. In this section we also introduce OWL, the ontology language, which served as the underlying formalism to represent CBPs. In section 5.2 we detail the approach by presenting the CBP ontology developed in OWL and also how the business level correctness requirements can be expressed and verified based on the ontology. Here we also provide the implementation of the approach along with its integration architecture in the underlying tool environment. In the subsection relating to related works, we briefly introduce a scenario in the context of an internal SAP project, in which the research results were practically applied in a customer prototype, before we discuss and compare other possible approaches used for the verification of business processes.

5.1 Basics

5.1.1 Introduction to Cross Organizational Business Processes

The concepts and implementations provided in this chapter were realized within the EU-funded research project ATHENA. The ATHENA project deals with the problem of interoperability between enterprise information systems. Today common business paradigm is dominated by service outsourcing, in which an enterprise focuses on its core business processes and has secondary process parts enacted on its behalf by service provider organizations. These kinds of business processes, within the ATHENA context, are called Cross-organizational Business Processes (CBPs) [Lippe et al., 2005], i.e. processes that cross two or more enterprises. Solutions to problems associated with CBPs are one of the main goals of the project. Support for the semi-automatic modeling and automatic execution of these processes were the focus of study in the different research groups, which investigated the problem at business and technical levels.

In ATHENA concepts were developed to classify process types pursuing different goals. Processes are divided into three levels of abstraction: a level suited for business people, an intermediate level suited for process analysts as well as business people, and a level suited for IT-experts. At this last level the processes may be executed by computer systems. Furthermore, ATHENA presents a concept to model cross-organizational processes without having to reveal the internal, private information of enterprises. This concept includes three different process types that vary according to the degree of information provided about a single enterprise as well as the degree of information provided about the whole collaborative process:

- Cross-Organizational Business Process: This process type is intended to explain the whole collaborative process and contains mainly abstract information about the roles the involved enterprises play
- Private Process: This process type is used only internally by an enterprise and contains all information regarded as necessary by internal users
- View Process: This process type hides sensitive information contained in the private process of an enterprise and provides partners with information on how to interact with the enterprise owning this private process.

Based on these concepts, modeling and execution tools were developed to support collaborative business processes on each level; the approach of having three different levels of

abstraction was also implemented. ATHENA provides the tool *Maestro*, which is used for designing CBPs. They can be executed by *Nehemiah*, a CBP Execution Engine. It is possible to model processes at the business level and to transform them to the technically detailed process-models used in Maestro. Apart from this and based on formal operators, a method was developed to enable horizontal transformation. Thus automatic transformations from view process to private processes and vice versa are also supported by the Maestro tool.

5.1.1.1 Modeling CBPs with Maestro

Maestro supports the graphical modeling of business processes. Business processes are graphically modeled as a set of abstract activities and the dependencies between them. Each activity may have a so-called *Task Profile* attached to it, which gives this activity some kind of functionality. A task profile can be either a manual user interaction (*user task profile*) or a Web service invocation (*service task profile*).

The management of task profiles is accomplished by a tool called *Gabriel*. By allowing the orchestration of web service invocations into business processes, Maestro paves the way for automated business process execution without human interaction. The actual enactment of business processes is then done by Nehemiah. Maestro realizes the process abstraction concept, i.e. it distinguishes between *private processes*, *view processes*, and *public processes* in order to retain internal knowledge of a company while inter-operating with external business partners on a business process level.

The main graphical components of the Maestro business processes are: *activity nodes*, *coordinator nodes, sender nodes,* and *receiver nodes*. In a valid Maestro business process, these four node-type elements are connected with each other by means of the *directed edges*.

Coordinators control the actions which take place between activities, sender nodes, and receiver nodes. Several kinds of coordinators exist, each of them influencing the activity flow in a certain way. The different kinds of coordinators are: *begin* coordinators, *end* coordinators, *choice* coordinators, *while* coordinators, *merge* coordinators, *sync* coordinators, *fork* coordinators, *null* coordinators, and *do* coordinators.

During the creation of the CBP, for each communication among the business partners that were determined, a sender node is inserted into the business process of the business partner who actually invokes this communication, and a receiver node is inserted into the business process of the business partner who retrieves information during this communication. The insertion of these nodes is necessary due to technical reasons and, by attaching appropriate service task profiles to these sender and receiver nodes, data exchange between the business partners is then realized using web service technology.

A CBP in Maestro can be remodeled in terms of changing the sequence of activities, coordinators, sender nodes, and receiver nodes. Or a CBP may be changed by the way the task profiles are attached to the activities of the business process. Depending on the kind of task profile, this change could refer to a manual user task or to a web service invocation.

5.1.1.2 Relationship between a CBP and a business process according to BPD

The main focus of business process modeling based on the concept of CBPs is the design of interactions between different business partners in a collaborative scenario. Here web services are the main facilitators for realizing the communication and exposing the functionality between different business partners. Thus modeling CBPs in Maestro and executing them in the Nehemiah engine is bounded to the usage of web services, which we consider s related to the

implementation level in business process management. In contrast, a business process designed according to the definition of a BPD is not bound to the usage of a certain technical implementation, as it is the case with CBPs. The involved parties in BPD are defined through users and their roles. During the implementation, by assigning roles and users appropriate technical endpoint addresses, (for instance using web service addressing etc.), it will be established whether those roles and users are inside the company or remote (external business partners). Naturally, external business partners will be assigned remote addresses.

According to BPD, one of the first class entities of a business process is the data created and consumed in a business process, which is reflected by the notion of business document. Maestro does not provide any modeling entities to design the data in CBPs.

However due to the concept abstraction provided by BPD a CBP can be treated as a BPD on a conceptual level. A CBP cannot be considered as a BPD, because it does not support the notion of business documents (and all relating concepts such as the states through which a business document instance can go).

In Table 9, a mapping between the core modeling elements of CBPs as introduced in section 5.1.1.1 and the core concepts of BPD as introduced in section 4.2.2 is provided.

Table 9 Mapping between CBP and BPD concepts

CBP	BPD
Activity Node	Activity or State Change Command (scc)
Coordinator Node	Condition
-	Business Document
Edge	Transition
-	User
-	Role
Attachment of a user or task profile to an activity node	Enacts
-	creates / reads/ modifies
Begin	Start
End	End
User/ Service task profile	Activity or scc
Private/ Sender/ Receiver task profile	-
Private/ View/ Public view	-

5.1.1.3 Limitations of CBP Verification in the context of Business Process Compliance

A verification of business processes designed in Maestro has certain limitations. These limitations are related to the fact that not all concepts of a BPD are supported in the model of CBPs (see Table 9). We consider the lack of business documents in a CBP especially to be one of the main shortcomings of the CBPs. See section 0, for a detailed discussion on the necessary of

inclusion of business documents in a model for business processes in the context of business process compliance.

It is possible to verify the model of a system in general (and in our case particularly business processes) through the elements that build the system, i.e. by the elements that the static design of a system contains. Thus it is not possible to verify the model of CBPs according to the elements that a CBP does not provide. A CBP only reflects the *activity flow* in a business process by supporting the notion of activity nodes, coordinator nodes etc. Thus the verification of a CBP presented in this chapter provides only the verification of the *activity dimension* of a business process design.

5.1.2 Ontologies in Web Ontology Language (OWL)

In this section we explain some basics of the Web Ontology Langage (OWL) that are necessary to the understanding of the verification approach for business process discussed in this chapter. The CBPs introduced in the previous sub-sections are modeled in OWL.

OWL [OWL2004] is an ontology language developed by the W3C Web-Ontology Working Group. OWL was developed as an extension of RDF Schema [RDFS2004]. OWL is based on Description Logics (DL) [Baader et al., 2003]. DL is a decidable subset of First-Order Logic.

Knowledge in DL is represented as a hierarchical structure of classes (also called concepts), thus as taxonomies. A DL system is usually divided into two parts: the TBox and the ABox. The TBox defines terminological knowledge and consists of declarations describing general properties of classes, thus it contains the definitions of classes and its relations. The ABox of a DL system contains the definition of instances (also called individuals).

OWL exists in three dialects, namely OWL-Lite, OWL - Description Logics (OWL-DL), and OWL-Full, which differ in terms of expressiveness and decidability. OWL- Lite is a subset of OWL - DL, which in turn is a subset of OWL Full. Since OW-Lite's expressiveness only provides vocabulary for defining taxonomy with some simple constraints, it is much simpler to provide tool support for this dialect, especially in terms of reasoning. OWL-DL offers a greater expressiveness while still being decidable, whereas OWL-Full supports the full OWL expressiveness but is no longer decidable and is therefore not supported by today's reasoning tools. As a result, when one requires reasoning support with good expressivity, OWL-DL is the OWL dialect one should use [Dean et al, 2004].

OWL uses RDF's XML syntax. Each resource that is being defined can be assigned to a Uniform Resource Identifier (URI) consisting of the namespace of the ontology and a string identifying the resource. OWL ontologies use namespaces as their identifiers, therefore each ontology has to have a unique namespace.

The main resources for the representation of OWL ontologies are *classes*, *properties*, and *individuals*. Classes are defined using *owl:Class* elements. Basically, two predefined classes exist in OWL, namely *owl:Thing*, the universal class of which every other class is a subclass, and *owl:Nothing*, an empty class, of which every other class is a superclass. Expressive elements like *owl:subClassOf* or *owl:disjointWith* may be used to further specify a class. The *owl:subClassOf* element specifies that the class is a subclass of a given other class, whereas the owl:disjointWith element determines the following: Given two OWL classes *ClassA* and *ClassB*, which are disjointed from each other, if an individual is an instance of *ClassA*, then it cannot be an instance of *ClassB* at the same time, and vice versa. For a full list of *owl:Class* elements, please refer to [Dean et al, 2004].

An individual stands for a single real world object being an instance of one or more classes. In OWL, relations between individuals are modeled by properties. Two types of properties exist: 1)

Object properties relate individuals with other individuals and 2) Datatype properties relate individuals to data type values such as an integer or a string value. XML Schema data types are supported for the definition of OWL data type properties.

Another important feature of OWL is that it allows the user to define restrictions on top of properties. With the *owl:Restriction* element, a number of restrictions can be defined that constrain the individuals of a class in terms of the number of relations to other individuals through a certain object property they have. We will not discuss further details on the syntax or the semantics of OWL here as this is not our intent in this section. This section should serve to give an insight of OWL basics. For full and formal definitions on the syntax and the semantics of OWL refer to [Dean et al, 2004].

5.1.3 SWRL – A Semantic Web Rule Language

The expressiveness of OWL does not support the expression of rules, which is regarded as an important additional expressive feature. Therefore, a rule extension to OWL ontologies is needed. Semantic Web Rule Language (SWRL) [Horrocks et al., 2004] is currently one of the most promising and most widespread semantic web rule languages. SWRL combines the OWL sublanguages OWL-DL and OWL-Lite with a sublanguage of Rule Markup Language (RuleML) [RuleML]. SWRL allows the defintion of the Horn-like rules for OWL-DL and OWL-Lite. These Horn-like rules consist of a body (also called antecedent) and a head (also called consequence). This relationship could be visualized as follows:

$$body \rightarrow head$$

Such a rule is to be read as follows: If the conditions specified in the body apply, then the conditions specified in the head likewise apply. Both the body of a rule and the head consist of a conjunction of one or many atoms. A general SWRL rule could be visualized as follows, where A_1 to A_n represent the body atoms and B_1 to B_n represent the head atoms:

$$A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow B_1 \wedge B_2 \wedge \ldots \wedge B_n$$

The atoms, that the rule body and the rule head consist of, can be of the form C(x), P(x,y), sameAs(x,y) or differentFrom(x,y), where C is an OWL description, P is an OWL property, and x, y are either variables, OWL individuals or OWL data values [Horrocks et al., 2004]. Accordingly, atoms can be formed from unary predicates (OWL descriptions or OWL classes), binary predicates (OWL properties), equality predicates (sameAs), or inequality predicates (*differentFrom*). Consequently, an atom consists of a predicate, being one of the four predicates mentioned above, and a set of variables, OWL individuals, or OWL data values.

SWRL rules are expressed using the vocabulary of the underlying OWL ontology, mainly with regard to OWL classes, individuals, and properties. SWRL also offers further expressive vocabulary such as built-in predicates. We will not go into detail on those concepts as they are not needed to achieve the goals of this thesis. For a full and formal definition of the syntax and semantics of SWRL, refer to [Horrocks et al., 2004].

A drawback when it comes to OWL-DL knowledge bases with SWRL rules is that these knowledge bases cannot be reasoned by today's ontology reasoners due to the undecidability problem, which is further discussed in [Horrocks et al., 2004b]. One approach for overcoming this problem was developed by [Motik et al., 2004] and led to a subset of SWRL called DL-safe

rules, which is restricted to some extent in order to make reasoning over OWL-DL ontologies with rules decidable.

DL-safe rules make reasoning over OWL-DL ontologies with rules decidable by restricting the expressivity for rule definition. The structure of the rules regarding the rule body, the rule head, and the fact that they consist of one or many atoms is exactly the same for SWRL rules as for DL-safe rules. The restriction of DL-safe rules is best expressed by the definition of a DL-safe rule, which is given by [Motik et al., 2004] as follows: "A rule r is called DL-safe if each variable in r occurs in a non-DL-atom in the rule body." DL-atoms are the ones introduced as atoms in SWRL.

5.1.4 KAON2

KAON2 [KAON2] is an ontology management tool for managing OWL-DL ontologies and rules. Furthermore it offers rule support for OWL-DL ontologies by supporting the DL-safe subset of SWRL. The class diagram, which visualizes the structure of a KAON2 Rule object, is shown in Figure 30. On the basis of this class hierarchy, we developed the rule expression mechanism for the verification of business processes.

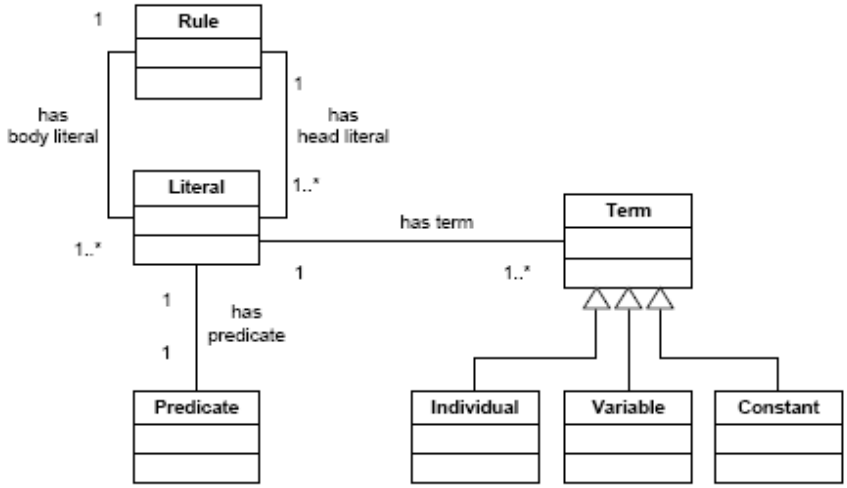

Figure 30 KAON2 – Rule Object

KAON2 uses the concept of Backward Changing to reason over DL-Safe rules on top of an OWL-DL knowledge base. Backward chaining starts with a list of goals and works backwards from the head of rule to its body to check whether there is knowledge, in terms of individuals, available that will support any of the heads of the rules. Thus backward chaining is considered as being “goal-driven”, meaning that it starts with a consequence which the engine tries to satisfy. If it can not, it will search for consequences that it can, known as 'sub goals', that will help satisfy some unknown part of the current goal - it continues this process until either the initial consequence is proven or there are no remaining sub goals.

For a complete discussion on the underlying concepts and reasoning algorithms used in KAON2, refer to [KAON2].

5.2 Approach for Business Process Verification

The solution provided in this thesis for verification of business process models includes following conceptual steps:

- Define a formal description for the business processes as a formal ontology.
- Express the business level correctness requirements (BLCR) of an enterprise's specific business process definition according to the terms and concepts defined in that formal ontology
- Store the specific business process definition as a semantic enriched model according to the formal ontology (Business Process Model Instance)
- Use an Inference Engine, which takes as input the declarative rules and the semantic process model instances and infers whether the current business process model instance violates the existing set of given rules.

The steps above are visualized in Figure 31, including the technology used for implementing the approach: For the business process definitions the CBPs modeled in Maestro are used. The CBP ontology has been developed in OWL-DL. Business Level Correctness Requirements (BLCRs) are represented as DL-Safe-rules. KAON2 Inference Engine is used as the underlying reasoning infrastructure to verify whether a business process model instance satisfies the set of BLCRs.

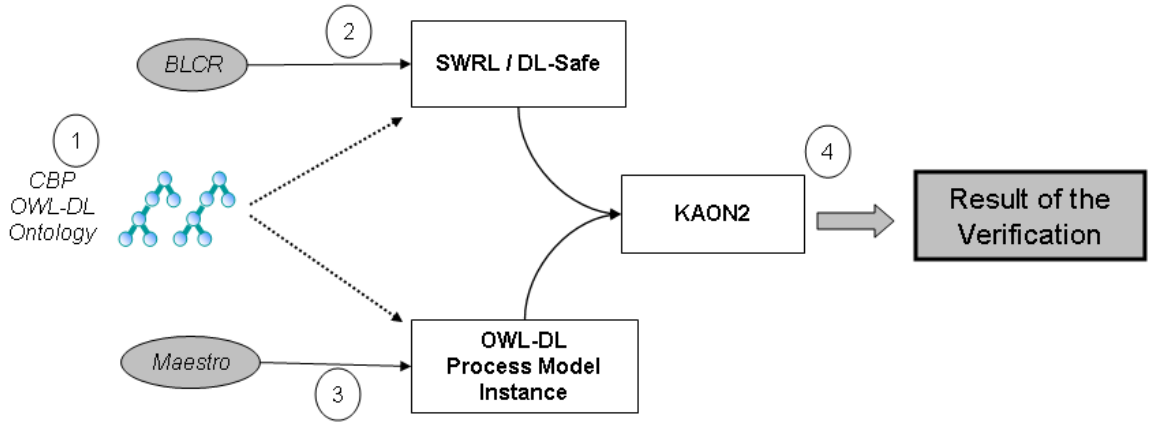

Figure 31 Conceptual steps for Business Process Verification

At design time, business processes are modeled in Maestro and then saved to the business process repository. The business processes can then be enacted by the Nehemiah engine at execution time. The latter step is out of scope of business process verification.

The verification mechanism proposed is supposed to be integrated in this sequence right after the business process modeling step, which is illustrated in Figure 32. The figure outlines that, after the business process has been defined or modified, it is verified by a reasoner in a business process verification step, before being added to the business process repository.

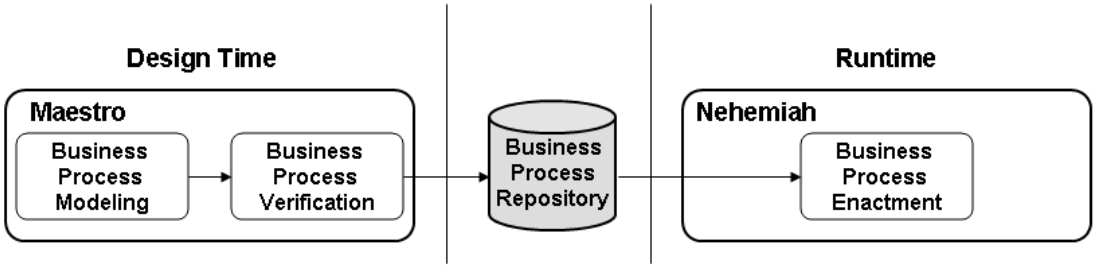

Figure 32 Sequence of process modeling including a verification step

In the following we go into the approach in greater detail:

5.2.1 CBP Ontology

The semantic representation of CBPs is necessary for the verification mechanism because defining the structure, the concepts, and the relationships of Maestro business processes semantically builds the basis for being able to express and to capture the business level correctness requirements, i.e. the constraints. In a way, the semantic representation provides the vocabulary for expressing these constraints. Practically, the business process expert needs to be able to save a semantic process model instance of the business process model he is currently designing in Maestro. This, in a first step, requires the creation of an ontology, which we called a semantic CBP model or *CBP ontology*.

It contains all relevant classes, concepts, and relationships regarding CBPs and acts as a blueprint for each business process model instance ontology that is created. In the following we further detail the CBP ontology developed:

The ontology defines all relevant classes and properties that can be used for the semantic description of CBPs. In order to avoid redundancy, it is common to model the classes and properties as a model ontology and to store the individuals, i.e. the actual business process model instances, in a separate OWL file, which we call business process model instance ontology. The taxonomy of the model ontology is shown in Figure 33.

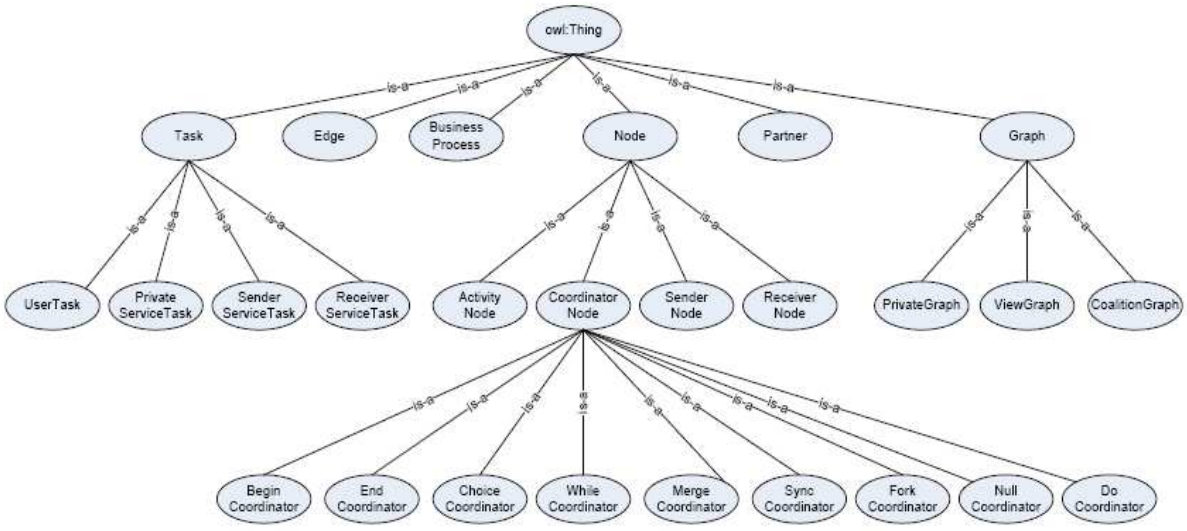

Figure 33 Taxonomy of CBP Model Ontology

The root class of the CBP ontology, which is the class *owl:Thing* for all OWL ontologies, has the following subclasses (see Figure 34):

- Graph
- Business Process
- Partner
- Node
- Edge
- Task

Graph represents a Maestro business process graph containing one or several business processes run by different companies. Its subclasses are *CoalitionGraph*, *PrivateGraph*, and *ViewGraph*. It is connected with the class *BusinessProcess* through the object property *hasBusinessProcess*. Note that the number of business processes which may be connected to it through the *hasBusinessProcess* object property differs depending on the graph type.

The *BusinessProcess* class is related to *nodes* and *edges* by the object properties *hasNode* and *hasEdge*, respectively. It can be linked to the class *Partner* through the object property *hasPartner*.

The class *Partner* stands for one business partner, who is linked to the *BusinessProcess* class by its object property *hasPartner*.

The class *Node* represents any kind of node contained in a business process. By means of the object properties *isPredecessorOf*, *isDirectPredecessorOf*, *isSuccessorOf*, and *isDirectSuccessorOf* each individual of the *Node* class is in some way linked to all other nodes within the according business process. Its subclasses are *ActivityNode* (representing an activity in a business process), *SenderNode* (standing for a node that enables outgoing communication in a CBP), *ReceiverNode* (standing for a node that enables incoming communication in a CBP), and *CoordinatorNode* (representing a node that aligns and manages the execution flow of a business process instance later). Activity nodes can be connected to an individual of the *ServiceTask* class through the object property *callsServiceTask*. In the same manner, sender nodes and receiver nodes can be connected to the *SenderServiceTask* class by the object property *callsSenderServiceTask* or to the *ReceiverServiceTask* class by the object property *callsReceiverServiceTask*, respectively. In addition to linking service tasks to activity nodes, individuals of the class *UserTask* can be linked to activity nodes through the object property *callsUserTask*.

The class *Edge* links two individuals of the *Node* class, which are both found within one graph, with each other. It is connected to exactly one source node and to exactly one target node by the object properties *hasSourceNode* and *hasTargetNode*.

An individual of the *Task* class refers to a task profile that can be attached to a node of a business process. The subclasses of this class are *UserTask, Private-ServiceTask, SenderServiceTask*, and *ReceiverServiceTask*. These different kinds of task profiles are needed to correctly represent the communication between a process model and its implementation, i.e. the connections between nodes within Maestro and a web service endpoint.

Each class in the above ontology is shown as a rectangle in Figure 34.

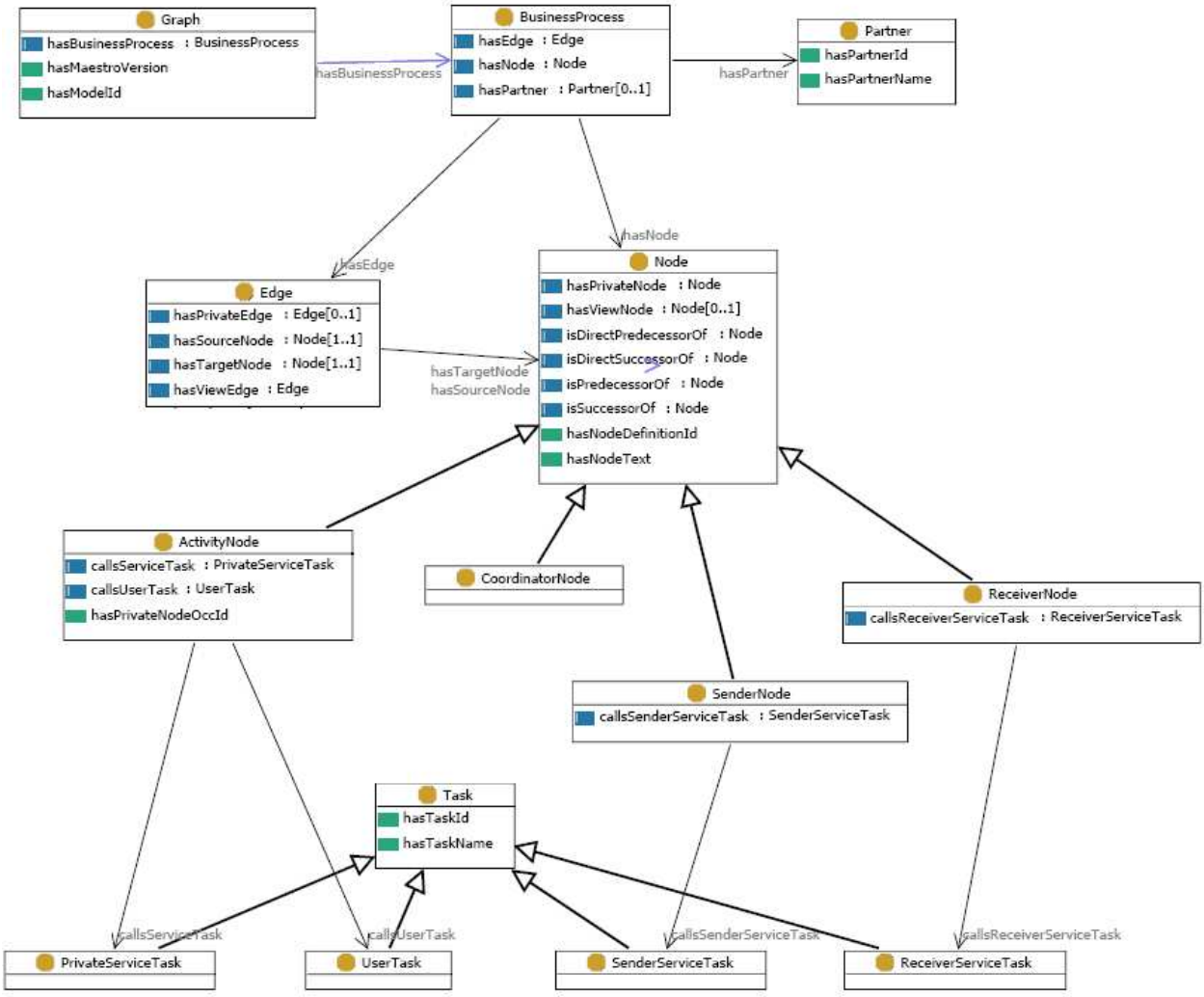

Figure 34 Main classes and properties of the CBP ontology

5.2.2 Expressing Business Level Correctness Requirements

A business level correctness requirement that can be captured could be, for example, a constraint through which web services have to be called in a certain activity of a business process, or which specifies that after a certain activity of a business process, a certain user task must be performed. Although these constraints are related to a certain business process, they should be decoupled from the actual technical representation of the business process, so that they can persist, regardless of whether the belonging business process is redesigned or even deleted. Thus, the constraints are to be stored separately from the semantic business process model instance itself.

We should extend the business process modeling sequence shown in Figure 32 by the extended modeling sequence depicted in Figure 35.

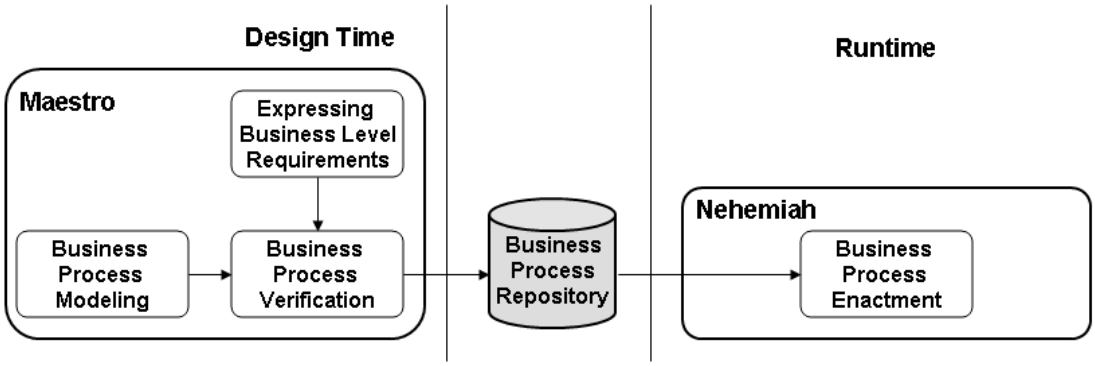

Figure 35 Business process modeling sequence including expressing business level requirements

Expressed in terms of literals and the vocabulary given by the model ontology described earlier, a rule could be defined using the rule editor for CBPs shown in Figure 36:

Figure 36 Add rule dialog

The "Add rule" - dialog constrains the vocabulary that can be used for defining business rules by offering only the available vocabulary in combo boxes. For each literal, regardless of whether it is a body or a head literal, the same vocabulary exists. Each literal of a rule consists of a predicate and a set of terms. The predicates that may be used for modeling the rules are confined to binary predicates, also called properties. Binary predicates, as the name suggests, have exactly two terms, which is why the literals of the "Add rule" - dialog always consist of one property and two terms. The properties available in the "Add rule" - dialog reflect the object properties and data type properties defined in the model ontology. In order to make the "Add rule" - dialog more user friendly, it does not show the names of the properties as modeled in the ontology, but rather displays a description of them that is easier for the user to read. The two terms that belong to every property are also called domain and range. Figure 37 identifies the domain, the property, and the range of an example literal displayed in the "Add rule" - dialog. The domain of a literal, with regard to the "Add Rule" - editor, is always some kind of variable. The property may be an object property or a data type property. When it is an object property, the range is a variable, whereas when the property is a data type property, the range is a string value.

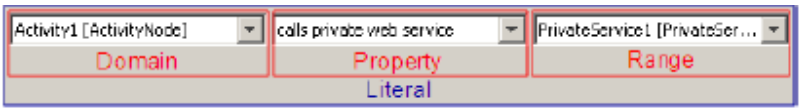

Figure 37 Structure of the Add Rule Dialog

5.3 Overall Architecture and Implementation

The architecture of the prototypal verification extension consists of the following three main parts: The Maestro application provides the UI, the KAON2 framework, which is responsible for ontology and rules management and additionally for reasoning over the ontological Knowledge base, and the ontological KB containing OWL ontologies and rules. The functionality offered by the KAON2 framework can be separated into three parts: Reasoning Engine, Ontology Management, and Rules Management, as shown in Figure 38. It gives an overview on the architecture of the prototype for business process verification integrated in the Maestro tool. On the one hand, Maestro enables the process modeler to save model instance ontologies of business processes, which are then created through KAON2 Ontology Management and saved as OWL ontology files. On the other hand, it allows for the creation of rules by the business process expert, which are processed in KAON2 Rules Management and saved to rule files. These rule files together with the CBP ontology and the process model instance ontology build the ontological knowledge base. The CBP ontology provides all relevant concepts and the relationships between them regarding business processes, whereas the instance ontology, which can be seen as an instance of the model ontology, represents an actual business process modeled by the process expert. On the basis of this ontological KB, reasoning can be conducted by the KAON2 reasoning engine and the results can be passed on to the Maestro UI and thus to the user, i.e. the process expert.

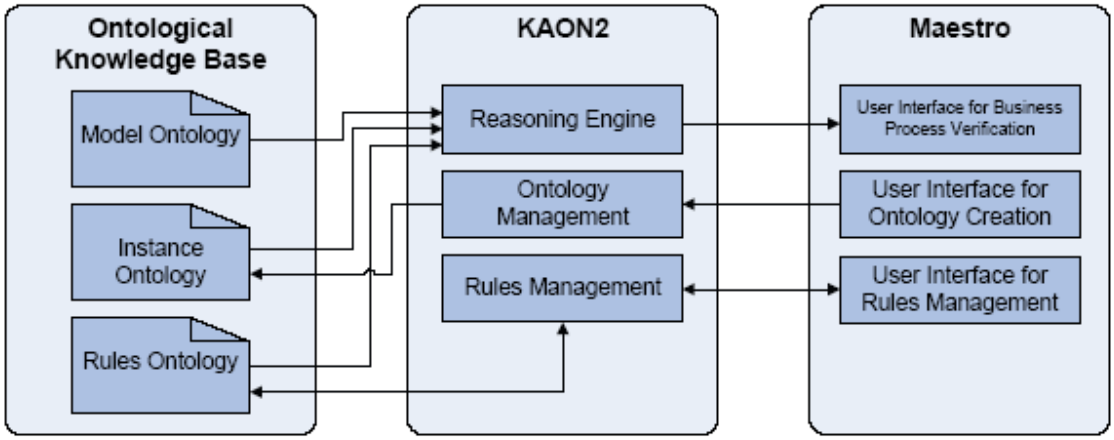

Figure 38 Overview Architecture of the Business Process Verification Approach

Because the CBP ontology is an OWL-DL ontology that contains all concepts and properties of Maestro business processes in general, it is regarded to be TBox knowledge. It is business process independent and therefore static, and it builds the basis for each process-specific KB. Accordingly, only one global OWL ontology file for the model ontology exists. The instance ontology is an OWL-DL ontology containing business process specific information. It consists of individuals of the concepts that are described in the CBP ontology. Therefore, the knowledge

contained is ABox knowledge. Consistent to the idea of ontologies, the process model instance ontology has to import the model ontology, since it uses knowledge from the model ontology. The creation of instance ontology is to be started by the process expert through the Maestro UI after he has modeled a business process. The instance ontology is stored in an ontology file that is named according to the name of the business process to which it belongs.

On the basis of the existing KB consisting of the model ontology and the instance ontology, rules can be expressed by the business process expert. The rules are saved in a separate OWL ontology file, which contains only information on the rules. These rules are business process-specific as well. The rules ontology is also imported by the instance ontology, so that the knowledge from the rules ontology is also available in the instance ontology. To clarify the structure of the ontological KB, it is visualized in Figure 39.

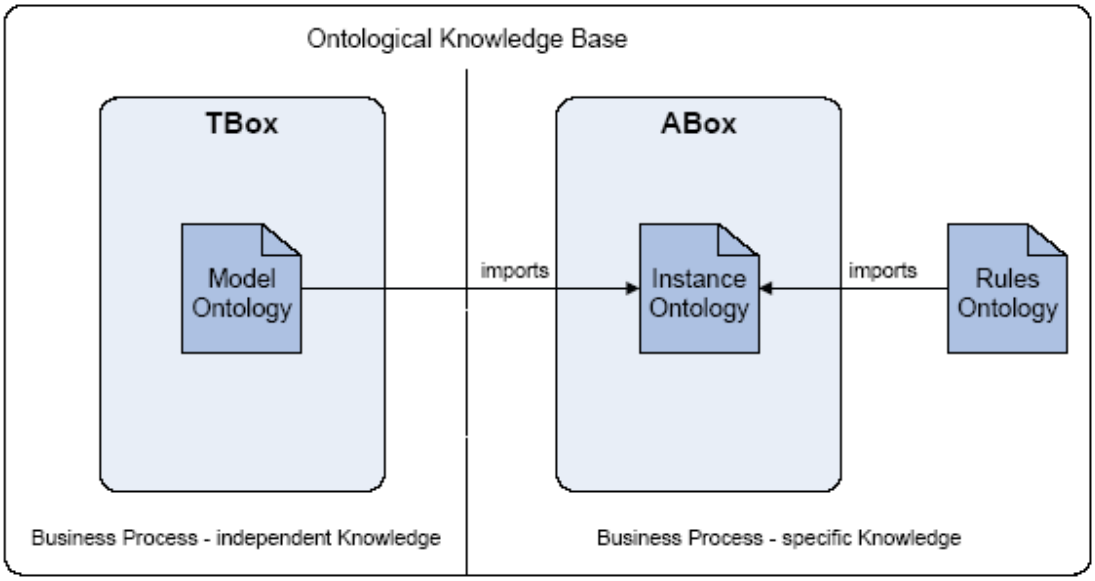

Figure 39 Ontological Knowledge base

5.4 Related Work

5.4.1 On the Application of the Approach in the context of an internal SAP project

We used the presented approach for the verification of business process models in the context of an SAP internal project in the area of carrier-shipper-solutions. The integration of carrier (web) services in a standard Order-To-Cash business process is enterprise-specific. The situation is that the same web service, provided by a certain carrier company, may be integrated in different activities of the Order-To-Cash business process run by different customer enterprises. This results in different variants of the same business process. The verification approach was used as a prototype using Maestro to verify whether the carrier web services were integrated in the correct way, as required by a customer company in the Order- To-Cash business process. The scenario is as follows:

Two different business situations for two different customer enterprises (shipper 1 and 2 in Figure 40) lead to different configurations of the same core-carrier services *Calculate Rate*, generate *Routing Code, Labeling* and *Manifest*. These services are provided as Web services by a

carrier company and are integrated into the standard order-to-cash-process in each customer enterprise. In the case of process variant 1, the *rate calculation* and the *routing code calculation* are done during *Sales Order* whereas *label generation* is done after the goods are packed. In process variant 2 the *routing code calculation*, *rate calculation* and *label generation* are all performed after the goods have been *packed*. The verification approach presented in this chapter provided a mechanism that enables a business user to express and verify the business level call dependencies for each process variant in SWRL using the Maestro rule editor extension. Once the SWRL statements are added to the KB, the business user can use the verification mechanism to determine whether the current technical configuration of the CBP still satisfies the previously expressed BLCRs on its CBP.

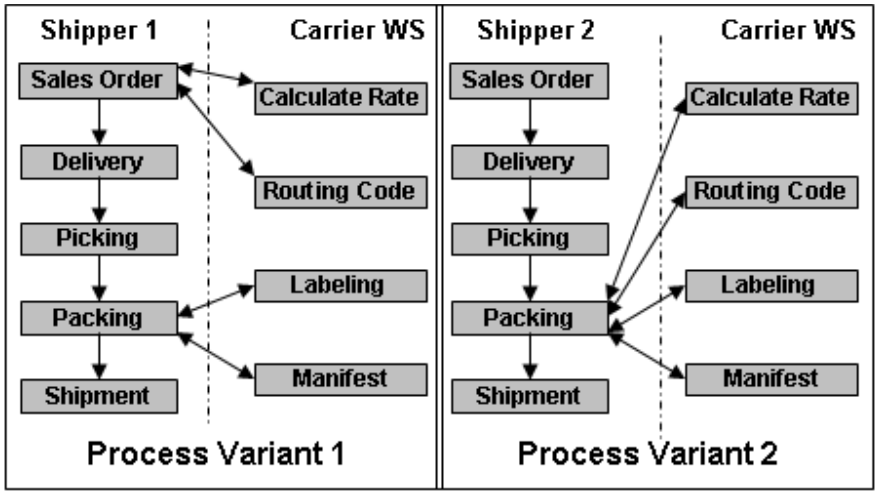

Figure 40 Two different shipping process variant

5.4.2 On Model checking of business processes

One could question the usefulness of developing separate process model ontology as introduced in section 5.2.1, when ontologies like OWL-S have already been proposed. We argue that the motivation behind the development of OWL-S was rather dynamic web service discovery, selection and composition, where our objective is the verification of already existing process models (built on top of a set of already existing services in the case of CBPs). Further the verification of OWL-S process models is done manually and requires human interaction, whereas our approach supports the automated formal verification, which is opposed to traditional techniques such as testing and simulation and has two main advantages (i) formality - the intuitive correctness claim is made formally; and (ii) verification - the goal of the analysis is to prove or disprove the correctness claim.

There are two approaches introduced in related works for the verification of OWL-S process models: In [Narayanan et al., 2002] a Petri net-based operational semantics is proposed, which only reflects the control-flow of a process-model. [Ankolekar et al., 2005] additionally models the data flow and applies the SPIN model-checker as an automatic verification tool. As discussed in section 4.3.2, however, OWL-S suffers from shortcomings which make it not the ideal model ontology for the verification of process models.

[Liu et al., 2007] proposes an approach for process models expressed in the Business Process Execution Language (BPEL) that are transformed into pi-calculus and then into finite state machines. Rules captured in the graphical Business Property Specification Language [Xu et al.,

2006] are translated into linear temporal logic. Thus, process models can be verified against these rules that can stem from compliance requirements by means of model-checking. The design of Web Services composition languages, such as XLANG and BPEL, also claims to be based on pi-calculus. However [van der Aalst, 2005] has appealed that more solid work should be done to prove the effectiveness of pi-calculus in modeling business processes. According to [van der Aalst, 2005] the main challenges when using pi-calculus are related to the complexity of the models developed with the pi-calculus in order to express the rather simple workflow constructs that are subjected to the verification.

Very similar to the proposed SWRL-based verification of business processes in this thesis is the work presented in [Stojanovic et al., 2006]. In that work business processes modeled in Ontoprocess-Tool can be stored semantically in OWL and verified using SWRL. The main difference lies in the underlying process model. While the business process model instances created according to CBP ontology in Maestro are executable business processes (by Nehemiah), the processes modeled in ontoprocess are not executable.

5.5 Conclusion

In this chapter we presented an approach and implementation for the verification of business process models. In the context of business process compliance, the verification of business processes lowers the risk of designing and using business processes that have a noncompliant structure according to a set of predefined rules. Verification is used as a tool during the modeling of new business processes or reengineering old ones. It ensures compliance of the process models before their execution and consequently increases the reliability of business process operations.

The verification is automated through the use of formal methods based on ontological representation of process models in OWL-DL and by using SWRL to express the correctness requirements on the structure of business process models. The approach was implemented in the context of the EU-funded project ATHENA. There the verification approach was applied to cross organizational business processes (CBPs) graphically modeled in a business process modeling tool called Maestro. The Maestro modeling tool was extended by the verification mechanism and a user-friendly rule editor to express business level correctness requirements on process models. The business level correctness requirements are internally transformed into SWRL/DL-Safe rules, thus the end-user does not need to have any technical and specifically logic knowledge. The concepts behind the business process according to CBPs were compared to the business process definition introduced in section 5.1.1.2 and the current shortcoming of the approach in terms of possibility to verification of business documents was discussed.

As was previously mentioned, in the context of regulatory requirements such as the Sarbanes Oxley Act, the law requires that some rules/constraints in terms of controls be effective during the execution time of business processes. In such cases a design time approach as introduced in this chapter is not sufficient to satisfy the requirements. Thus, a next step is required to expand the approach so that it considers the runtime of business processes. Such a step, the compliance validation of business process executions, will be presented in chapter 8.

6 Control Model for Business Process Compliance

The notion of controlled entities (CE) in business process compliance was introduced in chapter 4. The following types of controlled entities were identified in business processes: Transition, Business Document, User, Role and Control. Throughout chapter 4 a precise formal model for the first four controlled entities was developed and their relations within a business process identified.

In the current chapter we are concerned with developing a model for the CE of type Control. According to this model a control will be defined on a business process model. The business process model is defined according to Definition 4.2 (BPD). The interplay of the developed model of CE of type Control in this chapter and BPD exposes the semantics of the relation *controls* from BPCD (see Definition 4.1). In this chapter we propose a state model for controls. The state model of control is required for managing the control docoumentation (acting as a business document) in the business process concerned with designing the controls in a company. Consider that such a business process is not an operative business process, such as Purchasing, Sales etc. Regulations such as SOX require not only that controls be assured in the daily operations of business processes, but also that the process of managing controls be well defined. By managing the controls according to the proposed state model a company can effectively prove to external auditors that the company has documented its controls (as required by SOX).

We start this chapter by giving an example of such a state model of a control in section 6.1. We continue in section 6.2 by developing a formal model of controls. This is done in a bottom-up manner in four sub-sections: section 6.2.1 - 6.2.3 provide a set of entities that are used together to formally define the formal model of a control in section 6.2.4. In section 6.3 related works are discussed and followed by the conclusion of this chapter.

6.1 Control State Model

A control is treated as a business document in a business process: it has a certain state life cycle. Regulations such as SOX require that a control itself goes through certain phases, starting at its definition and over to its design, through the way the problems with the design of a control are dealt with, and up to and including its application on a business process and its monitoring. A company has to prove that the current set of internal controls existing in the enterprise have gone through such a well defined process, i.e. the life cycle of a control is managed from its creation, the issues that are identified within the design of a control are handled, and the use of the control in daily operations is both managed and monitored.

In this section we present such a life cycle of the business document *control* in terms of the state model that a control can go through. The concept of the state model of a business document was explained and formally specified in section 4.2.2.1.

Below the state model of a control is formally given according to the formal definition of a business document's state model (see Definition 4.5). The state model is determined based on the following sources:

- Analysis of the mainly non-IT related COSO framework as a de facto-standard for realizing the internal controls compliance recognized by regulation bodies and compliance/auditing experts
- Participation in internal controls compliance projects

- Analysis of commercial software products, such as SAP's MIC-Tool or Oracle's Internal Controls Manager, to promote the management of internal controls projects.

Below, the specification of control state model is presented and visualized (Figure 41), followed by the textual description of its state names and their possible values:

Control State Model:

S = {DESIGN, ASSESSMENT, ISSUE, MATURITY, VIOLATION}
SV = {Scoped, Designed, Evaluated, NotAssessed, Assessed, Effective, Released, Open, Remediation, Closed, Informal, Tested, Monitored}
initialStateValue (DESIGN) = Scoped
initialStateValue (ASSESSMENT) = NotAssessed
initialStateValue (ISSUE) = Open
initialStateValue (MATURITY) = Informal
finalStateValues (DESIGN) = {Evaluated}
finalStateValues (ASSESSMENT) = {Assessed, Effective, Released}
finalStateValues (ISSUE) = {Closed}
finalStateValues (MATURITY) = {Tested, Monitored}
finalStateValues (VIOLATION) = {NotViolated, Violated}
assignedValues(DESIGN) = {Scoped, Designed, Evaluated}
assignedValues(ASSESSMENT) = {NotAsssessed, Assessed, Effective, Released}
assignedValues(ISSUE) = {Open, Remidiation, Closed}
assignedValues(MATURITY) = {Informal, Tested, Monitored}
assignedValues(VIOLATION) = { NotViolated, Violated }

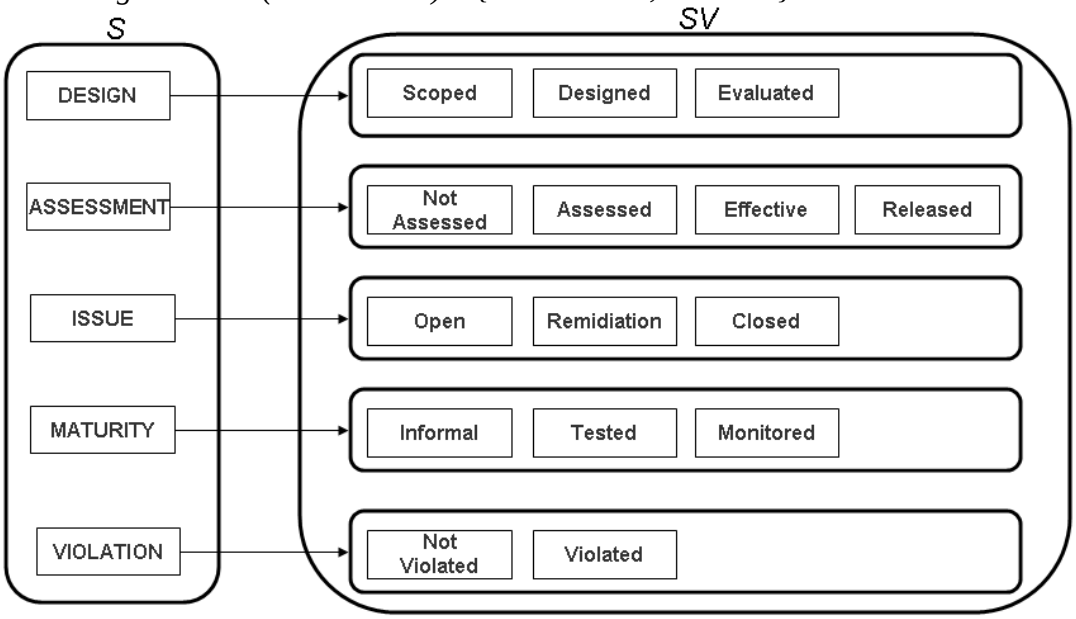

Figure 41 Visualization of a Control State Model according to Definition 4.5

The business level meaning of each state name $s \in S$ and each state value $sv \in SV$ (see Definition 4.5) is described below:

DESIGN: When a control is in the state of *Scoped*, the control has been recognized as necessary to a relevant business process. After a control has been scoped, it is *Designed* by a compliance expert and its design is *Evaluated* by a business process expert.

ASSESSMENT: During Assessment of a control the following facts about a control must be covered:

- Control is documented properly
- Control design achieves the related control objective(s)
- Control mitigates or avoids risk(s)
- Do other controls exist which could achieve the same control objective(s) in a faster or cheaper way?

An *ASSESSED* control is considered as effective, if it permanently satisfies the above four properties. The assessed control will then be released, i.e. become part of the internal controls (Set *CTLS* in Definition of *BPCD* – see Definition 4.1). The assessment of a control can however discover some issues in a control design. When this happens, the control is classed in the *ISSUE* state.

ISSUE: This state occurs when a control design contains any deficiencies. In this case the issue must be remediated. Practically speaking, a control in state *ISSUE* signals that some shortcomings have been discovered within the control, and that those shortcomings were reported when a control was being assessed or tested. A control which falls into this state requires the following documented information in order to be considered as internal controls compliance certified:

- Cause: What causes the shortcoming to occur?
- Implication: What are the implications of the shortcoming?
- Owner: Who is responsible for the remediation of the discovered shortcoming in control design?
- Identifier: Who has identified the shortcoming?
- Identification time: When was the shortcoming identified?
- Priority: Which priority has the remediation of the discovered shortcoming been assigned?
- Status: In which status is the current shortcoming (*ISSUE*-state values: *Open, Remediation, Closed*)
- Remediation plan: Which actions are or will be undertaken to remediate the shortcoming?
- Validation Date: When will the remediation of the control's shortcoming be validated?

The validation procedure of a shortcoming is as follows: After the *ISSUE* state on a control is *remidiated*, the issue will be *closed.* In this case, the control has to be re-assessed (*ASSESSMENT* state). In all other cases an issue is considered as being *open.*

MATURITY: The Maturity state of a control is related to the positioning of a control inside the control environment of an enterprise and to the way its effectiveness is assured. An *informal* control indicates that the control is in place, but has not been documented, systematically designed or assessed and therefore may contain issues. A control which is *tested* has been assessed, and the discovered issues have been remediated, but the control is not *monitored* during daily operations.

VIOLATION: This state is related to the execution phase of a business process, during which the effectiveness of a control is monitored (*MATURITY* state is *monitored*). If a business process instance violates the conditions of a control, the control is assigned the state: *Violated*.

A control can be added to the set *CTLS* in BPCD (Definition 4.1), if its creation and management have gone through the state life cycle described above and if its state name

ASSESSMENT has the value *Released* and its *MATURITY* state has been assigned either *Tested* or *Monitored* value.

6.2 Control Model

Now that we have introduced the state model of a control, we will continue with its internal design and its composition as a controlled entity.

A closer look at the controls presented in the scenario (see chapter 2) intuitively exposes the following model for a control:

All of the controls had an **event**, after which occurrence, during the course of the execution of a business process a set of **conditions** had to hold (or not hold). Such events can have a business level semantic or they can be related to certain points in time, i.e. the beginning of each month. We call the former type of events *BusinessEvents* and the latter type *DateEvents*. Embedded in each control design is the definition of necessary **actions**, which must be undertaken if the conditions of a control fail, i.e. if the control is violated. Keep in mind that this is a different situation from that when the *ISSUE*-state of control is *Open* or *Remidiated* (see control state model in section 6.1). In the latter situation, the control design has some deficiencies (control deficiencies or significant deficiencies, see section 3.1) or even material weaknesses. This means that a control, even if it works as designed, is not able to prevent or detect some risks and fails to fulfill its control objectives. In contrast, actions that have to be undertaken in the case of violations of control conditions relate to the execution time of business processes. This is when the controls are actually applied. In this work, we refer to these actions as ***Recovery Actions***. For each *control*, at least one *recovery action* must be assigned which reacts to the violation of a control during the execution of a business process (a business process instance that has caused a control violation).

Figure 42 represents a control model as described above. Each part of the figure is defined in the following sub-sections in a **bottom-up** manner: We begin by introducing the triggering event-part of the control. After, the models of control condition and recovery actions are each presented in separate sub-sections. Based on the definitions provided in these three sub-sections we specify the model of a control in sub-section 6.2.4.

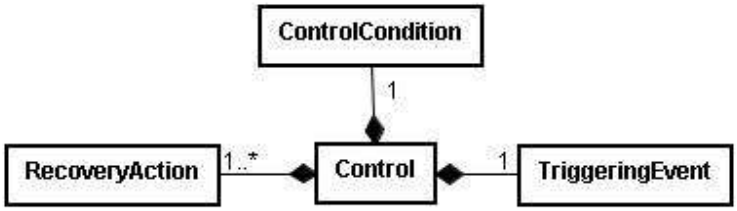

Figure 42 High level overview of Control model

The explanations require the existence of the following functions returning an instance of transition, where *trs* is a transition according to Definition 4.14 and *act* is an activity according to Definition 4.12:

- *model_next (trs)* returns an instance of a transition following immediately after the given *transition trs* in the current business process definition (BPD)
- *model_previous (trs)* returns an instance of a transition immediately before the given *transition trs* in the current business process definition (BPD)

- *model_transition (act)* returns the transition in the current business process definition (BPD), which leads to the invocation of the *activity act*.

6.2.1 Designing the Triggering Event of a Control

In this section we formally define the triggering event of a control, as it was shown in Figure 42.

In order to become active a control must be triggered in a *scope*. It can be triggered at a certain point in time or at regular intervals (*DateEvents*) or it can be activated by the occurrence of business level events (*BusinessEvents*). Capturing and triggering such events is the main issue in achieving the automation of the control process. Therefore, the set *EVENTTYPES* is as follows:

EVENTTYPES = {DateEvent, BusinessEvent}.

DateEvents

We introduce the formal definition of a DateEvent in a control in a bottom-up manner. First, some required basic sets and data types are introduced. Then some definitions are given, based on which the final definition of a DateEvent will be provided in Definition 6.6.

We assume the existence of a data type *Date* in the form *(dd, mm, yyyy)* in which:

- *dd* is a number $\{dd \in N \mid 1 \leq n \leq 31\}$ specifying the day in a *Date*,
- *mm* is a number $\{mm \in N \mid 1 \leq n \leq 12\}$ specifying the month in a *Date* and
- *yyyy* is a number $(yyyy \in N)$ specifying the year in a *Date*.

The recurrence of a *DateEvent* is specified using the *frequency* of the recurrence of the *DateEvent*. Different kinds of frequency can be defined, which are given in the set *FREQUENCIES*. The frequency of a *DateEvent* can be on a daily, weekly, monthly or yearly basis:

FREQUENCIES = {Day, Week, Month, Year}

In addition, the following basic sets are required, where the notation ".." is used as an abbreviation for the rest of elements in the set (which are obvious):

DAYS = {Monday, .., Sunday}
MONTHS = {January , .., December }

Each *Frequency*-type (elements in *FREQUENCIES*-Set) can be specified in a certain way (i.e. configured). Their specifications, together with examples, can be found in the following four definitions:

Definition 6.1: DayConfig

The configuration of daily-frequency $Day \in FREQUENCIES$ is given by $DayConfig = (n)$, where

- n is a number $\{n \in N \mid 1 \leq n \leq 361\}$.

Example: Using the parameter *n* in Definition 6.1, the number of *days* specifying the recurrence of the DateEvent will be given, for instance "each fourth days" with $n = 4$.

Definition 6.2: WeekConfig

The configuration of weekly-frequency *Week ∈ FREQUENCIES* is a tuple *WeekConfig = (n, on)*, where

- *n* is a number $\{n \in \mathbb{N} \mid 1 \leq n \leq 52\}$
- *on* is a total function $DAYS \rightarrow \{TRUE, FALSE\}$.

Example: Using the parameter *n* in the definition above, the number of *weeks* before the recurrence of the *DateEvent* and the *day(s)* in a *week* will be given, for instance "*each second week on monday*" with *n = 2* and *on (Monday) = TRUE* and *on (Tuesday) = FALSE* and *on (Wednesday) = FALSE* etc .

Definition 6.3: MonthConfig

The configuration of monthly-frequency *Month ∈ FREQUENCIES* is a tuple either of the form *MonthConfigDayNumberBased = (n, m)* or
MonthDayInWeekBased = (o, d, m) , where

- *n* is a number $\{n \in \mathbb{N} \mid 1 \leq n \leq 31\}$
- *m* is a number $\{m \in \mathbb{N} \mid 1 \leq n \leq 12\}$
- *o* is a number $\{o \in \mathbb{N} \mid 1 \leq n \leq 4\}$
- *d ∈ DAYS*

Example: Using the *MonthConfigDayNumberBased*- form of monthly-frequency, it is possible to specify a *DateEvent* with a *recurrence* of the form "on the 10th of each second month", which would be represented by the tuple *MonthConfigDayNumberBased = (10, 2)*.

Example: Using the *MonthDayInWeekBased*- form, it is possible to specify a DateEvent with a recurrence of the form "on the second Friday of each month", which would be represented by the tuple *MonthConfigDayNumberBased = (2, Friday, 1)*.

Definition 6.4: YearConfig

The configuration of yearly-frequency *Year ∈ FREQUENCIES* is a tuple either of the form *YearConfigDayInMonth = (n, m)* or
YearConfigDayInWeekInMonth = (o, d, m) , where

- *n* is a number $\{n \in \mathbb{N} \mid 1 \leq n \leq 31\}$
- m ∈ MONTHS
- *o* is a number $\{o \in \mathbb{N} \mid 1 \leq n \leq 4\}$
- *d ∈ DAYS*

Example: Using the *YearConfigDayInMonth*– form of yearly-frequency, it is possible to specify a DateEvent with a recurrence of the form "on each 23rd of December", which would be represented by the tuple *(23, December)*.

Example: Using the *YearConfigDayInWeekInMonth*- form, it is possible to specify a DateEvent with a recurrence of the form "on each third Friday in December", which would be represented by the triple *(3, Friday, December)*.

The recurrence of a *DateEvent* can be constrained using the *duration* of the recurrence of a DateEvent. The definition of the *duration* of a *DateEvent* is as follows:

Definition 6.5: Duration
The *duration* of the recurrence of a *DateEvent* is a tuple either of the form *DurationEndsOnDate = (ed)* or *DurationEndsAfterRecurrence = (n)*, where:

- *ed: Date* specifies the end date of the *DateEvent*
- *n* is a number *(n ∈ N)* specifying the number of recurrences of the *DateEvent*, after which the *duration* of the DateEvent will expire.

It is now possible to specify a DateEvent formally:

Definition 6.6: DateEvent
A *DateEvent* is a data type specified through a triple *(bDate, freq, rConfig, d)*, where:

- *bDate : Date* specifies the beginning date of the *DateEvent*
- *freq ∈ FREQUENCIES*, where the following rules apply:
  - if *freq = Day*, then *rConfg : DayConfig*
  - if *freq = Week*, then *rConfig: WeekConfig*
  - if *freq = Month*, then either *rConfig: MonthConfigDayNumberBased* or *rConfig: MonthDayInWeekBased*
  - if *freq = Year*, then either *rConfig: YearConfigDayInMonth* or *rConfig: YearConfigDayInWeekInMonth*
- *d* is the duration, either of the form *d : DurationEndsOnDate* or *d:DurationEndsAfterRecurrence.*

BusinessEvents

A *BusinessEvent* is an event that defines the boundaries between each business process step in a business process. A step in a business process causes a business document to change its state or a "business-level" activity to be invoked.

Building on top of the process model provided in section 4.2.2.1, the specification of a *BusinessEvent* in a business process is straightforward:

A *BusinessEvent* in a business process is represented as the execution of an *activity* (according to Definition 4.12) or *state change* of a *business document* (see *state change command scc* in Definition 4.13), if certain *conditions* (according to Definition 4.11) are satisfied.

This is equivalent to a *transition* (as defined in Definition 4.14) in a business process definition (Definition 4.2).

Based on the description above, we capture the notion of *BusinessEvents* required for control modeling by the transitions existing in a business process definition and refer the reader to section 4.2.2.1 for a detailed introduction of transitions and accompanying examples.

Regardless of whether the events are of *BusinessEvent* type or *DateEvent* type, an event has to be specified within a *scope*. The scope is basically the extent of the business process execution or time for which the control will be triggered and throughout which the conditions of a control must hold. There are different types of scopes:

SCOPES = {Global, Before, After, Between}

A *Global* scope monitors the entire business process execution. This means that the conditions of a control must always hold during a business process execution. The scope: *Before* monitors

the execution of a business process up to a given event. This means that the conditions of a control will be checked immediately before the given event is executed, and up to the time when the specified event has occurred. The scope: *After* monitors the execution of a business process after the occurrence of an event. This means that the conditions of a control will be checked immediately after the execution of the event. The *Between*-scope monitors any part of the execution from one given event to another event.

The concept of *scope* is inspired by previous research presented by Dwyer at al. in [Dwyer et al., 1999], which will be discussed in the related works section of this chapter (section 6.3.1).

Using the concept of *scope* introduced above and the two different event types, it is now possible to formally specify the event-part of a control:

Definition 6.7: TriggeringEvent

A triggering event in a control is a triple of type *event = (scope, eventtype, events)*, where

- *scope* ∈ *SCOPES*
- *eventtype* ∈ *EVENTTYPES*
- *events* is a tuple *events = (beginEvent, endEvent)* that adheres to following rules:
  - if *eventtype = DateEvent*, then *beginEvent* and *endEvent* are both of type *DateEvent*
  - if *eventtype = BusinessEvent*, then *beginEven* and *endEvent* are both of type *Transition*
  - if *scope = Global*, then *events = (null, null)*
  - if *scope = Before* or *scope = After*, then *beginEvent = endEvent*.

6.2.2 Specification of Control Conditions

Conditions of a control apply to a certain business situation related to the current instance of a business process that requires a special treatment upon its occurence.

In order to formally capture the control conditions we require the notion of *control statements*. Control statements are by nature closely related to the *statements* (Definition 4.10, see section 4.2.2.1) that can be used to express conditions of transitions (see Definition 4.14) and activities (Definition 4.12) in business process definition (Definition 4.12). Modeling the conditions that describe a control violation requires additional types of statements, which are listed in Definition 6.8. The parameters used there have the following types: *trs* is of type *Transition* (see Definition 4.14), *tri* is a transition instance (see Definition 4.15), *rle* is a role, *usr* is a user, *n* and *m* are natural numbers (*N*), *f* ∈ *FREQUENCIES* (introduced in section 6.2.1), *ctl* and *ctl'* are controls (will be formally defined later), and *ce* is a controlled entity.

Definition 6.8: Control Statement

A control *statement* for a control *ctl* on a business process *repository instance BPRI* (according to Definition 4.17) of a *repository BPR* (according to Definition 4.4) can be one of the following:

- a predicate *EXECUTING (trs, rle)*, which returns *TRUE* if the role *rle* is executing the transition *trs* in *BPRI*
- a predicate *EXECUTING (trs, usr)*, which returns *TRUE* if the user *usr* is executing the transition *trs* in *BPRI*
- a predicate *EXECUTED (tri, usr)*, which returns *TRUE*, if the user *usr* has already executed the given transition instance *tri*

- A predicate *EXECUTED (tri, rle)*, which returns *TRUE*, if any user having the role *rle* has executed the given transition instance
- a predicate *EXECUTED (tri, n, m, f)*, which returns *TRUE* if the transition *tri* is executed $n \in$ N- times in the last period specified by $m \times f \in FREQUENCIES$
- a predicate *EXECUTED (tri, fromDate, toDate), which returns TRUE if* the transition instance *tri* is executed on a Date between *fromDate* and *toDate.*
- a predicate *VIOLATED(ctl', n, m, f)*, which returns *TRUE* if control *ctl'* has previously been violated $n \in N$- times in the last period specified by $m \times f \in FREQUENCIES$ in BPRI, i.e. the state value of its *VIOLATION*-State has been *VIOLATED*. Further $ctl \neq ctl'$ must hold
- a predicate *CONTAINS (CES<ceType>, ce*), which returns true if the set *CES* consisting of controlled entities (replace *<ceType>* by either *Control or Role or Transition or BusinessDocument or User*) contains the given entity instance *ce*
- three predicates *SIZE_EQUALS(CES, n), SIZE_GREATER_EQUALS(CES, n), SIZE_SMALLER_EQUALS(CES, n)*, which each return *TRUE* if the number of elements in the set *CES* consisting of controlled entities are respectively: equal to greater or equal to, or smaller than, the number *n*.

Consider that the result of both types of *EXECUTING*-statements return *TRUE* **before** the transition is actually executed by the role or the user specified.

Using the control statements together with the statements introduced in Definition 4.10, the control conditions can be specified in following way:

Definition 6.9: Control Condition
A *control condition* is a conjunction or disjunction of *statement*, (according to Definition 4.10), *negated statements*, *control statements* (according to Definition 6.8) and *negated control statements*.

6.2.3 Recovery Actions of Controls

A control is originally defined by a compliance expert in an enterprise. His main objective is to design the control and to monitor its effectiveness. As we previously mentioned, (see section 3.1.1), a compliance expert has little or no knowledge of the implementation of a business process. The detailed knowledge on how to bring a business process model and its instances into a compliant form/state is the task of a business process expert. The control model for business process compliance in this thesis recognizes this fact by introducing *Role-Based Recovery Action Modeling*. During control design (i.e. after the *DESIGN* state has the state value *DESIGNED*, see section 6.1), a business process expert checks the control (i.e. its recovery action- part) to determine whether it could have a negative influence on the operational effectiveness and efficiency of the business process (Assuring the business objective). After this, the *DESIGN*-state of the control is assigned the value: *EVALUATED*.

Here are the different possible types of recovery actions:

- ***Ignore:*** The control violation is ignored.
- ***Block:*** The current instance of the business process, which generated a control violation, is blocked.
- ***Notify (User, Message):*** A notification message for the specified user: *User* is created with the given message: *Message*.
- ***Retry:*** The activity that generated the violation is repeated.

- ***Rollback (Activity):*** The current instance of the business process that generated the control violation is rolled back to the given activity: *Activity*.
- ***Instantiate (User, RecoveryProcess):*** A previously designed recovery business process *RecoveryProcess* is instantiated parallel to the current instance of the original business process that generated the control violation. The recovery process itself is an autonomous business process. Its task is to remove the conditions that caused the original business process instance to violate the control conditions. The instance of the recovery process is assigned to the specified user *User* in order to enact it.

Note that combinations of the above listed recovery actions are also possible, for example *Retry & Notify, etc.*

In the case of a control violation a compliance expert defines the recovery actions as minimally as possible with regard to avoid influencing the business process logic. The decision of which recovery action needs to be selected in a certain control design is made by the compliance expert. This decision depends on the enterprise-specific risk assessment, which may vary for the same kind of control from enterprise to enterprise. After the control is initially designed by a compliance expert, and includes a recovery action, a corresponding business process expert is notified about the creation of a new control. The business process expert can now review and edit the recovery actions for the control originally designed by the compliance expert.

The valid combination of recovery actions set by the Compliance expert and business process expert follows these basic rules:

- A control violation always requires a reaction, a single *Ignore* in particular is never allowed, since the existence of a control with such a recovery model makes that control meaningless
- The recovery action designed by a business process expert is never allowed to "weaken" the original recovery action designed by the compliance expert. For instance, if a compliance expert requires a *Block & Notify* on a business process instance in the case of a certain control violation, the business process expert is not allowed to redesign the recovery of a control to only *Notify*.

In order to clarify the role-based recovery action modeling we give an example of its application below:

6.2.3.1 Scenario Revisited – Role-Based Recovery Action Modeling

The description of the following situation is visualized in Figure 43. Recall the required control "Minimum Numbers of Suppliers" (control CA3) specified for CustomerA in our scenario (see section 2.2). The compliance expert in that enterprise designs the control according to the risk assessment of the company and decides to select the ***Block & Notify*** recovery action in the case of the control violation. The compliance expert at this stage is not concerned with all the *possible blocked* purchasing process instances (having material type 5 in their *PO* if the number of valid contracts to possible suppliers of this material type becomes lower than 2). This is represented in the step 1 in Figure 43).

During evaluation of the control, the business process expert who possesses detailed knowledge of the Purchase-To-Pay process (see section 2.1) is informed of the creation of the new control (step 2 in Figure 43) and checks the recovery action of that control. Since the business process expert has the business objective "Purchase Goods" in mind, he is aware that some process instances may be completely blocked by that control design, and that this effect is not desirable. Further he is aware of a business process *RfQProcessing*, which creates a so-called *Request for Quotation (RfQ)* from a supplier. The business objective of *RfQProcessing* is to

contract the selected supplier in the Supplier-Relationship-Management (SRM) - system of CustomerA. As a consequence, the business process expert modifies the recovery action model of the control by adding the recovery action *Instantiate (RfQProcessing) & Retry* to the control design (step 3 in Figure 43).If a control violation occurs later on in the execution of the business process, *RfQProcessing* is enacted in parallel, in addition (because of the recovery action *Instantiate*) to the current P2P Process instance. The process step is retried again (because of recovery action *Retry*) and, if the control violation no longer exists (perhaps because *RfQProcessing* has increased the number of contracted suppliers in the backend system SRM to 2 or more), the process instance can continue. The latter explanations are not illustrated in the figure because it relates to the execution time of business processes. We are concerned with the designing of controls in a business process model.

Consider that the application of the above strategy would eliminate the necessity for the integration of the *RfQProcessing*-sub-process in the *Purchase-Request*-sub-process as was necessary in the scenario in the case of CustomerA (see section 2.2.5).

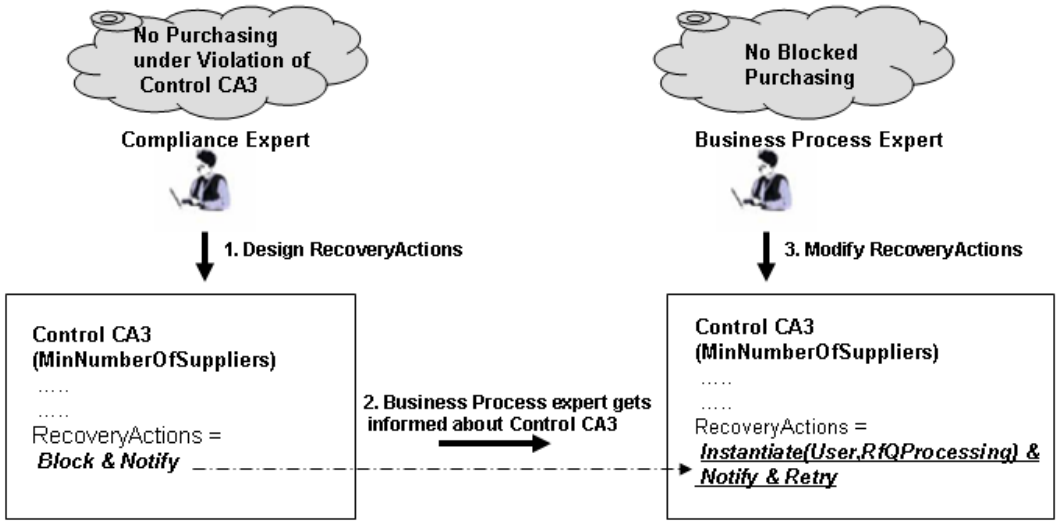

Figure 43 Role-Based Recovery Action Modeling exemplified

6.2.3.2 Definition and Application of Recovery Actions

Based on the introduction above, the application of the proposed *recovery actions*, according to the formal model of a business process (BPD, see Definition 4.2), is given.

The implication of the application of a designed set of recovery actions in a control definition has an impact on the way the *transitions* (see Definition 4.14) are invoked in a business process instance. Some recovery actions may fulfill additional tasks, such as sending a message to a particular user (*Notify* recovery action) or instantiating an autonomous business process instance (*Instantiate*), which then removes the conditions in a system which were responsible for the control violation in question.

Definition 6.10: RecoveryAction
A RecoveryAction for a business process bp is an expression of the form

$$tMod \otimes job$$

in which:

- *tMod* is an expression of the form *i_next(trs) = trs'*, where

trs and *trs'* are each a transition in the *bp* model,
the function *i_next(trs)* sets the next *transition* that will be taken in of the current instance of *bp* on the *transition trs*.

- *job* represents the invocation of an activity *act* in a transition of the form *if TRUE then invoke act* (see Definition 4.12). The existence of *job* for the specification of a recovery action specification is optional. However, if *job* is specified then the activity *act* will always be invoked.
- ⊗ is an operation that causes the parallel execution of *tMod* and *job*.

The definition above is applied to each type of *recovery action* using the introduced functions *model_next (trs), model_previous (trs)* and *model_transition(act)* in section 6.2. The formalization is as follows:

Igonore*:*

i_next(t) = model_next(t) ⊗∅

, where ∅ means there is no activity required in the *job*. The *tMod*-specification of *Ignore*- recovery action sets the next transition in the current business process instance to that transition that was originally modeled in the business process definition (determined by *model_next(t)*). In this case the business process instance continues to execute as originally designed, i.e. the control violation is *ignored*.

Block:

i_next(t) = t ⊗∅.

Notify (User, Message):

i_next(t) = model_next(t) ⊗NotificationActivity,

, where

Activity *NotificationActivity*

- *name = "NotificationActivity"*
- BDI_{read} *={usr:User, msg:Message}*
- $BDI_{\mathrm{mod}\,ify}$ *= {usr:User}*
- *P = TRUE*
- E_{create} *= ∅*
- $E_{\mathrm{mod}\,ify}$ *= ASSIGNED (usr,Message).*

In the above specification an employee who is notified about a control violation is represented as a business document (*User*) in the system. The message box of the employee that is changed by invocation of the activity is one of the possible *attributes (A)* of the *User* business document (see Definition 4.6).

Retry:

i_next(t) = model_previous(t) ⊗∅.

Rollback(Activity)**:**

i_next(t) = model_transition(Activity) ⊗∅.

Instantiate(RecoveryProcess, User)**:**

i_next(t) = model_next(t) ⊗InstantiationActivity,

, where

Activity *InstantiationActivity*

- *name = "InstantiationActivity"*
- BDI_{read} *={usr:User, prc:RecoveryProcess}*
- BDI_{modify} *= {usr:User}*
- *P = TRUE*
- E_{create} *= NEW (prc, RecoveryProcess)*
- E_{modify} *= ASSIGNED (usr, RecoveryProcess).*

6.2.4 Specification of a Control for a Business Process

Based on the definitions introduced in the previous three sub-sections we shall now introduce the formal definition of a *Control* for a business process:

Definition 6.11: Control

A *control* for a business process definition *bpd* is a tuple *ctl = (cbdt, event, cc, RAS)*, with:

- *cbdt* specifying the *business document type* according to Definition 4.6 of the control with the following *header attributes*:
  - *bpd*, the business process for which the control exists
  - *risk*, that the control must mitigate
  - *account*, the entry in the general ledger, which the business process is relevant for
  - *co*, the control objective of the control
- *event* is a tuple *e = (scope, eventtype, events)* according to Definition 6.7
- *cc* is a *control condition* according to Definition 6.9
- *RAS* is a non-empty set of *recovery actions* according to Definition 6.10.

The state model of the business document *cbdt* in the definition above can be found in section 6.1 (*Control State Model*).

Consider that if a control contains an event of type *BusinessEvent*, the statements used in a control condition (Definition 6.9) may be the same as the statements of conditions (Definition 4.11) occurring in the transitions of a business process definition. But they have different purposes and meanings: while the conditions in a transition of a business process definition have as their main purpose to describe the process flow (which conditions must hold for the progress of the process instance), the control conditions in a control describe the parameters that cause a violation of that control. In the latter case a recovery action must be instantiated and applied to the current instance of a process model. While in most cases the conditions of a control must be different from the conditions in a transition, this is not formally required in our proposed control model. The possibility of separate modeling of conditions in a control and transitions in a business process raises the modeling approach's level of flexibility. This flexibility is achieved by differentiating between business and control objectives in business processes (see section 2.4.3).

6.3 Related Work

6.3.1 On System Specification Properties

Our definition of the two elements of a control *scope* and the *control statements* were conceptually based on work done by [Dwyer et al., 1999]. They have analyzed over 500 examples of program requirement properties and found that nearly all conformed to eight temporal property patterns within five scopes. Although their patterns are used for defining formal requirements on program specifications, they can be applied to internal controls compliance and their monitoring requirements. Indeed, the controls on an operative business process which we deal with in the course of this work are for the most part technically reflected on an implementation level in the form of *IF-ELSE*-Statements in program code. The goal of the Dwyers system specification properties is to present these kinds of system properties. Thus, we argue that they are very well suited to applications in the design of controls for business process compliance.

Beyond different kinds of scope in the event-part of a control (see section 6.2.1) the concept of different variants of *EXECUTING*, *EXECUTED*, *CONTAINS* and *SIZE_EQUALS* control statements used in the control conditions are inspired by the patterns of system specification properties presented in [Dwyer et al., 1999]. They present the following patterns:

- *Absence* describes that the defined scope is free from state P
- *Existence* describes that a state P occur within the scope
- *Bounded Existence* describes that a state P must occur k times within the scope
- *Universality* describes that a state P is true throughout the scope
- *Precedence* describes that a state P must always be preceded by state Q in the scope
- *Response* describes cause-effect relationships. An occurrence of the state P must be followed by an occurrence of state Q.
- *Chain Precedence*: a sequence of state must always be preceded by sequence of other states in the scope
- *Chain Response*: a sequence of states must always be followed by a sequence of other states in the scope.

We are able to present the patterns above using different kinds of statements and control statements. Dwyer patterns are widely adopted and applied in different contexts and in other research [Li et al., 2005] [Robinson, 2005]. As stated before the experiments and empirical research in [Dwyer et al., 1999] have shown that the scopes and patterns are expressive enough to represent different kinds of system requirements, and thus we argue that the control statements are able to express the control requirements on operative business processes.

For a detailed description of the Dwyer scopes and patterns and their semantics, please refer to [Dwyer et al., 1999].

6.3.2 On Exception Handling in Business Processes

The proposed model of recovery actions is closely related to the concept of exception handling in software applications in general and in particular to those in workflows.

Although exception handling is not explicitly a core component of internal controls, since COSO [COSO92] does not explicitly state how to do exception handling in an internal control process, our study argues for the requirement of the definition of an explicit exception model as part of our proposed model for internal controls. Namely, COSO proposes in its component "Control activities" that "exceptions should be acted upon and reported if necessary". Further in its component "Monitoring" COSO requires "spotting quickly on significant inaccuracies or

exceptions to anticipated results" and states that the "effectiveness of the internal control system is enhanced by timely and complete reporting and resolution of exceptions". We consider the term exception in following since they arise when a rule is broken. The term exception is used by IT- and Accounting experts for semantically the same thing (in context of compliance).

At this point we would like to discuss the related work in the area of exception handling: In [Russel et al., 2005] a classification framework for exception handling for workflows is offered. They determine a comprehensive range of exceptions that are capable of being detected and provide a useful basis for recovery handling and resolution of exceptional situations. Additionally, their research found that there are three different possible recovery actions in the context of workflow exception handling: no action, rollback and compensate.

From the point of view of realization, we see the resolution of exceptional situations detected by internal controls during runtime of processes as being well within the context of software system error recovery [Lee et al., 1990]. There are two main strategies of error recovery: backward and forward error recovery. Backward error recovery is based on rolling system components back to a previous correct state. Forward error recovery transforms the system components into any correct state. This is mainly the same result as [Russel et al., 2005] reported for workflow management systems. According to [Christian, 1989], backward error recovery has a limited applicability and modern application systems involving human beings, Commercial-Off-The-Shelf (COTS) components, external devices and several organizations rely on forward error recovery. As compliance management of internal controls clearly falls within these categories of applications, we propose an adapted forward error recovery strategy performed at the application level by a business user and not at a technical level by a system administrator. A business expert in charge in a push-driven manner is notified with a request to resolve the exceptional situation (see for instance *notify*- or *instantiate*- recovery actions). After motivating the forward error strategy we come to conclusion that its usage is the preferred exception handling mechanism for internal controls compliance in business processes.

[Charfi et al., 2004] uses Aspect Oriented Programming (AOP) techniques to extend the functionality of a BPEL process with additional activities. This is closed to the idea of separating internal concerns from business processes management. We agree with the argumentation given in [Charfi et al., 2004] that there are several concerns, in particular from our point of view the regulatory requirements such as SOX, in a business process management life cycle that have to be separated from process designs. However the work which uses the AOP technology addresses the implementation level of business processes. A control can be implemented using AOP.

[Giblin et al., 2006] provides temporal rule patterns for regulatory policies, although the objective of that work is to facilitate event monitoring. A conceptual model based on UML Profile is defined as a basis for defining compliance rules. But the work does not explicitly state how to reason over the UML Profiles and how they may be related to the business process model and execution levels within enterprises.

[Governatori et al., 2006] uses a logic-based formalism called Formal Contract Language (FCL) [Governatori et al., 2005] to describe business contracts on business processes. FCL could be used to implement controls on business processes. However, the work in [Governatori et al., 2006] uses BPMN as a target platform for applying the FCL-statement and does not state how the FCL-statements on BPMN process models are related to the execution of business processes, during which the controls actually have to be checked. We would require this specification in order to support a preventive nature of business process compliance in daily operations. The objective of the presented control model in this chapter is to validate the compliant execution of business processes according to the condition of a control, if and when its triggering event becomes valid. We intentionally do not bind the control model presented in this

chapter to a certain logic like FCL, because we believe that this step should be relegated to the implementation level of a control.

6.4 Conclusion

This chapter provided a precise formal model of controls that can be designed on a business process. The model builds on top of the controlled entities in a business process that were introduced and formalized in chapter 4.

A control has the following structure:

- A Triggering event, which is bounded to a certain scope. There are two different types of events: DateEvents and BusinessEvents. The latter has a business level meaning. BusinessEvents are captured in the control model by the occurrence of a certain transition in a business process.
- Control condition describes which conditions must be satisfied by a business process during the scope of its triggering events. A set of control statements were introduced for describing the control condition. They can be used together with statements (see Definition 4.10) to formulate the control condition of a control.
- A set of recovery actions that will be invoked if the control condition is violated in the scope of its triggering event. Different types of recovery action were introduced that can be combined together in a control.

Several types of control statements and the concept of scope are inspired by System Property Specification Patterns [Dwyer et al., 1999].

This chapter also provided a state model of controls that can be used during design and management of controls.

7 Pattern-Based Design of Controls in Business Processes

The control model introduced in chapter 6 serves as the input for the implementation of the controls in business processes. Since the conceptual model presented is rather formal and technically oriented, its usage would be too difficult and hard to understand for compliance experts. For this reason, a pattern-based approach for the definition of the controls on business processes is proposed in the current chapter. The patterns should simplify the design of controls for non-technical persons by providing a high-level-language for internal controls. The patterns of controls on business processes will be mapped onto the proposed model of a control (Definition 6.11).

This chapter is organized in the following way: First, the motivation for using a pattern-based-approach for control design is given in section 7.1; then the nature of a control pattern is discussed and set in context to related work on pattern-based approaches for conceptual modeling (section 7.2). Based on the discussion, the attributes that describe a control pattern are introduced. Based on the structure of a control pattern, the formal definition of a control pattern is given in section 7.3. Section 7.4 introduces the set of control patterns identified in this thesis. In that section the formal model of control pattern is applied to each control pattern in the repository. In section 7.5 we introduce the notion of control pattern instantiation. Control pattern instantiation is a procedure that generates the control condition of a concrete control based on a given control pattern. Section 7.6 concludes this chapter.

7.1 Motivation for Using a Pattern-Based Approach for Control Design

A violation of controls represents an exceptional situation in the enactment of business operations as manifested in operative business processes. The methodology presented here is inspired by the observation that similar controls are often defined for mitigating certain risks, which need to be managed in a similar fashion. The same risks may occur in different business processes. For instance unauthorized or unapproved enactment of business level activities is a certain type of risk that may occur in several business processes. Business level activities in a purchasing process are, for instance, approval of an internal purchase request "*ApprovePR*" or selection of a supplier "*SupplierSelection*" (see section 2.1). Both activities can be subject to the risk of *internal misuse*. Typical controls that are used in practice to mitigate these kinds of risks are the application of the "*4-eyes-Principle*" or separating the duty of enacting a certain business level activity among different users or roles (*Separation of Duties- SoD*, see section 2.3). Another example of a risk that occurs frequently in business processes is that received goods or services are not in line with the order originally sent to a business partner. A concrete example of this in a purchasing process is the situation that the received goods have a different quality or quantity than requested. Furthermore, a certain situation in a business process may represent a risk, for instance, a situation where a goods shipment is received from a supplier other than the one to whom the purchase order was originally sent. This is possible when the original supplier uses a subcontractor to fulfill the order. Typical controls to mitigate the possibility of fraud in such constellations are to *compare the business documents* produced as result of the execution of different activities in an instance of a business process. In such a way the situation representing the above described risk can be determined. Applied concretely to the purchasing process, a control that compares the business documents *Goods Receipt, Purchase Order* and *Purchase*

Request with each other (including their attributes) can detect any undesired mismatch in the quantity or quality of the received supplier shipment (*3-Way-Match* – see section 2.3). In addition such a control would detect any fraudulent situations with respect to the current business partner (Supplier identification, its address or bank information are not identical on Goods Receipt - and Purchase Order- Business documents).

These observations led to the idea of taking advantage of repetitive patterns in control design, in order to reduce the modeling effort and provide the compliance experts with reusable process knowledge. The result is a set of patterns of internal controls on business processes. Each pattern acts as a generalized description of actions that are frequently used in mitigating similar risks. They define typical rules or set of rules that capture the knowledge about the occurrence of a situation that violates a control and about the actions that can be performed to handle the violation.

Taking the perspective of a standard software provider, providing this set of patterns in a repository, where a certain pattern can be selected, instantiated to a real control, and applied to business processes brings a higher level of system and component reusability to the ERP/BP products. Taking the perspective of a customer company, building their compliance on top of such a pattern repository can reduce the required domain specific knowledge in compliance projects.

7.2 Analysis and the Structure of a Control Pattern in Business Process Compliance

The control patterns provide the basis for the terminology in which the compliance experts communicate about the domain. We have determined the set of control patterns, which will be presented shortly, *empirically* by analyzing following kinds of popular ERP business processes:

- Purchasing,
- Sales and
- Human Resource Management.

In addition, the corresponding side-processes (such as for example Goods Return, Payment, Dunning, etc. in case of Purchasing) were in the focus of the analysis. The information used for analysis was provided by the consulting companies Deloitte and PriceWaterhouseCoopers. In a contribution to a part of the SAP's internal controls documentation tool called MIC (Management of Internal Controls) [MIC], they describe a best practice model for the above business processes (including all necessary controls). The business processes, including the control proposals, act as documentation and are not related to the implementation level of business processes. The best practice recommendations are not enterprise or industrial-sector specific, meaning each company using the best practice recommendations can select a sub-set of recommended controls according to its enterprise-specific risk assessment.

Our analysis of the different process descriptions including the controls that were provided by Deloitte and PriceWaterhouseCoopers resulted in a categorization of controls, which then represents the proposed control patterns. Based on this analysis, we present in Figure 44 the different categories of control patterns (Control Pattern Repository). We recognize that this is not a *complete* list of possible control patterns that may be required in practice. However based on the auditing know-how of the auditing companies involved in the description of the analyzed business processes mentioned above, we believe that the presented patterns reflect a large part of possible controls in practice. There follows a brief description of each pattern category type,

without details of its sub categories. A detailed description of each pattern will follow in section 7.4.

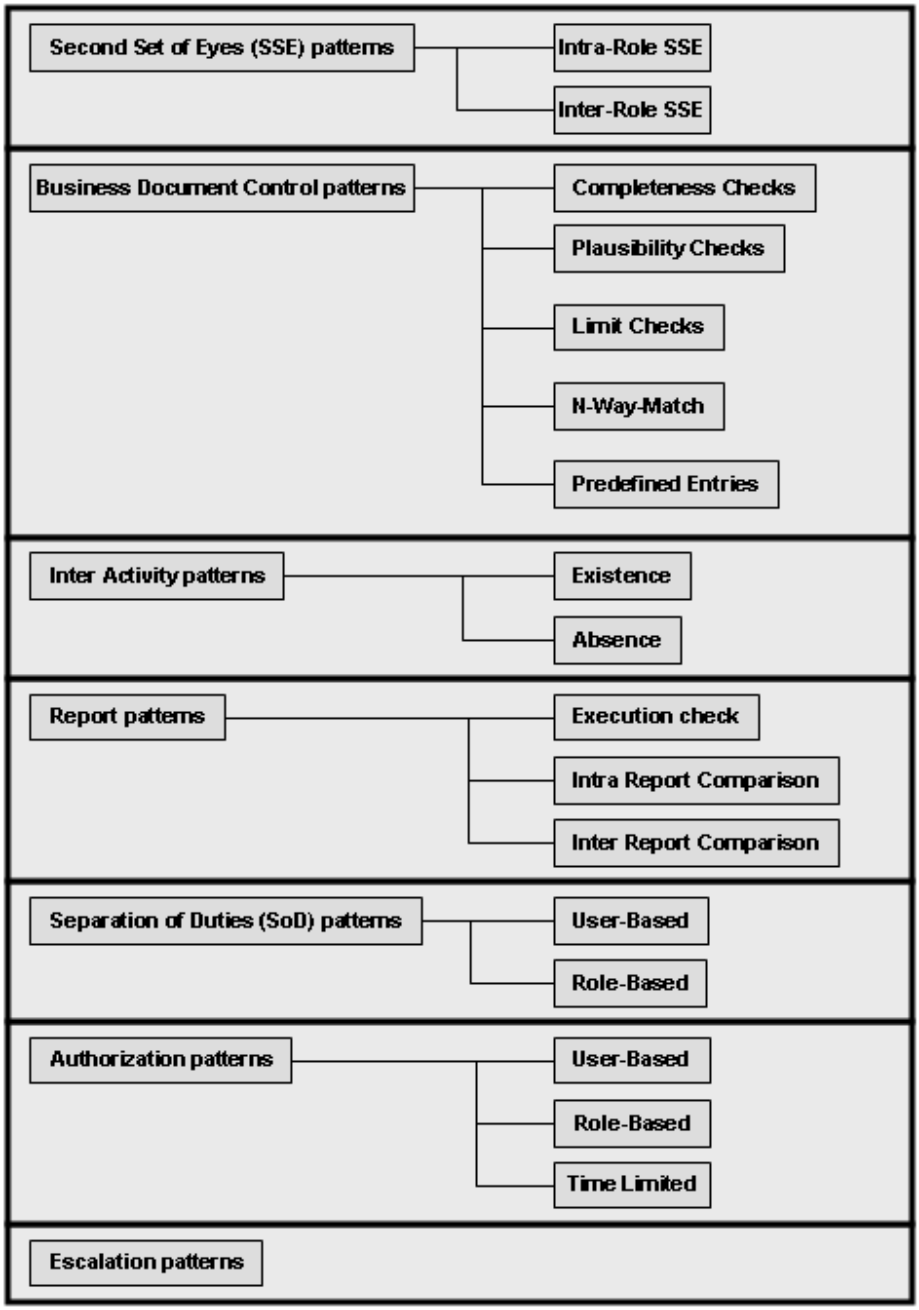

Figure 44 Control Pattern Repository

SSE patterns: We already mentioned this kind of control pattern briefly in the scenario section, where certain transactions were shown that required the SSE-principle. Here we add the comment that a control demanding a "higher number of eyes" would also be possible and would fall into this category as well.

Business Document control patterns: Here the syntax and semantics within and between different business documents are subject to controls.

Inter Activity patterns: The controls satisfying these patterns require that certain activities occur (or are absent) if a set of other activities occur in a business process (or a side process).

Report patterns: Reports are collected based on attributes on certain types of activities and business documents in an enterprise during a certain period, e.g., monthly turnover reports. The purpose of report control patterns is not the definition of a report, but rather to control that a report has been generated and/or the respective reports are compared to each other as required in the control.

Separation of Duties (SoD) patterns: In order to minimize fraud or misusage it is required that an activity is divided into sub activities and each sub activity is executed by different users or roles.

Authorization patterns: These controls limit users/roles access to resources.

Escalation patterns: If control conditions are ignored by the responsible users, this fact can/has to be escalated to responsible entities in the enterprise.

A detailed description of each pattern type and its subcategories will be given in section 7.4. The idea is to provide for each control pattern a corresponding mapping to a control according to a control model (Definition 6.11). The control model introduced there represents a more technical view on the controls and its introduction is aimed to facilitate the use of formal methods by system developers/technical personnel who have the task of implementing the controls in ERP/BPM Systems. The control patterns and the control model are kept implementation-independent in their nature in that they are not bound to the usage of a certain system-specific implementation. Each development team can select its favorite and suitable technical representation of the controls, which can vary from database-oriented/SQL to a temporal logic such as LTL (see Figure 45).

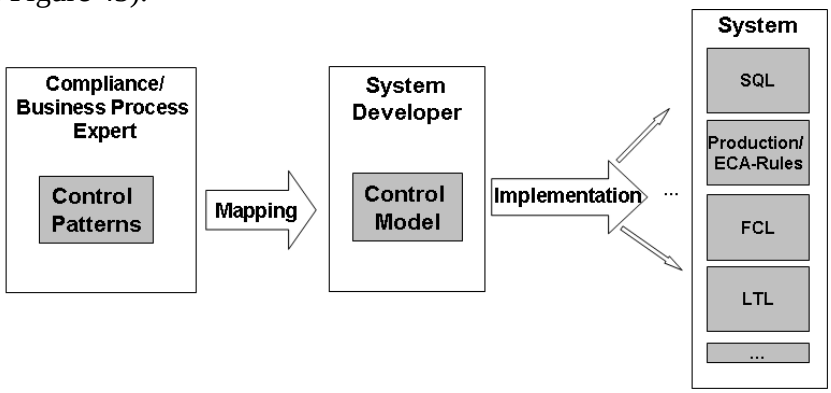

Figure 45 From a Control Pattern to its technical implementation in a system

The description of the control patterns follow the spirit of pattern-based design, which originally aimed to provide a reusable approach to solve a recurring problem instance in a certain domain.

Christopher Alexander, a well-known building architect is widely acknowledged to be the originator of the pattern idea [Alexander, 1979]. He explains that “each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution”. Further he adds that “as an element in the world, each pattern is a relationship between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain spatial configuration which allows these forces to resolve themselves” [Alexander, 1979].

The usage of patterns in software applications in general became popular by the introduction of object-oriented design patterns known as the Gang-of-Four (GoF)-patterns [Gamma et al., 1995]. Control patterns for realizing business process compliance are introduced using an analogy to software design patterns in object-oriented systems. Software Design patterns are "descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context" [Gamma et al., 1995]. In contrast, control patterns are descriptions of rules, predefining certain conditions on the occurrence of certain events during the execution of a business process.

Significant research exists on the modeling of control flow in business processes by using patterns to identify commonly used constructs [www.workflowpatterns.com].

On a similar note, [Giblin et al., 2006] provides temporal rule patterns for regulatory policies, although the objective of this work is to facilitate event monitoring rather than the usage of the patterns for support of compliance in business processes.

Significant work has been contributed in [Casati et al., 2000] for pattern-based exception handling in workflows, which we consider as highly related to our pattern-based approach. Especially the proposed algorithm for pattern specialization in [Casati et al., 2000] can be reused and applied to the control patterns proposed in our work.

However, control patterns and the patterns mentioned above have in common that both are intended to give some guidance on how a *problem* can be solved by using the concepts of an underlying *model*. In the case of business process compliance the *problem* is the occurrence of a potential *risk*. The underlying model in the case of control patterns is the model of a business process introduced in chapter 4, while in the case of software design patterns the model is the object-oriented model. By applying the idea of patterns to business process compliance the reusability of a design is facilitated as well. In the case of control patterns, these reusable designs are an abstract means to capture a certain kind of compliance requirement in a generic and thus system independent manner. The compliance expert is relieved from the task of capturing compliance requirements in controls in a recurring manner.

A pattern language is provided using a certain predefined form for each pattern description. A pattern description is constituted using certain attributes in the form that build the pattern language. Several pattern forms exist in the literature, each one differing from the other in the categories of attributes they emphasize in the pattern description. Among others there exist the Alexandrian form [Alexander, 1979], the GoF form [Gamma et al., 1995], and the Coplien form [Coplien, 1995]. All forms contain the basic attributes to specify a pattern: name, problem statement, context, description of forces, solution and related patterns. The attributes to specify a control pattern proposed in this work are aligned with these common attributes. The attributes and their descriptions for pattern specification are:

- **Name** of the pattern: Since the *name* of the pattern should become part of the vocabulary of the community, it should be easy to remember and refer. The name must be intuitive in the sense that it gives an image of the intent of the pattern.
- (optional) A (nested) list of **super type categories** of the given pattern, in order to identify the pattern in the pattern repository

- **Risk situation**: The risk situation in a control pattern becomes the problem statement part of the pattern. The aim of the control pattern is described, as well as how it can be used for mitigating certain risk situation(s).
- **Control Objective(s)**: The control objective types (Operations, Financial Statement or Compliance) for which the control pattern can be used.
- **Solution**: How the control can be used to mitigate the risk. Warnings about the pitfalls of using the pattern should also be given. (how does this pattern become an *Anti-Pattern*) Additionally, this attribute should reference variants of the pattern.
- (optional) **Related to**: the possible dependency links between different types of control patterns.
- (optional) **Example**: a concrete control in the Purchasing process that follows the given pattern.

7.3 Control Pattern Formalization

Using a pattern means instantiating its abstract description into a control model according to the formal definition of a control in chapter 6 (see Definition 6.11). Considering each control pattern description given and the formal definition of control model, reveals that while all control patterns have many aspects in common, each pattern requires certain *pattern-specific parameters*. A pattern-specific parameter is an element of a control pattern, which is always required in order to capture the information required for the definition of a control according to the pattern. A pattern-specific parameter will be reflected in the control condition in the corresponding concrete instance of a pattern. For instance consideration of the *SoD-Pattern* reveals that the specification of a control according to this pattern requires at least a set of transitions, whose enactment must have been separately done by different users or roles (depending on whether user or role-based SoD is intended by the compliance expert, see Figure 44). But taking the *Escalation pattern* for instance, the design of a control according to this pattern requires the selection of a concrete control (possibly from a control repository) and the selection of the number of ignored violations, after which the recovery action (see Definition 6.10) of the control will be invoked. To simplify the design process of controls required for business process compliance, *pattern specific parameters (PSPs)* for each of the patterns types shown in Figure 44 have been identified, which are included in a pre-defined template for designing a control according to a certain control pattern.

The application of *PSPs* during pattern-based control design is either reflected in the *control conditions* (Definition 6.9) of a control or in its *triggering event* (Definition 6.7) or in both parts. We remain generic with *recovery actions* (Definition 6.10) of a control, because the decision on which recovery action to select is enterprise-specific and needs to be determined by compliance experts. This means that the compliance expert is allowed to specify any recovery action-type presented in section 6.2.3 during instantiation of a control pattern into a concrete control according to Definition 6.11 (its *RAS*-part).

Using the notion of *PSPs* and the attributes specifying each pattern (see section 7.2), the formal definition of a control pattern is given below, where the notation *[X]* means that the existence of *X* in an entity defined according to the tuple in the definition is optional:

Definition 7.1: Control Pattern (cp)
A control pattern is a tuple *cp = (pName, pRisk, pCO, pSol, [pREL], [pEx], pCat ,[psp]),* with:

- *pName* is the name of the control pattern

- *pRisk* is the description of the risk situation in which the control pattern can be applied
- *pCO* specifies the control objective types of a cp with *pCO* $\subseteq$ *{Financial Reporting, Operations, Compliance},*
- *pSol* is the textual description of the approach that can be applied to mitigate the risk situation given
- *pREL* is a set of other control patterns to which a *cp* is related (see attribute "Related to" introduced in section 7.2)
- *[pEx]* An example usage of the control pattern
- *pCat* specifies the categorization of the control pattern as a pair *pCat = ([supC], [subC+]),* where *supC* defines the name of super-category and *subC* defines the sub-category of *cp (Notation X+* means that *X* occurts *0* to *n* times)
- *psp* defines the possible pattern specific parameters of *cp*.

The formal specification above intentionally leaves the formal description of pattern specific parameters (*psp*) open due the diversity of the parameters for each control pattern type. Pattern specific parameters for each of the proposed control patterns will be given in section 7.4.

7.4 Control Pattern Repository

A brief description of each pattern type given in Figure 44 was given in section 7.2. In this section the current content of the control pattern repository is further explicated and specified. This is done by the description of the business level usage of the currently provided control patterns and the introduction of pattern-specific parameters (PSPs) for each control pattern.

SSE Patterns

- **Name:** Second Set of Eyes (Also known as 4-Eyes-Principle)
  Risk situation: Financial mis-statements can be made either through the intentional fraudulent misuse of resources and transactions by internal employees or unintentionally through incorrect business decisions.
- **Control Objective(s)**: Financial Reporting, Operations
- **Solution**: Set up an approach for business related activities involving financial transactions, so that **at least** two people sign off on each activity independently of each other. Select the number of such activities and the people involved carefully. A high number of activities under the control of SSE can reduce the operational efficiency of the business process. An activity can block the progress of the business process, if it is subject to SSE and the grant for enacting that activity is revoked later. An activity can block the progress of the business process, if it is subject to SSE and at the same it is subject to SoD, where the role or (one) of the roles (in the case of Inter-Role SSE) does not appear in the list of roles in SoD.
- **Related to**: SoD, Authorization
- **Example**: Approving high volume purchase requests and orders or those related to material types not ordered for a certain time or period should be done by two different employees.

There are two variants of the SSE control pattern: For Intra-Role SSE it is sufficient that employees executing a transition under the control of SSE have the same role, whereas Inter-Role SSE requires different roles for each employee.

Pattern specific parameters of this control pattern are given below:

$PSP_{SSE} = (trs, n, RLS)$, *with*

- *trs* is the transition under control
- $n \in N$ is the number of users (number of eyes), where $n \geq 2$ and
- $RLS \subseteq ROLES$ is the set of roles of users, who execute *trs*.

The following example will show that the pattern specific parameters given above can be used for both types of SSE-control patterns.

Example 7.1: Application of pattern specific parameters for SSE control patterns
In Figure 46 we give an example of the pattern specific parameters of two different controls adhering to two different SSE-control patterns. As can be seen, using pattern specific parameters in pattern-based design of controls will further simplify control design for business process compliance. This is due to the fact that it is possible to support compliance experts with predefined UI-templates for each control pattern. The area in figure marked with GP (Generic Parameters) can occur in any type of control patterns, thus they are generic and not further specified. They are technical control conditions (Definition 6.9) and will be manually added by a compliance expert in order to customize the control.

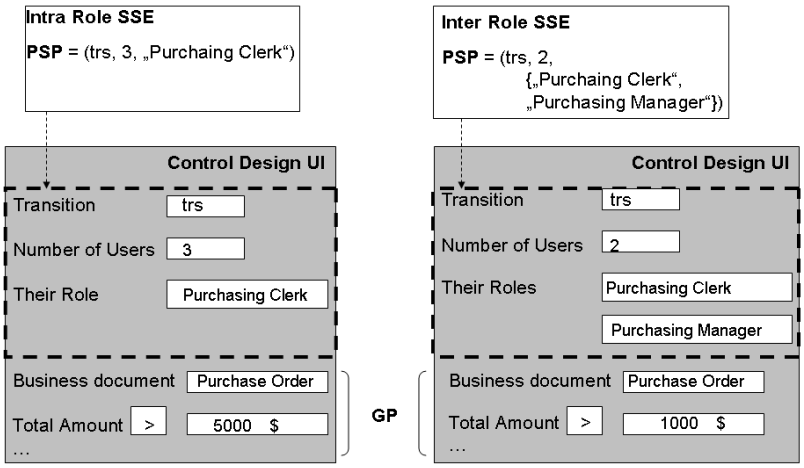

Figure 46 Application of PSP for SSE

Inter Activity Control Patterns

- **Name:** Inter Activity Control Pattern
- **Risk situation**: Operational effectiveness and efficiency of a business process becomes negatively affected, if upon occurrence of a certain unexpected business related situation no appropriate reaction to its occurrence is predefined.
- **Control Objective(s)**: Operations
- **Solution**: Identify an unusual unexpected business situation in a business process (or a side-process of it). In case this unexpected business situation occurs the control checks whether:
    - another business event, whose occurrence is contingent on the occurrence of the unexpected event, has occurred as well (possibly after some elapsed time), (Inter-Activity- Existence) or

- another business event has not occurred (possibly after some elapsed time)

- **Related to**:
- **Example**: A sample usage of the Inter-Activity-Existence pattern is the *substitute concept* for users with the role Approver in a purchasing process. The control requires the assignment of a substitute approver within the first day of absence of the originally assigned approver.. All approving tasks of the absent approver will be rerouted to the substitute employee. This control ensures that all purchasing orders will be processed in a timely manner, if an employee with a key responsibility falls out.

The Pattern specific Parameters for both types of this control pattern are identical.

$PSP_{Inter-Activity}$ = *(trs, trs', n, m, f), with*

- *trs* is a transition
- after the occurrence of *trs*, the transition *trs'* has to occur as well (respectively not occur in case of an Absence Pattern) ($trs \neq trs'$)
- $n, m \in N$ and $f \in FREQUENCIES$ specify the following: *n* is the number of invocations of *trs'* (in the case of an Existence pattern at least n times, in the case of an Absence pattern at most n times); in the last m frequencies (for example $m = 3$ and $f = Week$ specifies *3 weeks*).

Business Document Control Patterns

- **Name:** Business Document Control Pattern
- **Risk situation**: Business documents produced (probably by fraudulent activities or software errors) could result in undesired activities or results.
- **Control Objective(s)**: Financial Reporting, Operation
- **Solution**: Check the content (syntax and semantic) of a business document instance or compare several instances of different business document types in a business process to determine whether a business document is complete, accurate and valid. There are several sub-types of this control pattern:
  - Completeness Checks: Verify whether all mandatory data-fields (attributes) in a business document instance are filled with a correct data type.
  - Plausibility Checks: Verify whether the attribute values of a business document instance is plausible.
  - Limit Checks: Certain attributes of different business documents instances belonging to the same business process instance are bound to a mathematical relationship.
  - N-Way-Match: The value of certain attributes of different business documents instances belonging to the same business process instance must match each other.
- **Example**:
  - Completeness Check: The number of items in a *PO* business document instance must be the same as the number of items in the corresponding *invoice* document received from a supplier, if invoice splitting is not indicated.
  - Plausibility Check: The PO Creation date is not after PO Approval Date
  - Limit Check: Check availability of requested material in warehouse without assignment: Do not accept purchase orders with a material type in their items, where the current amount of that material type still available in the warehouse (in stock) is higher than $10000 and the amount of the order is lower than $1000 .

- 3-Way-Match: Check whether Purchase Order, Invoice and Delivery Business Documents of a purchasing process instance have the same supplier identification and purchase request identification.

Due to the diversity of different business document control patterns, there are no pattern specific parameters provided. The controls can be modeled in a generic way using the general control conditions-part in a control as introduced in section 6.2.2.

Report Patterns

The aim of report patterns is not the definition of the reports, but rather to check their generation in the system by required users. A report is a special type of Activity with no preconditions and effects:

Definition 7.2: Report

A report over a repository instance RI is a tuple *report = (name,* BDI_{read} *,* bdi_{report}*)*, such that:

- name ∈ ACTIVITIES is the name of the report
- $BDI_{read} \subseteq RI$ is a set of business document instances to be read and
- $bdi_{report} \in RI$ is a business document instance representing the *report*-data.

The definition above does not state how bdi_{report} is acquired using the elements in BDI_{read}.

Based on the definition above the specification of different report patterns is as follows:

- **Name:** Report Control Pattern
  Risk situation: The necessary analysis of business transactions accessible through periodic reports fails because the employees assigned to run and analyze the reports are remissed in their duties. Thus undesired business situations (such as continuously reduced monthly turnover) remain undiscovered.
- **Control Objective(s)**: Financial Reporting, Operations
- **Solution**: Check whether the necessary reports in the system are run by the assigned employees and compared to each other on a periodic basis.
- **Related to**:
- **Example**: Check whether the necessary report, which generates a list of open purchase requests that have not been converted into purchase orders, is run on a periodic-basis.

There are three different variants of this pattern:

- Report Execution Check: Controls of this pattern assure that a report is run on a periodic basis.
- Intra-Report Comparison Check Pattern: This pattern checks whether the variance of certain attributes between different instances of the same report type generated in a certain frequency (daily, weekly, monthly, yearly) exceed a predefined value.
- Inter-Report Comparison Check Pattern: This pattern checks whether the variance of certain attributes between different instances of different report types generated in the same period (day, week, month, year) exceed a predefined value.

In the following section, the pattern-specific parameters for each type of report pattern are introduced:

$PSP_{Report-Execution-Check}$ =*(r, n, m, f)* with:

- *r* is a report according to the Definition 7.2,
- *n, m ∈ N and f ∈ FREQUENCIES* specify the number of generations (invocation) of r in a certain period (for example *n = 1, m = 4, f = week* specify that the report has to be generated 1 time in a 4 week period).

$PSP_{Intra-Report-Comparison}$ =*(r, CATTS, V, assignVariance, assignVarianceType, f)* with:

- *r* is a report
- *CATTS* $\subseteq A$ is sub-set of attributes of bdi_r (the business document representing the data of report r - see Definition 7.2)
- $V \subseteq \Re$ is a set of real numbers with *#V = # CATTS*, where *# A* means the cardinality of set *A*
- *assignVariance* is a total function *CATTS→ V*, which specifies the allowed variance of each attribute in *CATTS* on a periodic (*f ∈ FREQUENCIES*)– basis
- *assignVarianceType* total function *V→ PCD*, which specifies the type of each variance in *V* (*PCD* is set of primitive data types, see section *4.2.2.1).*

$PSP_{Inter-Report-Comparison}$ = *(r, r', $CATTS_r$, $CATTS_{r'}$, V, assignAttribute, assignVariance, assignVarianceType, f)* with:

- *r, r'* are each the reports to be compared *(r ≠ r')*
- $CATTS_r \subseteq A$ and $CATTS_{r'} \subseteq A$ are each a sub-set of attributes of the report-specific business-documents bdi_r and $bdi_{r'}$ with $\#CATTS_r = \# CATTS_{r'}$
- $V \subseteq \Re$ is a set of real numbers ($\#V = \# CATTS_r$)
- *assignAttribute* is a total function $CATTS_r \rightarrow CATTS_{r'}$ specifying which attribute of each of the given report types have to be compared,
- *assignVariance* is a total bijection between the set representing the results of *assignAttribute*-relation and *V*. It specifies the allowed variance between the respective attributes of *r* and *r'* on a periodic (*f ∈ FREQUENCIES*)– basis
- *assignVarianceType* is a total bijection *V→ PCD*, which specifies the type of each variance in *V* (*PCD* is set of primitive data types, see section *4.2.2.1).*

Figure 47 represents the role of *assignAttribute, assignVariance* and *assignVarianceType* relations in PSP-Definition of *Inter-Report-Comparison* control pattern by an example. In the figure two reports *R1* and *R2* are used, where *R1* has the attributes *a, b* and *c* and *R2* has the attributes *d, e* and *f.*

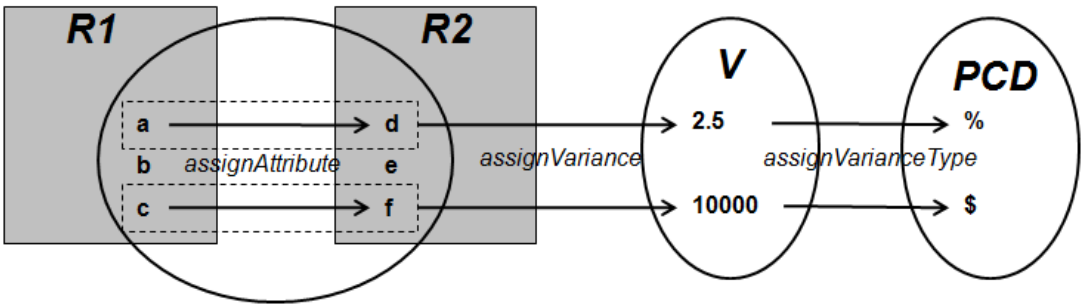

Figure 47 Illsutration of PSPs for Inter-Report-Comparison

SoD Patterns

- **Name:** Separation of Duties Pattern (Also known as Segregation of Duties – SoD)
  Risk situation: A deliberate fraud might occur when the completion of an activity is the duty of a single person.
  Control Objective(s): Financial Reporting
- **Solution**: Divide a business level activity into two or more sub-activities, which together fulfill the original activity. Define for each sub-activity a separate employee having the same role (User-based- SoD) or a separate role (role-based-SoD) to achieve the sub-activity. Consider that it is possible that an employee can have more than one role. Role-based- SoD becomes ineffective if an employee has the roles required to achieve each sub-activity in SoD. An unreasonably high number of activities under the control of SoD can reduce the operational effectiveness of a business process. In case of user-based-SoD make sure that the substitute-concept for the user with the duty for each sub-activity is implemented and assure its effectiveness.
- **Related to**: SSE, Inter-Activity
- **Example**: See control CB2 in scenario (section 2.2.3).

Below the pattern-specific parameters for use- and role-based SoD are introduced:

$PSP_{User-Based-SoD}$ = *(TRS, rle), with*

- *TRS* is a set of transitions in the current business process
- *rle* is a role. Each user executing a $trs \in TRS$ must have the role *rle.*

$PSP_{Role-Based-SoD}$ = *(TRS, RLS, assignRole),* with

- *TRS* is a set of transitions in the current business process
- $RLS \subseteq ROLES$ is a set of roles.
- *assignRole* is a total bijection between *TRS* and *RLS*.

Authorization Patterns

- **Name:** Authorization Pattern
  Risk situation: A deliberate fraud might happen when an employee intentionally can misuse resources or has unlimited access to resources.
- **Control Objective(s)**: Financial Reporting
- **Solution**: There are several (complementary) solutions: i) check the restricted access of users to roles (User-based- Authorization), ii) check the role's grant to execute an activity (Role-based) or iii) limit users' grants to execute certain activities, if they did not execute that activity for a certain period (Time Limited Authorization pattern)
- **Related to**: SSE, SoD
- **Example**: One-Time-Supplier-Creation-Control: To accelerate the reactivity of a company to changes in the supplier market, it is possible to give the right to certain employees in the purchasing department to create supplier-entries with incomplete information data of the supplier. This should enable the company to enter into short-term business-relationships with a business partner (Suppliers who will be contracted possibly only once). This type of supplier is called a *One-Time-Supplier* and certain users are allowed to *create* such supplier entries. The control says that it should be ensured that this right (Executing One-Time-Supplier-Creation-Activity) is ***revoked*** from those employees

in the purchasing department who did not create such a supplier-type in the last 3 months. Such a control is of type “Time Limited”.

Below the pattern-specific parameters for different types of authorization patterns are introduced:

$PSP_{\text{User-Based-Authorization}}$ = *(usr, rle)*, with:

- *usr* $\in$ *USERS* is a User, which will be checked if it has the role *rle*
- role *rle* $\in$ *ROLES.*

$PSP_{\text{Role-Based-Authorization}}$ = *(rle, trs)*, with:

- *rle* $\in$ *ROLES* is a Role, which will be checked if it is allowed to execute the Transition
- *trs: Transition.*

$PSP_{\text{Time-Limited-Authorization}}$ = *(usr, trs, n, m, f)*, with:

- *usr* $\in$ *USERS*, is the employee, who should execute the
- *trs: Transition*
- *n,m* $\in$ *N* and *f* $\in$ *FREQUENCIES* specify together the period of time, during which the user is required to enact the transition (for example n = 3, m = 1, f = year specify that transition should have been enacted by the user 3 times in the last 1 year).

Escalation Patterns

- **Name:** Escalation Pattern
  Risk situation: The control environment set up in an enterprise does not provide an atmosphere in which employees can behave compliantly. Another risk is that the policies are not taken seriously by the employees, or they don’t pay attention to the control violations caused by them.
- **Control Objective(s)**: Financial Reporting. Operation, Compliance
- **Solution**: Notify certain instances in the company (employees), if a control is violated more than a certain number of times during a certain period.
- **Related to**: All other control patterns
- **Example**: If purchase orders that require SSE are approved by only one employee frequently (say weekly), then the purchasing manager has to be informed.

Pattern-specific parameters for escalation pattern are:

$PSP_{\text{Escalation}}$ = *(ctl, n, m, f)*, with:

- *ctl* $\in$ *CTLS* is the control, which is violated
- *n, m* $\in$ *N,* and *f* $\in$ *FREQUENCIES* specify together the number of control violations in a period (for example *n = 3, m = 1, f = week* specify that the control ctl is not allowd to be violated more than 3 times weekly, otherwise this will be escalated).

7.5 Control Pattern Instantiation

Control pattern instantiation is a mechanism for creating a control (according to Definition 6.11) based on a specific control pattern. The intent is to create a specific control enforcing the desired usage of the pattern in a specific business process. Instantiation consists of binding all the pattern-specific parameters of a control pattern to a control condition (see Definition 6.9).

In order to be able to evaluate control conditions capturing a business situation that violates the control, it is required to obtain the set of instances of any controlled entity (CE) (Transition, Role, Business Document, Control, User) in a business process instance according to a given selection criteria. By checking the attributes of the set of retrieved CEs, it can be decided whether a control is violated or not. For this reason we introduce the concept of *Query* of a CE. The query of a CE (*CEQuery*) determines the set of all instances of that CE according to a given filter.

The following definition specifies different types of queries, where the sets used in the Definition are: *TRANSITIONS* is a set of transitions (Definition 4.14), *TRANSITIONINSTANCES* is a set of transition instances (Definition 4.15), *CTLS* is a set of controls (Definition 6.11), *ROLES* is a set of roles and *USERS* is a set of users.

Definition 7.3: A *CEQuery* on a business process *repository instance BPRI* of a *repository BPR* is any of the following functions:

- *queryRoles* $\subseteq (BPRI \times TRANSITIIONS) \rightarrow 2^{ROLES}$ is a partial function which returns a sub-set of roles in *ROLES*, who have executed a transition $t \in TRANSITIONS$ in the given $BPI \in BPRI$
- *queryBDIS* $\subseteq (BPRI \times R \times \{condition\}) \rightarrow 2^{RI}$ is a partial function, which returns a sub-set of business document instances in RI. For the result set 2^{RI} it holds that the type of the business document instance is the same as the business document type given by the business document type in *R* (in the domain of the relation) and all the instances satisfy the condition given.
- *queryTRANSITIONS* $\subseteq BPRI \times TRANSITONS \rightarrow 2^{TRANSITIONINSTANCES}$ is a partial function which returns a set of transitions instances of the specified transition type existing in the corresponding model of the given business process instance in *BPRI.* If no transition type is specified, all transition instances will be retrieved.
- *queryCONTROLS* $\subseteq BPRI \times CTLS \rightarrow 2^{CTLS}$ is a partial function that returns all the controls applied on a business process instance, which are of the type specified in the query. If no control type is specified, all controls of all types will be retrieved.
- *queryUSERS* $\subseteq BPRI \times TRANSITIONS \rightarrow 2^{USERS}$ retrieves the set of *users,* who have executed the specified transition type in the given business process instance in *BPRI.*

Although the different types of queries introduced in Definition 7.3 are invoked during the execution of business processes (in order to evaluate the control condition), during instantiation of a control pattern (which is related to design time of controls) a set of such queries must be created and stored together with a control condition. Thus a control instantiation of a control pattern is defined as follows:

Definition 7.4: Control Pattern Instantiation
Instantiation of a control pattern *cp* on a business process is a procedure ϕ_{cp}, which receives as inputs PSP_{cp} *and* a business process definition *bpd (*according to Definition 4.2*) and* generates a *control condition* (according to Definition 6.9) and a set of *CEQuery* (according to Definition 7.3).

As an example, the control pattern instantiation procedure for control pattern Intra-Role-SSE is described in detail below:

Control Pattern Instantiation $\phi_{Intra-Role-SSE}$ (According to Definition 7.4)

Input: $PSP_{Intra-Role-SSE}$ *= (trs, n , rle), business process definition bpd*
Output: *control condition cc; set CEQ containing elements of type CEQuery*

```
1  ∀bpi of type bpd {
2          T : queryTRANSITIONS (bpi, trs);
3          R: queryROLES(bpi, trs);
4          U: queryUSERS(bpi, trs);

           cc =
5          SIZE_EQUALS (R, 1) and
6          CONTAINS (R, rle) and
7          SIZE_EQUALS(T, n) and
8          SIZE_EQUALS (U, n);
9          CEQ =  { queryTRANSITIONS (bpi, trs), queryROLES (bpi, trs), queryUSERS
                  (bpi, trs)}
   }
```

The set *CEQ* containing elements of type *CEQuery* is given in line 9. The variables *T, R and U (Line 2-4)* are variables that will contain the results of the queries (which will be available at execution time). These variables are then used in statements of control condition (cc).

The control condition (cc), as another output of the procedure, contains the following statements: The control statement in line 5 checks that there is only one role, which has executed the transition *trs*. The reason is that the control pattern is of type Intra-Role. The control statement in line 6 verifies whether the required role *rle* in fact has exclusively executed the transition. The statement in line 7 checks that the *trs* is executed *n* times and the statement in line 8 checks that number of users, who have executed *trs* is *n*.

Note that the instantiation procedure only binds the pattern–specific-parameters to control conditions. The extension of a control condition is possible by manual addition of more control statements to a control condition, in order to refine the control. This is out of scope for the instantiation procedure of a control pattern. For instance by adding a statement like *GREATER_EQUALS(po, amount, 5000)* to an already existing control condition of a control definition adhering to the SoD-Pattern (possibly generated by the instantiation procedure), it is possible to make the control mandatory only for purchase order business documents instances (*po*) with a total amount higher than *$5000* .

7.6 Conclusion

Using the control patterns, the compliance expert can considerably reduce the compliance effort and improve the quality of the compliance design for a business process. This is possible due to the fact that instead of having to define a control from scratch, a compliance expert may browse or query the control pattern repository and select a control pattern of interest that mitigates an identified risk. The compliance expert then designs the control for the currently selected business process by configuring the control pattern into a real control (specifying values for control parameters). Furthermore, providing a control pattern repository and an approach for designing them into business processes can provide added value to the software of standard software providers. A major value-add is that their customers from different sectors can build their compliance on top of such a repository. This raises the level of reusability and usability of software and provides the compliance experts at the customer site with reusable knowledge provided by the patterns. In this chapter we presented such a pattern-based approach for designing the necessary controls for business process compliance.

Furthermore, a set of empirically determined control patterns were presented and described. For the description of these patterns a set of attributes were used. We introduced the notion of pattern specific parameters for control patterns. Pattern specific parameters are parameters related to a specific control pattern. They serve as the basis for generating a concrete control that adheres to a certain control pattern and also as input for generating control-pattern-specific user-interfaces. The process of generating such a control based on a control pattern we called control pattern instantiation. Control pattern instantiation was defined as a procedure that generates a control condition and a set of queries for each control. These queries will be used to assure the compliant behavior of business process instances by providing the basis for evaluating the control condition during the execution of business processes.

8 Compliance Validation of Business Process Executions

In chapter 4, we presented a detailed model of the business process model as a BPD. We showed in chapter 6 how the controls can be modeled in a business process model (Definition 6.11). The relationships existing between the models of a control and a business process were also described there.

The focus of this chapter is to engage the proposed models through an approach that ensures the compliant behavior of business process instances according to the defined controls. The system realizing this approach is called ICR (Internal Controls Repository), which is divided into two sub-components: ICR-Design and ICR-Execution. As we will see in this chapter, assuring the compliant behavior of business process instances requires a set of preparations on business process models. This set of preparations is and inherent part of our approach. A detailed description of the implementation of the approach is provided.

First, we make some introductory remarks on ICR. We continue in section 8.2 by discussing the foundations of the technologies used in the implementations. The discussion on foundations will be completed by describing the selected tool environment. Our approach is realized based on this tool environment. In section 8.3 the approach itself, which is used for the implementation of ICR, is described. In 8.4 we describe the implementation by giving

i) the requirements for the technical realization of the approach and

ii) the reasons behind our selection of the specific implementation of ICR-Design, motivated by the technical requirements listed before and

iii) the detailed implementation of the integration of control and business processes instances (ICR-Execution).

Related work will be discussed in section 8.5, after which this chapter will be summarized.

8.1 Introduction

Although the proposed modeling approach allows for a flexible and usable definition of controls on business processes (supported by the pattern-based design of controls presented in chapter 7), during the execution of business processes the separated model of business processes and the controls have to work in a tightly integrated manner in order to support the prevention of control violations produced by business process instances. Recall that the design and controls of business processes are not the only processes subject to compliance certification; the execution of business processes in daily operations (business process instances) must also be compliant. This implies that, while one must be able to separately design the controls and business processes, it is also necessary that the controls and business process instances be re-integrated during the execution of business processes. One of the main contributions of this thesis is that we render it possible to separate the design of controls and of business process models, done respectively by compliance experts and business process experts. Based on this, the approach presented in this thesis allows for the automatic detection and prevention of control violations of business processes during execution time without necessitating any human interactions.

In order to support the separation of the business and control objectives, and the automatic prevention of control violations produced by business process instances, this thesis introduces the possibility of another layer above the business process model and executions. The system responsible for realizing this layer is called "**Internal Controls Repository" (ICR)**. The ICR is divided into two sub-components:

- **ICR-Design**: According to the assessed risks, a set of controls is defined in ICR-Design.
- **ICR-Execution**: By executing a business process, ICR-Execution will be continually updated with information needed for the evaluation of defined controls in order to ensure that compliance tests will pass.

In chapter 6 and 7 we discussed how the controls can conceptually be designed in ICR (ICR-Design). In this chapter we are concerned with the application of controls existing in ICR-Design to support the compliance of business process executions.

The approach necessary for the realization of ICR-Execution is tightly linked to the way a control is technically realized, i.e. its concrete representation in ICR-Design. Recall that in chapter 6, we only presented the conceptual model of a control and did not make any statements as to its concrete realization, i.e. the formalism and the implementation selected for its technical persistence in a system for business process compliance.

We have decided to implement the system of ICR based on a rule-based approach. The reasons supporting this decision are as follows:

- Considering the core structure of a control according to Definition 6.11 as consisting of a triggering event, control condition, and recovery actions, a control can be read in the following way: if a triggering event occurs and at the same time a condition is fulfilled, then invoke the recovery actions. This is basically a rule.
- Rule based languages can be expressive enough to capture the control model.
- The declarative nature of rule-based languages yields an acceptable compromise between expressiveness and simplicity.
- The high abstraction level of rule-based languages allows formulating statements close to natural language formulations suitable for people with little or no technical skills. This has led to developments in the area of domain specific languages (DSL) [Mernik et al., 2005] built on top of rule languages.

Based on the rule-based presentation of controls this chapter will present an approach for integrating the controls with business process instances achieved in ICR-Execution.

8.2 Foundations

8.2.1 Introduction to Business Rules

Business rules provide a way to capture organizational knowledge in a structured and formalized manner [Herbst, 2000]. Further [Herbst, 2000] states that business rules can be defined as "statements about how the business is done, i.e., about guidelines and restrictions with respect to states and processes in an organization". The Business Rules Group [BRG2000] defines a business rule as "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".

[Taveter et al., 2001] and [Wagner, 2002] give a typology of formalized business rules based on [Bubenko et al., 1998], which distinguishes between the following types of business rules: Reaction rules, derivation rules, and integrity constraints. Reaction rules are further sub-divided into Event-Condition-Action-rules (ECAs) and production rules:

ECA-Rules are concerned with the invocation of actions in response to events. They state the conditions under which actions must be taken [Wagner, 2002].

In rule based systems, production rules are of the form "*IF c THEN a*", where *c* is a condition and *a* is any kind of action, including external procedures/methods. Production rules are very

similar to ECA rules. [Wagner, 2002] states that ECAs can even be considered as a special case within general concept of production rules. From a conceptual point of view, we see an ECA-rule as a production rule, where the notion of *event* in ECA is explicitly designed in the left hand side (LHS) of a production rule. However the underlying algorithms and infrastructure for processing both types of rules can be identical. In this thesis, we differentiate between production rules and ECA-rules through the way a rule is written, thus an ECA-rule can be transformed into a production rule. Because production rules serve as the *implementation* of the compliance approach during the execution time of business processes presented in this thesis, we introduce them in detail in sub-section 8.2.2.

Derivation rules allow for the derivation of knowledge from other knowledge by an inference or a mathematical calculation [Wagner, 2002]. Each rule expresses the knowledge that if one set of statements happens to be true, some other set of statements must also be, or will become, true. Derivation rules are the basis for the programming paradigm "Logic programming" [Lloyd, 1984], which relies on a subset of first order logic, referred to as Horn clause Logic.

According to [Wagner, 2002], an integrity constraint is an assertion that must be satisfied in all evolving state and state transition histories of a discrete dynamic system.

To the above-introduced categorization, we add that a clear semantic separation of different categories of rules is not a straightforward task. Many rules can be assigned to different categories and at the same time a certain category can be considered as a special type within another category: categories must not necessarily be mutually exclusive. For example, an integrity constraint can be interpreted as a reaction rule, if in the case of its constraint violation a reaction should follow.

8.2.2 Production Rules

8.2.2.1 The Structure of a Production Rule

The following concepts are generally central to a system employing production rules [BRG2000]:

- Terms
- Facts
- Rules

Terms are artifacts that are important to a domain. They build the language and the terminology for the presentation of the domain. In our model of business process compliance the conceptual model behind the terms are the set of controlled entities, namely the business documents, transitions, users and their roles, and the controls.

A *Fact* is an instantiation of terms or the instantiation of relationships between two or more terms. Examples for facts are: "*A PO with a total amount of 5000$ for material type 5 has been approved on 21.12.2005*" or "*User Smith has invoked the operation One-Time-Vendor-Creation*". In the area of business process compliance, the concept of facts is reflected in the model of business document instances, transition instances, business process instances and controls instances.

The core component of a production rule system is its rule engine (sometimes called inference engine). This engine is able to process rules and facts. The engine matches the facts against the existing rules to infer conclusions resulting in the execution of the actions defined in the rules. The matching process is called *Pattern Matching*. A production rule is itself a two-part structure of the following form:

IF <conditions> THEN <actions>

The *IF*- part in a production rule is called the *Left-Hand-Side* (LHS) and the *THEN*- part is called the *Right-Hand-Side* (RHS). In production rules the foundation used for expressing the LHS is first order logic, while the actions in RHS can be reduced to a set of invocations, which create new facts or modify or delete existing facts.

8.2.2.2 Components of a System for Production Rules

The basic functions of a production rule system are:

1. Creation of rules
2. Management of facts and
3. Decision of which rules to fire

In the context of production rule systems, the first functionality results in a technical component called *Production Memory* or *Rule Base*. We will use the term *rule base*. A rule base basically contains a set of production rules. The technical component responsible for the second functionality is called *Working Memory*. A *working memory* contains a set of facts and is related to the execution time of business processes in the case of business process compliance. Facts are managed in a working memory; they can be created (assertion), modified, or deleted (retraction). As previously mentioned, the third functionality is achieved by the *Rule Engine*. The rule engine has two main sub-components, the *Pattern Matcher* and the *Agenda*. While the pattern matcher determines the truth of conditions of rules that must be fired, the agenda is responsible for the execution order of rules that are fired. The agenda further verifies whether rules are in conflict with each other, in cases where more than one rule becomes true. In the latter case the agenda of a production rule system uses a *conflict resolution strategy* to resolve conflicting rules that have become true in parallel. All rules are evaluated against all facts in the working memory. For each eligible combination of facts (that is, the condition part of the rule is true) a rule instance (the action part) is created on the agenda. The agenda is basically a last-in-first-out-stack (LIFO) that keeps track of rules which have to be fired. After the agenda has been updated, rules are fired (their action part is executed). The execution of a rule may alter the working memory, which may lead to a new modification of the agenda (creation of new rule instances on the agenda). After these modifications, ineligible combinations of rules and facts are pruned from the agenda. The execution of rules continues until the working memory is stable and the agenda is empty. Figure 48 outlines the general architecture of a production rule system.

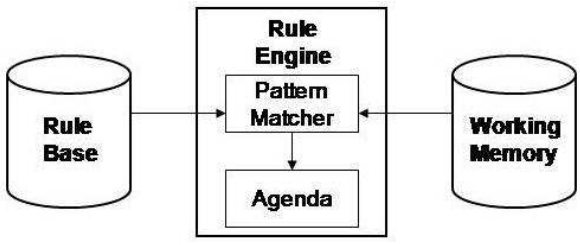

Figure 48 Rule Base, Rule Engine and Working Memory of Production Rule Systems

When facts are asserted to the working memory, the production rule engine creates "working memory elements" for each fact. In the working memory, a fact is stored in the form of a tuple and may contain an arbitrary number of data items. Thus it is possible to store business document instances as facts in the working memory and their assigned attributes as elements in tuples. The possibility of storing fact combinations allows for the creation of a working memory based on the "objects" as instances of classes from an object-oriented point of view. Many production rule engines (Drools [Drools] and Ilog Rules [ILOG] to name a few) use this

possibility to create rules according to the class model provided by an object-oriented application. The working memory then presents the current instantiation of classes (objects) produced by OO-runtime components (JVM in case of java). Thus it is possible that the class model of the current domain of interest provides the terms (the language) on which the rules are built. A very important aspect of building production rules on top of an already existing "domain model" is that it requires no further modelling efforts, since the terms relevant for building the rules are already designed and provided by the application (business process).

8.2.2.3 RETE Algorithm and Forward Chaining

The effort required for processing a larger rule set quickly increases if every combination of facts in the working memory is evaluated against every condition of the rules in the rule base. Most production rule engines are based on the RETE algorithm [Forgy, 1982] in order to increase the performance of rule processing. The RETE algorithm is responsible for implementing the PatternMatcher-component as described in the previous section. It builds a network of nodes in which each node represents a single condition of a rule. The RETE network allows for the identification of valid fact combinations that fire rules more efficiently. If rules contain identical conditions, the condition-node is created only once. Facts are then "filtered" by the RETE network. Only those facts and fact combinations that can cause a rule to fire are passed through the network. Since "valid" combinations of facts are identified more quickly, the identification of valid rules (those which have to be fired) is improved.

The RETE network has a single entry point and one exit point for each rule. The first part of the tree differentiates between types of objects that are processed in the network (*discrimination tree*). *Alpha Nodes* represent conditions of a rule. *Join Nodes* represent joins between conditions of one or more rules. The last node is the *Rule Node*, which controls the agenda. Whenever a valid combination of facts (a fact combination that matches the conditions) reaches this node, the agenda will be modified according to the action specified in the current rule.

Facts are propagated through the network via tokens. A positive token indicates that a new fact has been asserted in the working memory. A negative token indicates that a fact has been retracted from the working memory. If such a token reaches a rule node, the agenda will be updated accordingly. That is, putting the rule instance on the agenda in the case of a positive token and removal of the rule instance in the case of a negative token.

The RETE algorithm uses the principle of *Forward Chaining*. In contrast to *Backward Chaining* (see section 5.1.4), forward chaining has a reactionary nature and is "fact-driven". That is, it reacts on changes affected by facts asserted in the working memory. As facts are asserted in the working memory, the conditions of some rules, can become true, possibly concurrently, and are put on the agenda to be fired. In this way a conclusion is reached. Backward chaining is "goal-driven".

8.2.2.4 Truth Maintenance and Shadow Facts in Production Rule Systems

Another significant feature of compliance validation of business process executions supported by the rule engines of production rules is the concept of *Truth Maintenance* [Doyle, 1979]. This concept is used in the implementation of the approach (that will be presented in section 8.4) in order to achieve an accurate and efficient synchronization of the ICR-Execution when changes in a business process instance take place outside of the ICR-Execution.

Truth maintenance is related to the statefulness of the facts generated and managed in the working memory and assures the consistency of logical relationships generated by actions in a rule. Truth maintenance in production rules is made possible because the modification and

retraction of facts already existing in the working memory is possible. Rule engines that are able to support truth maintenance are called non-monotonic reasoners.

Closely related to truth maintenance is the concept of *Shadow Facts*. A shadow fact is basically a shallow copy of the productive data (for instance business document instances) generated outside of the production rule system, which has been asserted to the working memory. They represent a cashed copy of the asserted object as a fact in the working memory. The rule engine Jess [JESS] was one of the first rule systems which implemented the concept of shadow facts.

The basic relation behind truth maintenance and shadow facts is that a production rule system should guarantee the accuracy of the conclusions derived by the rule engine. A productive system (for instance a business process execution infrastructure or ERP system) may alter the data during the progression of a business process instance. In such a situation, the rule engine must somehow be informed of the changes which have occurred outside the rule system in order to derive conclusions that accurately reflect the situation in the outside world (Truthfulness). This can be achieved in two ways: i) by locking all the facts (and their according original productive data) during the inference process or ii) by maintaining a cache copy of the productive data and delegating all modifications through the rule engine. Thus the working memory is automatically synchronized to changes occuring outside of the rule system.

Figure 49 exemplifies the concept of shadow facts in the case of a business document instance bdi. Let's assume that during T_1 an initial fact version bdi_1 exists in a bp instance maintained by the infrastructure responsible for executing the business process definitions (BP Execution Engine). During T_2 the business document instance bdi_1 is asserted to the working memory. During T_3, the original business document instance bdi_1 is modified by the application and it is now an updated version of the original business document instance bdi_1, which we call bdi_2. The concept of shadow facts assures that during T_4, the bdi_1 fact is automatically updated in the working memory, so that the working memory now contains a fact bdi_2 (bdi_1 no longer exists in the working memory). Let's assume that during T_5 the value of the attributes of bdi_2 causes the pattern matcher to update the agenda by firing a production rule (putting it on the agenda). Let's further assume that the RHS-part of this rule causes the modification of bdi_2, so that during T_5 the working memory contains a new version of the bdi-fact, called bdi_3, as an updated version of bdi_2. The concept of shadow facts automatically assures that the bdi_2 instance in the bp instance maintained by BP Execution Engine is updated during T_6, so that the bdi of the bp instance has the value bdi_3 and not bdi_2 (as it was during T_3), and so on and so forth.

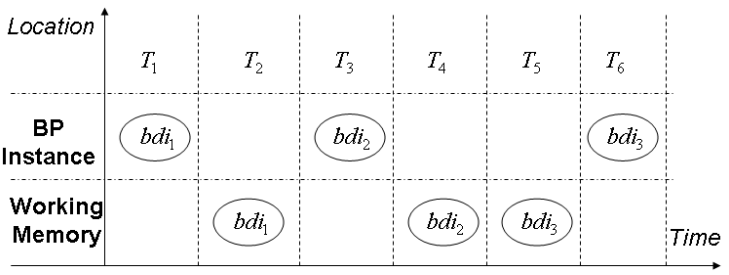

Figure 49 A business document instance in a business process instance and its shadow in a working memory

8.2.3 Tool Environment

From the point of view of implementation, there are two open issues which remain to be discussed:

1) How to design and execute the business processes and

2) How to implement the ICR.

We have chosen to implement the business processes in a java-environment with JBoss jBPM [jBPM] including a graphical Process Designer Tool and a Process Execution Engine. The processes in the jBPM designer can either be designed as BPEL or as jPDL [jBPM]. The latter one is a language developed by jBPM itself. jPDL leans on XPDL [XPDL]. We decided to design business processes with jPDL because it allows for concepts such as task management and identity management, which allowed us to completely simulate typical ERP scenarios necessary to our experimental environment. BPEL did however have limited usage in our approach: BPEL does not account for humans in a process, so it doesn't provide typical real life business scenarios [BPEL4People]. It was for this reason that we chose to first design the processes as jPDL in our prototype and then to use jBPM as its execution platform.

As for ICR implementation, we decided to implement the ICR based on production rules. A detailed technical discussion about the reasons for this decision is provided in the implementation part of this section. As the implementation platform for production rules we selected the Drools Rules Engine [Drools]. We selected Drools rules as a representative of several rule engines, because it is open source and is popular. Drools serves as a provider to *implement* the underlying control model and the approach for compliance validation of business process execution.

In the following two sub-sections, the basic technical concepts behind jBPM and Drools Rules, those which are necessary in order to understand the implementation part of the approach, are explained. For a detailed introduction to jBPM and Drools rules, please refer to the documentation of jBPM [jBPM] and Drools [Drools].

8.2.3.1 Business Processes with jBPM

jBPM is a java-based open source solution of a workflow engine.. A process definition in jBPM describes a state machine that is executed in the engine. In the following, a brief overview of the basic concepts behind jBPM is provided.

8.2.3.1.1 Business Process Design

Each business process is designed in the *Java Business Process Definition Language* (jPDL). The result of designing a business process in jPDL is a directed graph of nodes with edges between them. The following node-types are available:

- A *Task* node represents one or more tasks that are to be performed by humans. When the execution path arrives at a node of this type, a task instance in the task list of a specified business process participant will be created. After that, the node will fall into a wait state. When the users have performed their task, the completion of the task will trigger the further processing of the execution.
- A *State* node executes no business logic. This state can be useful when waiting for a signal from an external system (asynchronous business processes).
- A *Decision* node handles decisions in the flow, i.e. which leaving edge of the current node in the process execution should be taken.
- A *Fork* node can be used to split one path of execution into multiple, concurrent paths of execution.

- A *Join* node is the opposite of the *Fork* node and joins the multiple concurrent paths into one single path.

In the case where a node implementation has to invoke a certain functionality automatically (e.g. a web service), an *Action Handler* (see the next sub-section) must be implemented on that node. jBPM exposes its process design programming API, thus custom node types can be implemented as well. The design entities in jBPM which support the concepts of *users* and their *roles* in business process definitions are *swimlanes* and *actors* in jPDL.

8.2.3.1.2 Business Process Implementation

With a jPDL-process definition only the flow of execution is specified. The access to the underlying business logic encapsulated in different business systems (which provide business documents and their instances) is implemented by the notion of *Actions* that can be attached to the nodes.

When a node is processed during the execution of a business process, its Actions are invoked. An Action is implemented by implementing the interface *ActionHandler*. An *ActionHandler* satisfies *the Command-Design-pattern* [Gamma et al., 1995] and works in the following way*:* When the process execution engine encounters a node in the process definition that has an action associated with it, all *ActionHandlers* related to the node are invoked. *ActionHandlers* are instances of java code that can interact with external systems when executed. An *ActionHandler*-interface contains the method *execute*, which receives an *ExecutionContext* as an input-parameter. *ExecutionContext* is basically a Map-Data structure (set of key-value-pairs), which can contain any serializable java object. The implementation of the *execute*-Method of an *ActionHandler* can read and write received data into the *ExecutionContext*. In such a way data can be manipulated and populated across nodes in a business process instance. Our implementation of the business process context (see Definition 4.16) encapsulates the ExecutionContext of jBPM. The following listing is a simple example of an *ActionHandler*-implementation and its corresponding integration in a jPDL-defintion. The explanations of the code are given in the code as comments:

Listing 9.1

```
public class ApprovePurchaseOrderActionHandler implements ActionHandler
{
// execute is the only method of an ActionHandler-Interface. Every class of this interface must
// implement this method
 public void execute(ExecutionContext executionContext)
 {
      // read from context which purchase order has to be approved, where APPROVE-PO is
      // the key under which the purchase order can be looked up in the context
     Integer poId = (Integer)
            ctx.getContextInstance().getVariable("APPROVE-PO");

     // connect to the PurchaseOrder-Business-Layer (SRM) and approve the order:
     Connection connection = SRMFactory.getConnection();

     // invoke the approve-activity
     connection.approve(poId);

     // close the connection to SRM
```

```
        connection.close();
    }
}
```

The *ActionHandler* above can be integrated in the following way in a process definition:

Listing 9.2

```
<process definition name = "purchasing">
        …
<!—The activity Approve-PO, to which the action handler
        ApprovePurchaseOrderActionHandler will be  assigned -->
<state name = "ApprovePO">
        <!—An instance of the ApprovePurchaseOrderActionHandler-class will be generated and
          equipped with the context of the business process instance, when from the Approve-PO-
          Node the transition to Send-Po-Node is taken -->
        <transition to = "SendPO">
                <action class = "ApprovePurchaseOrderActionHandler"/>
        </transition>
</state>
<state name = "SendPO">
...
</state>
</process definition>
```

The jBPM engine will fire different kinds of events during a process graph execution. Events in jBPM specify moments in the execution of the process. An event occurs when jBPM calculates the next state. Each node type has its specific event types. A node can fire a *node-enter* event or a *node-leave* event by default. An event in jBPM is the hook for actions and is associated with a list of actions. When the jBPM engine fires an event, the list of actions is executed. Using the *node-enter* event we implement the control condition *EXECUTING* (see Definition 6.9). In such a way, we are able to capture the moment *before* a transition in a business process instance is in fact invoked, i.e. the *effect* of an *activity* (see Definition 4.12) or state, e.g. changing results on a business document instance in case of a *state change command (scc)* (see Definition 4.13). State change commands are commands of a business document type (see Definition 4.6) that cause a business document to change its state value (see Definition 4.5).

The logic behind a *decision* node, i.e. the calculation of the leaving edge selection, is realized using the implementation of a *DecisionHandler*-Interface. The principle behind a *DecisionHandler* is very similar to the one of an *ActionHandler*. The difference is that a *DecisionHandler*-interface contains the method *decide* which receives an instance of *ExecutionContext* and as a result returns the name of the leaving edge that should be taken. It is up to the implementation of the *DecisionHandler* to determine the name of the edge.

8.2.3.2 Process Execution

The jBPM execution engine is responsible for executing business processes designed with jPDL. The execution model of jBPM is about processing the directed graph, created during design of a business process, and interpreting it as a state machine. jBPM implements the graph interpretation and execution according to the chain of the command design pattern [Gamma et al., 1995]. Very briefly explained, the chain of the command pattern means that each node in the

graph is responsible for propagating the process execution. A wait state (node) means that a node does not propagate the execution. When an execution arrives in a node, *signals* can be sent to the execution. Sending a signal to the execution is an instruction to continue the progress of the current business process instance. In addition to the above described synchronous execution model, jBPM also supports asynchronous process executions. In this case a business process instance can handle signals received from external systems. jBPM relies on the concept of asynchronous messaging (such as Java Messaging Service JMS) to implement this behavior. For a detailed description of the jBPM execution model and the underlying graph process algorithms, please refer to [jBPM].

8.2.3.3 Production Rules with Drools Rules

For the implementation of the ICR, we used Drools Rules 4.0. Drools Rules is a java-based Rule engine. Rules in Drools can be written in a rule notation called DRL (Drools Rule Language) or as XML. We decided to produce the rule in DRL-format since it is more user-friendly to read, write and understand than XML. We also use the rule-management API, which is provided to create and manipulate rules in DRL. In the following we give a brief overview of the concepts behind rule authoring in Drools rules. For a complete documentation, please refer to [Drools].

The rule base-component of production rules (see section 8.2.2.2) is covered in Drools through a set of rule-files. A rule-file in DRL is made up of the following elements:

- *import*
- *globals*
- *functions*
- *queries*
- *rules*

Import elements import a set of classes that can be used in rule authoring, i.e. the available terms. They have the same purpose in a Drools rule file as the import statements in a java-class. *Globals* are global variables used to make application objects available to the rules. *Globals* are not asserted to the working memory, thus any changes to a global variable caused by the application logic will not be available to a rule using a global variable. *Functions* in Drools are used to put executable java-code into the rule source file, which can be invoked in the RHS of a rule. *Functions* in Drools may have a return type (java class) and receive different parameters as input. In the following we further detail the *Rule*-element in a rule file and the concept of *queries* on the facts in the working memory.

8.2.3.3.1 Rules in Drools

A rule in Drools follows the following syntax, where * means that the preceding element may occur 0 to many times:

rule "<name>"
  *<attribute>**
when
  //LHS
  *<conditional element>**
then
  //RHS
  *<action>**
end

In the LHS a set of conditional elements can be specified. In Drools 4.0, the following conditional elements exist:

- *pattern*
- *and*
- *or*
- *not*
- *exists*
- *forall*
- *collect*
- *from*
- *accumulate*
- *eval*

With *Pattern*, conditional element patterns of facts existing in the working memory can be specified. For example the pattern

?po: PurchaseOrder (id = 4711, totalAmount > 10000)

returns true if in the working memory a fact with the *id*-parameter-value 4711 and a *totalAmount*-parameter higher than 10000 were asserted. Inside the Pattern conditional element (its parenthesis) a set of constraints can be specified. Constraints can be separated by the following symbols ',', '&&' (conjunction) or '||' (disjunction). A field of fact is an accessible method of the according java object. Thus the model objects (java classes) should follow the java bean pattern [JavaBean]. A java bean is a container of attributes, where each attribute is accessed via "*get<X>*" or "*is<X>*" methods (assuming X is the name of the attribute). This means that in the above example the java-implementation of the PurchaseOrder-class must have two methods *getId()* and *getTotalAmount()*. The value of the fact matched by a pattern can be bounded to a variable using the *?* and *:* notations. A variable can then be used in other conditional elements or in the RHS of a rule.

The '*and*' and '*or*" conditional elements are used to group other conditional elements together. *'not'* has the semantic of the negation of first order logic's existential quantifier and determines the non existence of a fact in the working memory. *'exists'* is first order logic's existential quantifier and checks for the existence of something in the working memory. The *forall* conditional element will be evaluated as true when all facts matching the first pattern also match all the remaining patterns. The *collect* conditional element allows rules to reason over a collection of objects collected from the given source or from the working memory. This is the cardinality quantifier in first order logic.

Besides the above constructs, Drools provides the following types of conditional elements:

The *From* conditional element allows the user to specify a source for patterns to reason over. This enables the engine to reason over data not in the working memory. This could be a sub-field bound to a variable or the results of a java method call. In this way, out of the box integration with other application components and frameworks is made possible. The conditional element *accumulate* collects facts in the working memory in the same manner as the *collect* conditional element, but with the additional functionality that it allows a rule to iterate over a collection of objects, executing custom actions for each of the elements, and at the end returning a result object.

Eval allows any java-class-method (that returns a primitive of type *Boolean*) to be executed. This can refer to variables that were bound in the LHS of the rule and functions in the rule file.

Usage of *eval* should be kept to a minimum because its use will lower the declarative level of the rules.

In the RHS of a rule the following actions can be undertaken:

- *insert* will place a new fact in the working memory
- *retract* removes a fact from the working memory
- *insertLogical* inserts a fact to the working memory, but the fact will automatically be retracted when there are no more facts to support the truth of the currently firing rule (Truth Maintenance)
- *update* will modify an existing fact in the working memory (one that has been bound to a variable on the LHS) with new parameters
- invoke a *function* existing in the rule file.

8.2.3.3.2 Querying facts in the working memory

The concept of *Query* (not related to the concept of queries on controlled entities – CEQuery – in section 7.5) in Drools allows the user to retrieve the facts that match the conditions stated in the query. The queries are designed in the rule file and are invoked from outside the working memory, i.e. any java-based application, which has a reference to the working memory. A Drools query has the same structure as the LHS of a rule. The following is an example of a query in Drools. It retrieves all facts of type "*Control*" from the working memory, where their attributes *id*, *bpId*, and *violationStatus* have the specified values (see inline comments inside the code):

Listing 9.3

```
// name of the query and its input parameters
Query "Query Control Violation" (int controlId, int bpd)
      // specifies the query for all controls with violationStatus having the value Violated etc.
      Control( controlId = id, bpdId = bpd, violationStatus = "Violated" )
end
```

A java application can invoke the query and retrieve its result in the following way (see inline comments inside the Listing):

Listing 9.4

```
// establish a connection to the working memory
WorkingMemory workingMemory = getReferenceToWorkingMemory();

// initialize the input parameters of the query (as specified in query specification in Listing 9.3)
Object[] queryArguments = initializeQueryArguments(controlId, bpd);

// invoke the query on the working memory.
// the results are contained in the variable "controls"
QueryResults controls =
      workingMemory.getQueryResults( " Query Control Violation ",  queryArguments);

// each element retrieved by the invoked query is iterated.
for ( Iterator it = controls.iterator; it.hasNext(); ) {
      QueryResult result = ( QueryResult ) it.next();
      Control violatedControl = ( control ) result.get( "control" );
      // Do something with each retrieved fact (violatedControl) from working memory:
```

```
        doSomething(violatedControl);
}
```

8.3 Overall Approach

Our approach for compliance validation of business process instances spans over three phases: the Compliance Design phase, the Business Process Definition Adaptation phase, and the Compliance Enforcement phase. These phases describe the interplay of

- a control that will be included into ICR-Design with the business process definition, on which the control is designed, and
- a business process instance with ICR-Execution.

8.3.1 Compliance Design Phase

This phase reflects the creation of a set of controls (according to Definition 6.11) by a compliance expert on business process definitions (according to Definition 4.2) contained in a business process repository (BPR) (according to Definition 4.4). The process definitions in their original form as delivered by a standard software provider have a generic nature (in terms of compliance), because compliance modeling is customer-specific. Before this phase, the process definition in the BPR may be non-compliant in that they do not contain and realize the controls required by the risk assessment of the enterprise. Figure 50 illustrates the sequence of steps in the compliance design phase indicating the manual as well as the automated (system-supported) steps. The steps in the figure can be explained as follows:

First, a compliance expert goes through the relevant business process definitions (identified by the relation *isRelevant in BPCD*, see Definition 4.1), as they may be delivered by a standard software provider. Then the compliance expert selects a controlled entity (CE, see Definition 4.3) contained in the process definition. Then he selects a certain control pattern from the control pattern repository. He then instantiates the selected control pattern. This is achieved by configuring the control conditions, scope and event of a control according to the enterprise's specific requirements (setting the configurable PSPs of the pattern) after the control conditions are automatically produced by the instantiation procedure (see Definition 7.4) of the selected pattern. Further, the required recovery actions are set in the control. When the compliance expert decides to activate the control (create control in ICR-Design), then i) the control is stored in the ICR Design as a model adhering to the control specified in Definition 6.11 and ii) the step for supporting the role-based recovery action modeling is invoked. Upon creation of a new control in ICR-Design, the according business process expert is notified. He checks the recovery action part of the control and, if necessary, he modifies/extends the recovery action model of the control (see section 6.2.3). After this phase, the control in the ICR-Design is again updated.

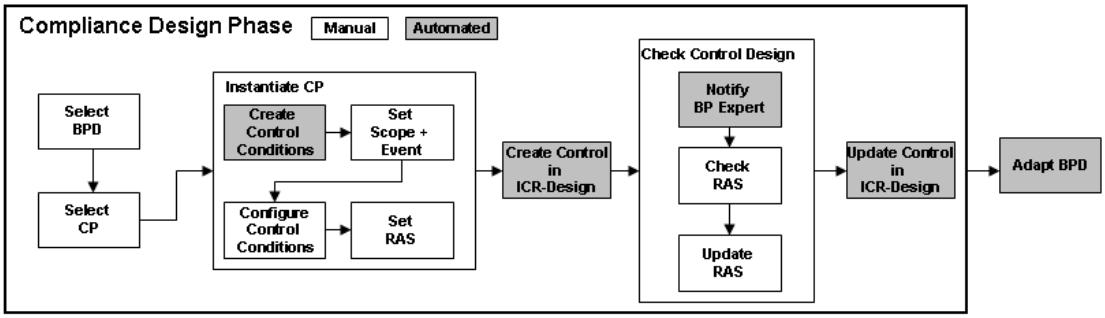

Figure 50 Sequence of steps in phase 1: Compliance Design Phase

In the next step, the currently selected CE in the process definition will be extended by the currently generated control in ICR-Design. This initiates the next phase of the approach, called the Business Process Model Adaptation Phase (Adapt BPD in Figure 50).

8.3.2 BPD Adaptation Phase

A BPD is originally in a control-free form. After phase 1 of the approach not only is a required control stored in ICR-Design, but the BPD currently selected is also extended by artifacts necessary to the required control. These artifacts have the task of implementing the recovery actions in a BPD. This means that that business process instance, which has caused a control violation, will behave as designed in the RAS-part of the control definition.

These additional artifacts in a BPD ensure that the BPD is not executed in a non-compliant way. Later on, during phase 3, the control in the ICR monitors that the controls are effective, i.e. that they operate as designed. This is required by law.

Definition 9.1: Business Process Model Adaptation
The process model adaptation of a business process definition is a tuple *adaptation = (bpd, bpd', ctl, TRS)*, in which:

- *bpd* $\in$ *BPR* is the process definition, which has to be adapted
- *bpd'* $\in$ *BPR* is the adapted process definition, with *bpd* $\neq$ *bpd'*
- *ctl* $\in$ *CTLS* is the control definition stored in ICE-Design
- *TRS* is a set of newly generated transitions in *bpd'*.

The definition above can be interpreted as specifying the business process definition adaptation as an operation. This operation receives a business process definition *bpd* and a control definition *ctl* as input and generates a new business process definition *bpd'* containing a set of transitions *TRS*, which did not exist before in *bpd*, as output.

A control model includes a specified set of recovery actions (see Definition 6.10), which are to be invoked if a control violation is detected. Different types of possible recovery actions were identified and discussed in section 6.2.3. In the case of a recovery action type *Instantiate (User, RecoveryProcess)*, no adaptation of the selected *bpd* as described above is required. The instantiation of the specified business process *RecoveryProcess* is dependent on the underlying implementation of the approach, i.e. its description is a technical issue. The realization of the recovery action on a *bpd* will be detailed in section 8.4, which deals with implementation.

8.3.2.1 BPD Adaption for a selected Recovery Action Model

Below we show the adaptation for a process definition *bpd* by a control *ctl*. The adaptation is illustrated in Figure 51.

Let

- *bpd* be the original control-free process definition consisting of (among others) two activities called *a* and *m* that are connected by a transition *trs*,
- *ctl* be the required control,
- the scope of *ctl* be *before* and the *business event* of the control event be *transition trs*, with *trs* having the following form:

  if c then invoke a,

  with $a \in ACTIVITIES$.
- the recovery action model selected be ***Retry*** *&* ***Notify*** *(mgr, msg) &* ***Instantiate****(bpexpert, rbp), with:*
  - transition *trs* being the transition that will be repeated by recovery action *Retry*, the business process *rbp* being the pre-designed recovery process definition for the process *bpd* in case of violation of *ctl.*
  - The user *bpexpert* will process the instance of *rbp.*

The set *TRS* of transitions in *bpd'* as specified in Definition 9.1 contains the following transitions:

trsNotOk *: if (VIOLATION(ctl, Violated))*
then invoke notify (emp, msg) & instantiate(bpexpert, rbp);

trs *: if (c and VIOLATION(ctl, NotViolated))*
then invoke a;

trsRetry*: if EQUALS (msg, to, emp)*
then invoke model_previous (trs);

trsCtl *: if (VIOLATION(ctl, Violated))*
then invoke scc = (ctl, updateControl, F), where
updateControl is the *name* of the *state change command* on a business document type *Control* and *F = VIOLATION (ctl, ViolationRecorded);*

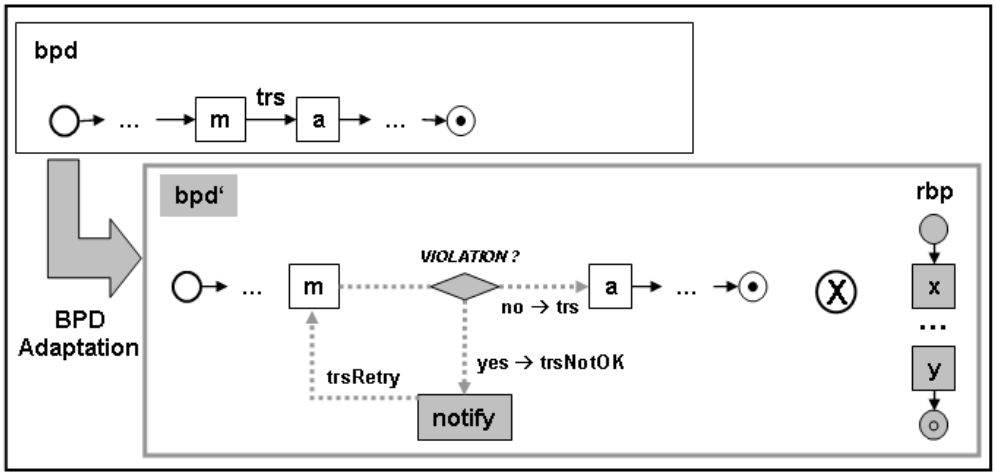

Figure 51 BPD Adaptation for Recovery Action model Retry & Notify & Instantiate

The new transitions exist in the adapted process definition *bpd'* in order to achieve the following functionality at a later stage during business process execution of *bpd'*:

1. The transition *trsNotOk*, in the case of a violation of the control, (*VIOLATION*-state of *ctl* has the value *Violated*), invokes the recovery action model *Notify & Instantiate*. For the specification of recovery actions, please refer to section 6.2.3.
2. The transition *trs* is a modified version of the original *trs* in *bpd* and only allows the process instance to later continue its original business logic (invocation of *activity a*) if no violation of *ctl* was recognized.
3. The transition *trsRetry* implements the behavior of the *Retry*- recovery action in *bpd'*. When it encounters a correct generation of a business document instance *msg* of type *Message* (its attribute "*to*" has the correct value), the process is continued in the activity *m*. We presuppose the existence of a business document of type *Message* having the attribute "*to*" indicating to which user the notification of control violation should be sent.
4. The transition *trsCtl* is responsible for updating the control state, which indicates that a control violation has been recognized and managed. (See *Control State Model* in section 6.1). For specifications on *state change command scc*, please refer to Definition 4.13. Since this transition is related to assuring the control objective (separated from business objectives) of a business process, it is not part of the process model represented in Figure 51. Indeed, technically the invocation of the *scc*-command updating the control state takes place in ICR-Execution, and will later be described.

Because the adaptation of process definitions for other selected recovery action models is very similar to the one presented above and can be derived using the one presented here,we will not provide other examples at this time.

8.3.2.2 Discussion

Using this approach, even if a compliantly designed and operating BPD becomes non-compliant (in terms of violating the conditions of a control), the control in the ICR will still detect the control violation which has occurred in the business process instance of that BPD. A business process is not compliant if it fulfills only its business objectives. The control objectives must also be fulfilled. A compliant business process can become non-compliant due to reengineering or the redesigning of a business process by a business process expert/developer

who is not aware of compliance requirements. In this case the implementation of recovery actions on the adapted BPD ensures that a business process instance does not continue its original execution course, but takes an execution path enforced by the recovery actions of the control definition. It is for the most part in this way that the approach enables compliance of a preventive nature. No business process instances that have caused a control violation will be allowed to continue. The BPD adaptation phase exists to extend an existing BPD by the necessary artifact in order to realize the above described behavior. It is important to note that any modifications of a BPD achieved during this phase are not related to the business objectives of the process, i.e. they do not adapt the original "business logic" of a process, but rather implement the consequences of the recovery actions- part of a control in a BPD. A business process instance based on such an adapted BPD during this phase cannot be considered as compliant, but is considered as **not** "non-compliant". The task of bringing a process model, and more specifcally a non-compliant business process instance, into a compliant form is still the responsibility of the relevant business process expert, who will be informed about the control violation (Concept of Forward Error Recovery, see section 6.2.3).

The cooperative interactions between the actors involved and the system during phase 1 (Compliance Design phase) and phase 2 (BPD Adaptation phase) are summarized in Figure 52. The descriptions of each numbered step in the figure are:

1. A compliance expert selects a relevant business process
2. Control Pattern Configuration by compliance expert:
    a. The compliance expert selects a certain control pattern and configures its pattern to specific parameters. If necessary, additional control statements are added to its control conditions
    b. The compliance experts sets the recovery actions of the control
3. Control activation by compliance expert
    a. The control pattern instantiation is invoked and the control is added into ICR-Design
    b. The BPD Adaptation is invoked on the relevant business process which in turn automatically generates an adapted BPD
    c. The original BPD is replaced by the adapted BPD
4. The relevant business process expert is informed of the creation and activation of a new control.
5. The recovery action model of the control is verified by the business process expert, regarding the business objectives of the business process. If necessary, the business process expert will modify the recovery actions of the control
6. Control activation by business process expert:
    a. The checked control will be stored in ICR-Design
    b. BPD Adaptation is invoked on the relevant business process
    c. The original BPD is replaced by the adapted BPD.

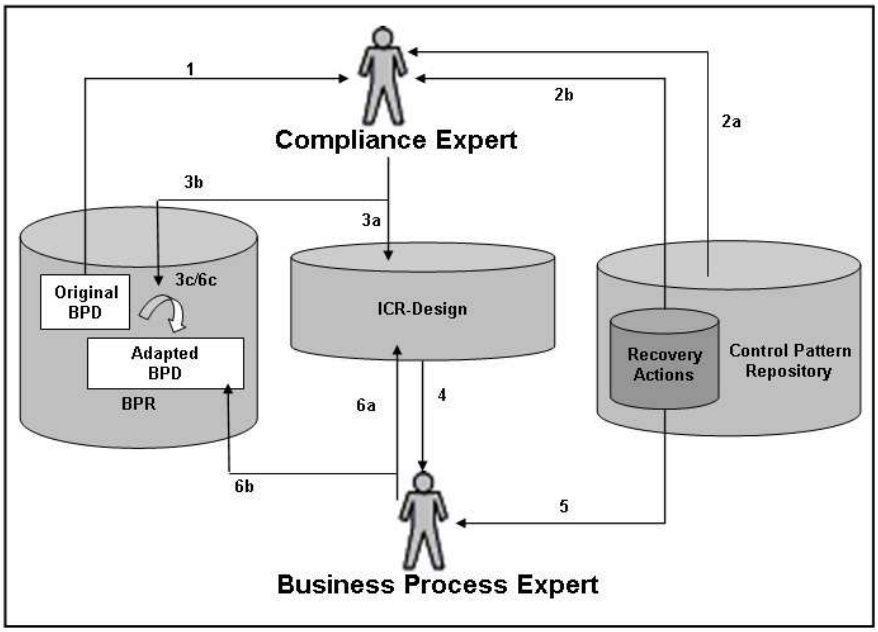

Figure 52 Overview of the interactions between roles and ICR-Design

8.3.3 Compliance Enforcement Phase

This phase enables the bidirectional interaction between business process management and internal controls management. It takes place during business process execution time and the component involved in the interaction with a business process execution infrastructure in this phase is ICR-Execution. In order to recognize and handle the control violations, ICR-Execution requires the following functional blocks (see Figure 53).

- Synchronize ICR-Execution
- Determine Control Violation
- Notify of the Control Violation
- Invoke recovery action

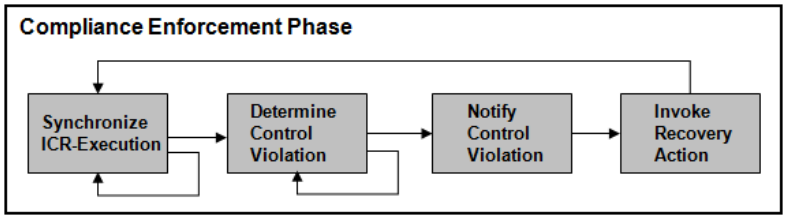

Figure 53 Required functional blocks during the compliance enforcement phase

The ICR-Execution will be continuously updated by information about the current instance of the business processes being enacted (Synchronize ICR-Execution). In the occurrence of any triggering control event, the control condition will be evaluated with help of the *CEQueries* (see Definition 7.3) in order to determine if it has been violated. If it has the ICR-Execution updates itself to take into account the fact that the control has been violated and then invokes the recovery actions in the control, which were defined during the compliance design phase and implemented during phase 2 of the approach (BPD adaptation phase).

In order to enable the automated generation of necessary information for ICR-Execution, it must be continuously updated, whenever a transition in the scope of a control is performed in a

relevant business process instance. The update of the ICR-Execution is done by the introduction of a Knowledge Base of Facts (*KBF*) which is enacted during the execution of a business process. With the help of the KBF, the current context of the business process instance (i.e. all relevant CEs) can be provided to the ICR-Execution.

The *KBF* updating the ICR-Execution are orthogonal to business process management and we introduce them on a conceptual level. The update mechanism of the ICR-Execution is dependent on the underlying BPM infrastructure, i.e. its description is a technical issue. The description of the technical realization will be provided in section 8.4. However, on an implementation level, the destination of a KBF is in all cases a network of addressable devices such as Trace/log files, a RDBMS, or a messaging destination such as a MQSeries/JMS's Topic/Queue. The destination of the KBFs can also be the ICR-Execution itself. This occurs when the underlying process execution infrastructure implements the *observer design pattern* or the *command design pattern* [Gamma et al., 1995]. The interaction of system components during this phase and the information exchanged between them is shown in Figure 54. It can be interpreted in the following manner:

- Business Process Instance Repository (BPRI, see Definition 4.17) contains the set of business process instances. It directly (or indirectly through the KBF) provides all data necessary to the evaluation of a control in ICR-Execution. This control relevant data is encapsulated in an entity adhering to the model of a business process instance according to Definition 4.17. The concrete information required by ICR-Execution is encapsulated in the business process context (see Definition 4.16), which provides the business document instances so far produced and consumed as well as the transition instances (see Definition 4.15) already enacted (*PATH* of the business process context). Further, the business process instance contains a reference to the current position of the instance.
- At the end of the above interaction, the control relevant data is extracted as a set of facts and is provided to ICR-Execution.
- Based on the control definition provided by ICR-Design and the set of current facts, ICR-Execution determines whether any control violations on any business process instances have occurred.
- If control violations are judged to have occured, this is communicated back to BPRI by changing setting the status of the violated control.

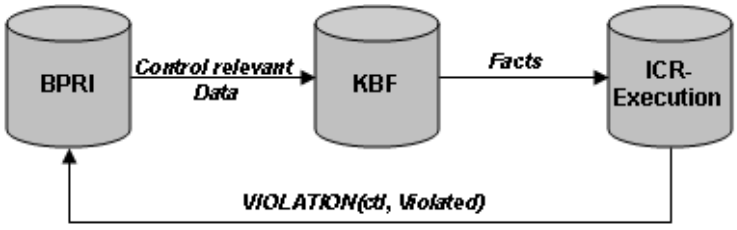

Figure 54 Interaction between a business process instance repository and ICR-Execution

In the following we describe the validation of a control *ctl* during execution time of a business process definition *bpd* with a recovery action model in the control of the form *Retry & Notify & Recover(rbp).* All steps are visualized in Figure 55:

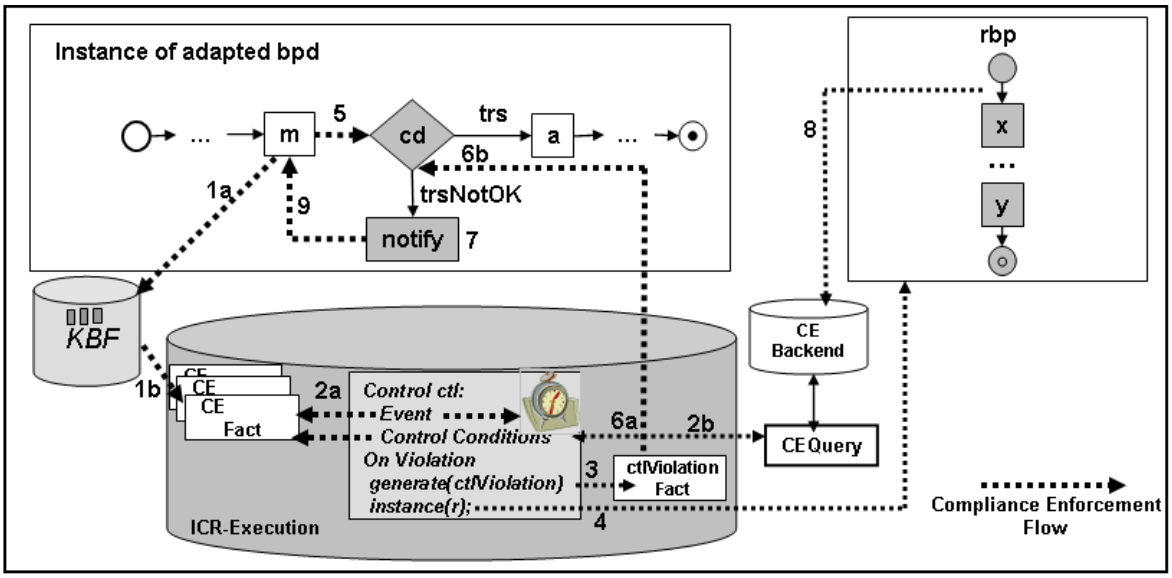

Figure 55 Interaction of a business process instance with a control (ICR-Execution)

1a. The control-relevant data is written to the KBF. Note that this can be done directly on the ICR-Execution itself (updating facts directly in it) depending on the underlying BPM Engine implementation. In this case, we continue with step 2a.

1b. The log entries are extracted and corresponding CE-facts are created and updated in the ICR-Execution.

2a. As the state of the ICR-Execution changes with the addition of new CE facts to or by updating previously existnet CE facts, the trigger of control ctl gets activated. The control condition of ctl is determined by the values of the CE facts in the ICR-Execution itself or by

2b. queries the set of facts (*CEQuery*). Consider that a *CEQuery* can collect data by forwarding the query to some other external systems (CEBackend).

3. If the conditions of the controls are violated, a new fact in the ICR-Execution (*ctlViolation*) will be generated signaling that control *ctl* has been violated (Corresponding to state *VIOLATION (ctl, Violated)* of a business document instance of *ctl*).

4. An instance of the recovery process *rbp* is generated and initialized with the necessary data.

5. The relevant business process instance *bpi* continues into the decision step (either transition *trs* or *trsNotOk*). This step is marked by the decision node *cd* in the figure.

6a. ICR-Execution will be queried to determine the *VIOLATION*- state of *ctl*.

6b. In the case of a *cViolation*-Fact with a *Violated*-value in ICR-Execution, the business process instance continues with the transition *trsNotOk*. Otherwise the transition *trs* will be made. In the latter case, the process continues as originally designed.

7. If *trsNotOk* is made, a notification message is generated for the responsible entity in the organization (*Notify* recovery action).

8. A business user or business process expert has the opportunity to process the *rbp* instance, which was generated as a recovery action of *ctl* (*instantiate*). As a result of this execution,

some CE will be updated. This may bring the business process instance to a state, in which it can be compliantly execute as specified by the control.

9. The process instance has now returned to transition *m* in the business process instance. The process instance will continue with step 1a.

Notice that the approach described above will detect a control violation in the ICR-Execution even if a business process expert/technical consultant removes the control from the process definition because he or she is not aware of the necessity of that control: the process context is always written to the ICR-Execution during step 1a/1b and the controls exist independently in the ICR-Design. Further, the described approach enables dynamic application of the controls during the execution phase of a business process. There is minimal overlap between business process design and compliance design. Thus, new controls can be designed for business processes by adding new control definitions to the ICR-Design, while the original design of the business process requires no manual change (BPD Adaptation phase is automated), which is one of the main advantages of the approach.

8.4 Implementation

Apart from conceptual soundness, one of the challenges inherent in such an approach is to assure the possibility of its efficient and scalable implementation. In this section we elaborate on the technical challenges we faced while implementing the approach during the internal SAP project which was introduced in section 1.2.

The system of the ICR is divided into two functional parts, ICR-Design and ICR-Execution. As we mentioned in the introduction of this chapter, we selected the production rules to implement the controls for the implementation of the ICR, and a RETE-Based Rule-Engine for the realization of ICR-Execution (Drools Rules). One may question why we did not choose the implementation approach selected in the verification of BP models (see chapter 5), which was a SWRL/OWL-DL-implementation. The answer, for the most part, relates to the nature of compliance validation of business process instances, which take place during the execution phase of business processes. A detailed discussion on the technical reasons for preferring a Production-Rules-Based implementation of the approach to SWRL/OWL-DL is given in section 8.4.1. Section 8.4.2 describes the implementation of phase 2 of the approach, the business process adaptation phase.

The main challenge regarding the implementation of the approach on top of a pre-existing BP- and Rule-Infrastructure is to find a way of integrating the business process instances running outside of the ICR. The integration must reflect the consequences of potential control violations detected in ICR-Execution. The integration must ensure the invocation of appropriate recovery actions. This is related to phase 3 of the approach, which is the compliance enforcement phase. The technical implementation of this integration in the system is presented in section 8.4.3.

8.4.1 Requirements for the Realization of ICR

The basis for the implementation of the ICR is the formal model of the controls (see chapter 6). The core model of a control can be expressed by an ECA-Rule.

- The triggering events and their scopes represent the even-part,
- the control condition cc represents the condition-part, and
- the RAS-part of a control definition represents the action-part

of an ECA-Rule. As mentioned in section 8.2.1, ECA-rules can be considered as a special type of production rule. From a conceptual point of view, we see ECA-rules as a production rule where the notion of *event* in ECA is explicitly designed in the left hand side (LHS) of a production rule. We decided to use a production rule execution engine infrastructure (Drools rules) for the implementation of the controls in ICR-Execution.

Further, we defined the following four requirements, necessary to the rule language and the infrastructure in order to effectively represent and execute a rule-based implementation of controls in the context of business process compliance:

R1) Business Process Instance Awareness
R2) Expressivity
R3) Actionable output to business process
R4) Querying external backend systems

Based on these requirements we finally selected two potential candidates for the implementation of ICR:

1) SWRL and associated inference engine and
2) Drools Rules (including its Rule Engine) recently adopted by JBoss Drools (JBoss Rules). It basically provides the same architecture and approach as Jess and we see it as representative of a family of business rule engines.

In the following we elaborate on each requirement. After each elaboration, we analyze how far the selected implementation alternatives, SWRL and Drools Rules, support that specific requirement.

8.4.1.1 Business Process Instance Awareness

The current state of ICR-Execution, i.e. the facts contained in it, is heavily dependent on the current business process instances and their context which is run inside the business process execution engine. In most real-life scenarios, we have to assume the pre-existence of an infrastructure for defining and executing business processes in terms of an ERP-system or a workflow engine. Usually the system of the ICR is independent of these infrastructures and they are not under its control. Thus the state in ICR-Execution is determined based on the context provided and changed from the outside. This means that the facts in ICR-Execution are not only generated or updated after a control is evaluated, it is possible that a BPM Execution Engine adds new facts or updates existing facts in ICR-Execution. Figure 56 illustrates the two possible ways that facts can be created or updated in ICR-Execution (Interactions in the figure are marked A and B).

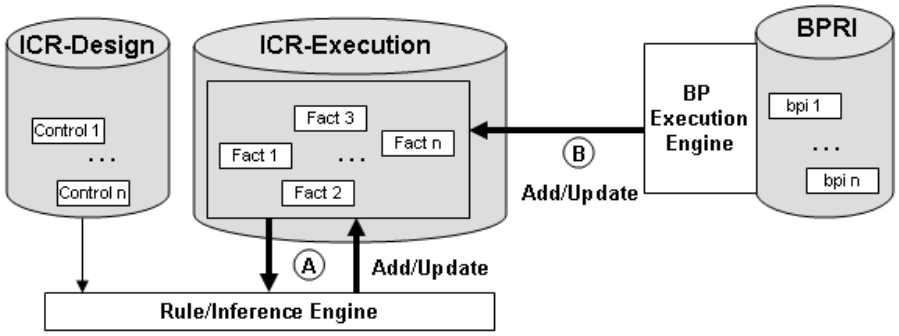

Figure 56 Possible Interactions (A and B) between BPRI (managed by BP Execution Engine) and ICR-Execution

In the following we further explain interactions A and B in the figure above:

Interaction A: Rule/Inference Engine adds/updates facts in ICR-Execution

As a rule representing a control is evaluated by a Rule/Inference Engine, the Rule/Inference Engine may consequently add new facts or update already existing facts in ICR-Execution as the result of the evaluation.

Example: An illustration of the example is given in Figure 57. Let us analyze the situation (time T_1) in which the ICR-Execution (its Rule/Inference Engine) determines that because a PR's total amount (an already existing fact in ICR-Execution belonging to a bp instance with *id = 4711)* is lower than 10000 $, the control for SSE is not violated even if there is only one *ApprovePR*-Activity enacted. This is signaled in ICR-Execution through an update of the according fact in ICR-Execution, e.g. *VIOLATION (ctlSSE, NotViolated)*. Later, (T_2) after a purchaser has updated her PR by setting the amount to 11000 $ (*UpdatePR*-Transition), the control is re-evaluated in ICR-Execution, before the execution of the *SendPO*-Activity. As the process instance again enters the scope of the control, the rule engine determines that for the same *PR* instance, the SSE control is now violated. Consequently, the previously added fact in ICR-Execution will be updated to *VIOLATION (ctlSSE, Violated)*.

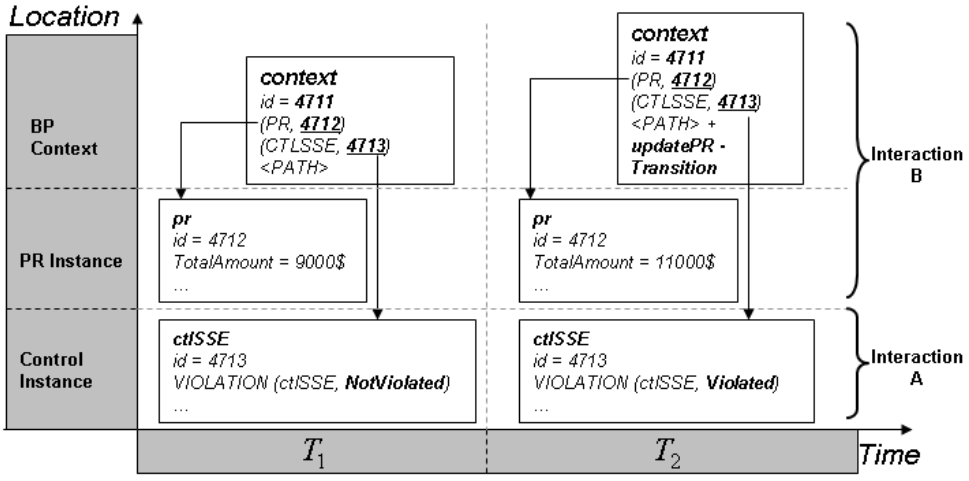

Figure 57 Example of Interactions A: Rule Engine adds/updates facts in ICR-Execution

Interaction B: BPM Execution Engine adds/updates facts in ICR-Execution

The enactment of a business process inside a business process execution engine may cause the creation of completely new facts or update already existing facts in ICR-Execution. Updating

existing facts in ICR-Execution is required for the compliance enforcement phase (see section 8.3.3) and visually shown by steps 1a and 1b in Figure 55 (Synchronization of ICR-Execution).

Example: An illustration of this example is also given in Figure 57 (marked as interaction B). When creating a *PR* business document instance during the purchasing process by a BPM execution engine, the following fact is added to ICR-Execution: *pr*(…, *TotalAmount=9000*). When a purchaser updates this already existing PR by increasing the amount of the PR document instance to 11000 $, the fact that was previously added to ICR-Execution must be updated to *pr*(…,*TotalAmount =11000*) in order to correctly reflect the state of the business process instance in ICR-Execution.

8.4.1.1.1 Business Process Instance awareness with SWRL

The use of SWRL for the implementation of ICR-Execution in a business process context did have the following limitations: When a user implements an ICR-Execution based on SWRL, the ICR-Execution must be continuously synchronized as an open world system with the *monotonic assumption* by the business process instance, and its context parameters must be provided by a closed world data-base-centric system (BPM Execution engine).

When it comes to the synchronization of the open world ICR-Execution, the originally existing closed world system forces us to simulate the closed world behavior in an open world environment. This is due to the fact that, when evaluating a control condition, i.e. when executing a rule by the inference engine, we have to assume that every fact used by the rule refers to the most recently provided business process context in the BPM Execution engine. Otherwise, the execution of the rule in an outdated business process context may result in a different and incorrect conclusion with regard to the violation of a control. For example, if a PR with a total amount of 9000 $ is not approved twice, the end of the specified control scope leads to a different conclusion than that of a PR with an amount of 11000 $ which was not approved twice. Thus this approach actually works in closed world environments, where each property of a specific fact maintains its most recent value (Concept of Truth Maintenance, see section 8.2.2.4). In other words, when a new value for a fact is provided, the previous value is overwritten. But in an OWL/RDF implementation of the facts in the ICR-Execution, monotony causes the approach to be less straight-forward. This issue is discussed in [Matheus et al., 2005] and two solutions to this problem are therein outlined. We summarize: 1) The closed world behavior can be manually implemented in an open world environment through an external management of the incoming facts and the removal of the inconsistent tuples in ICR-Execution logically inferred based on the "older version" of the fact. 2) Every fact is provided with a timestamp and added to ICR-Execution. For compliance validation, the most recent fact, i.e. the one with the highest timestamp, is taken. However, both approaches will significantly increase the required computation resources as well as the complexity of the approach's implementation.

8.4.1.1.2 Business Process Instance awareness with Drools Rules

Through the usage of jBPM and Drools we are able to implement a real-time business process instance-aware ICR-Execution. This is made possible by the fact that all facts are added to ICR-Execution as "*Shadow facts*" supporting the concept of truth maintenance (see section 8.2.2.4). We discuss its role in our approach in the following:

For each fact in ICR-Execution, whether it was created by the Drools Rule Engine or by jBPM, a iava bean class [JavaBean] is developed. Thus for each fact in ICR-Execution a corresponding java bean instance exists in the Java Virtual Machine (JVM).

Java beans provide a component architecture that enables easier integration of applications. A property change notification mechanism is supported there that allows one object to become a registered listener of another object. The listener object will then automatically receive changes from the source object. This is the java-based implementation of the observer pattern [Gamma et al., 1995]. Within Drools, each java bean corresponds to what is known as a shadow fact. Thus a shadow fact is a "mirror image" of a java bean instance. All shadow facts are registered listeners of their java bean counterparts. Thus, whenever a java bean instance changes in the BPM Execution engine, a property change event is automatically generated for the given java object instance and its corresponding shadow fact is updated in ICR-Execution. Figure 58 shows the update-path of an already existing fact related to a purchase request (PR) from a Purchasing process instance inside a BPM Execution engine, to a corresponding shadow fact in ICR-Execution.

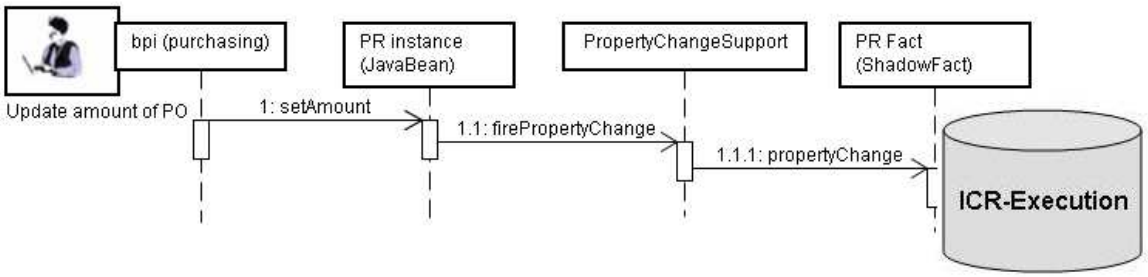

Figure 58 Sequence diagram showing an update to ICR-Execution from a business process instance bpi

8.4.1.2 Expressivity

The language must provide constructs which make it possible to directly or indirectly express the control definitions as ECA-rules. Directly or indirectly means that we do not require constructs to express the control definitions as one single statement. Several equivalent logical statements representing one control definition are also acceptable.

8.4.1.2.1 Expressivity with SWRL

The SWRL Rule format [Horrocks et al., 2004, Horrocks et al., 2004b] has to be mapped to the form of a control definition (an ECA).

To express a control definition, all the terms in a control definition as described in 6.2.2 must be expressed as atoms that constitute the antecedent and the consequent in SWRL. SWRL as an extension of OWL, along with the classes and properties that are defined in OWL, are then both used to define a control as a constraint over those classes, properties, and instances. The antecedent of a control definition consists of an *Event* of a *control condition* describing a control *Violation* in a business process instance.

8.4.1.2.2 Expressivity with Drools Rules

Drools Rules provides production rules that are not conceptually ECA rules, but the Event-Model of a control (as described in section 6.2.1) can be designed in the LHS of a production rule with acceptable overhead. Thus the challenge when using Drools Rules in order to represent an ECA-rule is to provide a sufficient and easy-to-use event-object-model that can be used to construct a control definition. Such an event-object-model was specified by Definition 6.7.

Further, using the different types of provided conditional elements in Drools rules (see section 8.2.3.3.1) allows the expression of control statements (and control queries) in a control condition.

8.4.1.3 Actionable Output to Business Process Instances

The approach requires that an activity advised by a compliance expert and codified in the control statement in terms of the recovery action be executed in the business process instance in the case of a violated control. This actually means that when a violation is determined in ICR-Execution, the state of things in the outside world, i.e. the current business process instance running in a BPM Execution engine, has to be updated according to the recovery action. This update mechanism is triggered inside the ICR-Execution (see section 8.3.3).

8.4.1.3.1 Actionable Output with SWRL

SWRL allows the use of constructs in the consequent– part of a rule statement or built-ins which are modeled in OWL. SWRL does not provide any mechanisms to invoke operations outside of the OWL/SWRL knowledge base. When implementing the ICR with OWL and SWRL, the architecture must be realized in such a way that a separate component updates the business process instance in the BPM Execution engine.

8.4.1.3.2 Actionable Output with Drools Rules

Drools provides different mechanisms to affect the outside world based on the rule execution:

1) In the right hand side (RHS) of a Drools-rule, a shadow fact can be modified or retracted.
2) Methods on java-API level, which modify the according java object instances, can either be invoked directly within the rule itself or can be separately implemented in a function used in the rule.

Thus when implementing the ICR based on Drools we can realize the architecture in such a way that, based on the control-evaluation results, ICR-Execution updates the outside world in a push-driven-manner. The mechanism of the shadow facts can be used to modify the current business process instance that caused the violation directly from ICR-Execution. Recall that the parameters of the current business process instance are available in ICR-Execution as shadow facts as well. Thus their update causes the automatic update of the corresponding process instance in the BPM Execution engine.

8.4.1.3.3 Querying External Backend Systems

This requirement is closely related to the requirement of business process instance awareness of ICR-Execution.

During our analysis of different types of controls and the different ways in which it is possible to evaluate their conditions during execution time of business processes, we realized that it would be expensive to keep the ICR-Execution completely synchronized with the heterogonous environment with which we were faced. The heterogeneity is given through different backend systems containing different operational data such as SRM (Supplier Relationship Management) systems, CRM (Customer Relationship Management) systems etc, which contain relevant information about orders, contracts, business transactions etc. Further, the information necessary to the evaluation of a control condition cannot always be provided by the context of a business process instance itself. For example, consider a control pattern adhering to a temporary authorization pattern, let's say the "One-Time-Supplier-Creation" control (see use case of CustomerB in section 2.3). Such a control definition contains a control statement (see Definition 6.8) of the type *EXECUTED (usr, trs)* respectively *EXECUTED (trs, n, m, f)*. In order to evaluate

such a control it must be determined how often a certain user has executed a certain activity in a transition, in this case “One-Time-Supplier-Creation”. In this case it is not sufficient to determine whether the current user has invoked this transition in the current business process instance. All business process instances adhering to the given business process definition (in this case purchasing) and processed during the period defined the in control must be checked for transition instances of the transition in focus in order for the control to be evaluated. Thus the necessary information cannot be provided by the current business process instance. In this case, other backend systems containing data about previous user-transactions (such as LDAP etc.) have to be queried to collect the necessary input to evaluate the control conditions. This basically means that ICR-Execution will have to access the (transactional) data outside the BPRI and ICR-Execution in order to evaluate the condition part of the control.

Figure 59 illustrates the situation described above: The set of facts required for the evaluation of a control is A. Set A itself consists of two subsets B and C. Set C contains facts, which can be provided directly by the current business process instance *bpi*, for which the control must be evaluated. Set B contains those facts which have been produced by earlier executions of business process instances (set D in BPRI). The members of D are either already terminated process instances or those currently in execution. In order to create B, ICR-Execution must be able to collect the necessary facts from relevant backend systems by querying those backend systems (covered by concept *CEQuery*, see Definition 7.3). The target of these queries is either the ICR-Execution itself (the facts in it) or a relational database of the according backend systems, in which the operational data is stored.

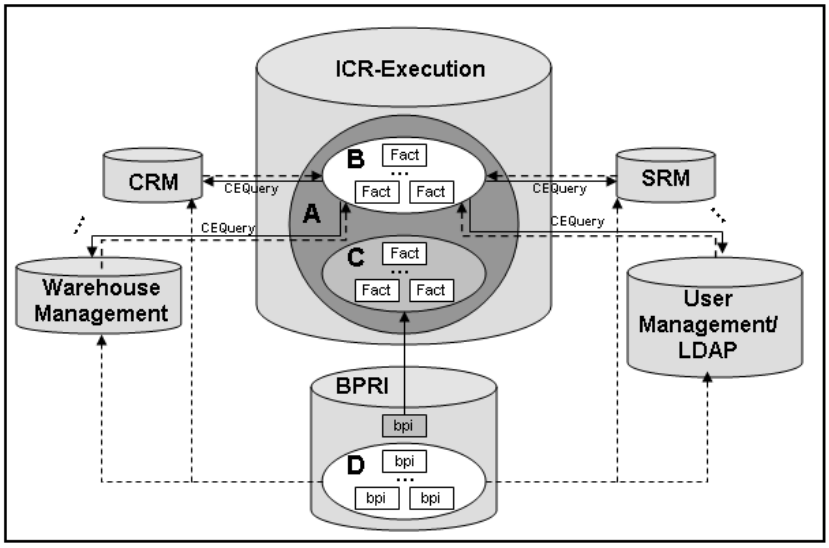

Figure 59 Querying data from external systems by ICR-Execution

8.4.1.4 Querying External Backend Systems with SWRL

SWRL is by nature a rule language and not a query language. RDF query languages like RDQL [RDQL] and SPARQL [SPAQRL] can provide SQL-like query functionality on triple stores. OWL ontologies can be stored in triple-store backends without loss of semantics.

Further, with SWRL Query Built-In [SQWRL] a built-in is provided to define queries on an OWL Knowledge Base. It defines a set of built-ins that can be used in SWRL rules to query OWL ontologies. The built-ins in this library can be used to turn SWRL into a query language. They provide SQL-like operations to format knowledge retrieved from an OWL ontology. The

resulting query language complies with the standard SWRL syntax and does not alter the semantics of the language.

The main issue when using SWRL to retrieve the data necessary for the evaluation of controls is that SWRL, and also SQWRL, require an OWL-Knowledge Base (Ontology), from which necessary data can be extracted. Unfortunately, for now and for the foreseeable future, most data will continue to be stored in relational databases [O'Conner et al., 2007]. To bridge the gap between the different underlying data models, i.e. relational and triples in case of SWRL/OWL, the relational data has to be reformatted into a SWRL-processable form, namely triplets.

One possible way to solve the problem would be to statically map a relational database to a triple-store. However, this approach suffers from several shortcomings: There is an issue of data duplication and there are questions about how frequently triple stores should be updated in order to reflect changes in the associated relational database. Applications with permanently changing data, (such as is the case in business process compliance), and requiring up-to-date information about business process executions require frequent synchronization, which may be cumbersome and problematic. Similary, supporting logical updates on the replicated data means that synchronization issues arise in the reverse direction.

[O'Conner et al., 2007] propose another approach for automatic or semi-automatic dynamic (i.e. during execution time of the underlying application) mapping between relational databases and triple-based formats. This is achieved in a separate software layer where SWRL-level queries are mapped into SQL queries in order to retrieve the required data from a database.

However both approaches described above represent a significant overhead for the realization of querying external backend systems with SWRL: Replicating the relational data in an OWL ontology as described in the first approach leads to problems of synchronization between the two sources. Implementing a separate software layer for mapping the SWRL/OWL based queries into SQL queries and turning the retrieved relational data back to OWL significantly raises the level of effort necessary for implementation as well as the complexity (execution time) of that implementation.

8.4.1.5 Querying External Backend Systems with Drools Rules

Relational data can be accessed in the LHS of a production rule in Drools by invoking a so-called Data Access Object (DAO) with the help of the *from*-conditional element (see section 8.2.3.3.1). DAO is a software design pattern encapsulating the data retrieval functionality with methods implementing the access to backend systems and retrieving data from there according to a specified set of retrieving filters. The result of such a method's invocation would the required set of data, which comes from a persistent media such as a relational database. A *CEQuery* would be implemented technically as such a DAO. When invoking a method of a DAO from the LHS of a production rule, the retrieved data represent facts in the working memory of the rule engine.

Figure 60 illustrates the architecture of querying external backend systems when using Drools Rules in a java-like notation: In the business process layer, an instance of a DAO (lets say *TransitionInstanceDao*) is created and initialized with the context of the current business process instance. This way the DAO is initialized with the user and role, (and possibly other CEs in a business process), currently executing the business process. The DAO in this case will retrieve the information required about the current user. The instance of the DAO is asserted in the working memory (step 1 in Figure 60). When the event specification of a control matches (in case of production rules some facts in the working memory fulfill the conditions in the rule), then the corresponding method of the DAO instance (now a fact in the working memory) will be invoked with the necessary retrieval filter parameters (set during control design phase, step 2). Next the

implementation of the DAO retrieves the data according to the filter (step 3) and returns the answer (number n) back to its caller (step 4). This way it is possible to implement a production rule which evaluates a rule based on a set of facts which were not asserted to the working memory by the application (business process layer).

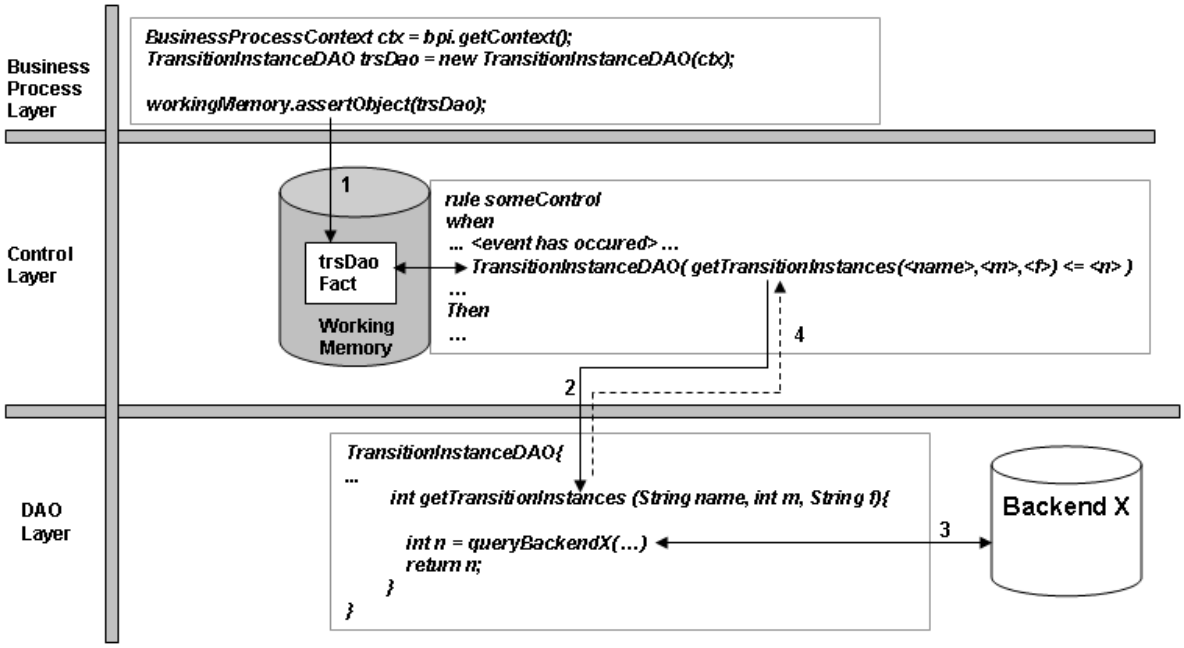

Figure 60 Querying an external Backend System (X) by a production-rule-based ICR-Execution

8.4.2 Realization of the Business Process Model Adaptation phase of the approach

Implementation of phase 2 of the approach includes two tasks:

1. Implementation of an Instantiation-mechanism for a given *bpd* in the case of recovery action *Instantiate(user, bpd)*
2. Modification of an existing jPDL process definition in the case of all other recovery actions.

The implementation of the above tasks is described in the following two sub-sections.

8.4.2.1 Implementing *Instantiate*-Recovery Action

The implementation of the instantiation functionality is related to the requirement of actionable output for business process instances (see section 8.4.1.3). As we have mentioned, the implementation is encoded as a Drools-function-call in the RHS of a production rule. When a control violation is detected, i.e. the facts in the working memory match the LHS of a rule, the function in the RHS is invoked. The encoding in a rule looks as follows. Comments are given inline in the listing:

Listing 9.5

```
rule "control X for business process <bpdId> "
    when
```

```
        // retrieve a fact of type ProcessInstance with a bpdId-parameter having the value
        // <bpdId>. The retrieved fact will be written in variable $pi, in order to be used
        // in other rule-parts
        $pi : ProcessInstance( bpdId = <bpdId> )

        // retrieve the business process context belonging to the selected process instance
        // ($pi). The resulting fact will be written in variable $context
        $context : BusinessProcessContext( bpiId = $pi )
        <events>
        <conditions>
    then
        …
        // the function createProcessInstance will be invoked. The input-parameter $
        // context is retrieved in the LHS of the rule. The input-parameter <rbpdId> and
        // <userId> specifies the process definition that an instance of it will be created.
        // The input parameter specifies the user to whom the created process instance
        // will be assigned, i.e. who will process it. The parameters <rbpdId> and
        // <userId> are part of the specification of the Instantiate-recovery action, thus
        // they are defined by the compliance expert during control design.
          createProcessInstance ( <rpbdId>,<userId> , $context ) ;
end
```

The implementation of the *createProcessInstance*-Drools-Function in the rule represented in Listing 9.5 looks as follows (the explanation of the code is inline in the code):

Listing 9.6

```
function void createProcesInstance (String rbpdId,
                                    String userId,
                                    BusinessProcessContext ctx)
{
    try {
        // get Reference to JBPM Execution Engine:
        jbpmContext = ctx . getJbpmContext ( ) ;

        // Load the definition of the given process (bpdId) :
        graphSession = jbpmContext . getGraphSession ( ) ;
        ProcessDefinition  processDefinition = graphSession .
                                    findLatestProcessDefinition ( bpdId ) ;

        // Create an instance of the process:
        ProcessInstance rpi = processDefinition . createProcessInstance ( ) ;

        // Create instances of roles and user modules
        SwimlaneInstance = getRoleOf(userId);
        TaskMgmtInstance tskmgtInstance = new  TaskMgmtInstance();
        SwimlaneInstance role = getRoleOf(userId);
        role.setActorId(userId);
        tskmgtInstance.addSwimlaneInstance(role);
```

```
                // Assign the created module in the process instance. This causes that
                // the process instance will be created as a task in the task list of the
                // specified user with id userId:
                rpi . addInstance(tskmgtInstance);

                // Start the process instance:
                 rpi . signal ( ) ;

                // A process instance of rbpId is now created and its first task is
                // assigned to user userId
        }
        catch ( Exception ) {
                // exception handling
        }
}
```

8.4.2.2 Modification of a jPDL- Process Definition

The modification of a jBPM process definition can be implemented in two ways: i) Using the Design-time API provided by jPBM to load an existing process definition and then to add the necessary elements in the process definition via the jBPM-API, or ii) modification of the process definition on an XML level via the XML-API. We selected the second alternative for the implementation of modifications: we chose to directly modify the process definitions via the XML-API by the use of XSLT.

The following is an example of a jPDL process definition before and after adaptation. The selected recovery actions were *Retry & Notify* (see Figure 61).

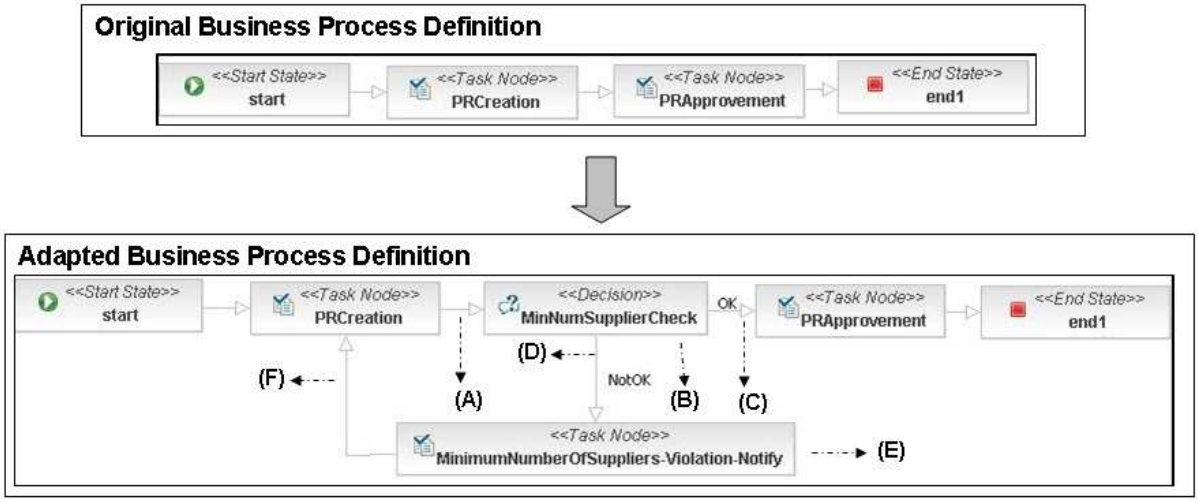

Figure 61 Application of BP Model Adaptation on a jPDL process

This is an example of the XML of the adapted process definition, where the modifications are marked in the code and shown in Figure 61 (Marked positions A – F).

Listing 9.7:

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definition
 xmlns=""  name="purchasing">
 <swimlane name = “PD">
</swimlane>
  <start-state name="start">
  <task name="Start Process"></task>
    <transition name="" to="PRCreation"></transition>
  </start-state>
  <task-node name="PRCreation" create-tasks = "true">
    <task name="Please enter purchase request data" swimlane = “PD">
     <controller>
       <variable name="material type" access="read,write,required"></variable>
        …
     </controller>
    </task>
    <event type = "node-leave">
     <action name = "validation"
                 class = "com.sap.research.icr.execution.ICRSynchronizer">
     </action>
    </event>
    <transition name="" to="MinNumSupplierCheck"></transition>          (A)
  </task-node>
  <decision name="MinNumSupplierCheck">
     <handler  class =                                                  (B)
        "com.sap.research.icr.execution.ControlEvaluator">
           <variableName>MinimumNumberOfSuppliers</variableName>
     </handler>
       <transition name="OK" to="PRApprovement"></transition>           (C)

      <transition name="NotOK"
                  to="MinimumNumberOfSuppliers-Violation-Notify">       (D)
      </transition>
 </decision>
  <task-node name="PRApprovement" create-tasks = "true">
    <task name="PurchaseRequestApprovement" swimlane = “PD">
    </task>
    <transition name="" to="end1"></transition>
  </task-node>
  <task-node name="MinimumNumberOfSuppliers-Violation-Notify"
                  create-tasks = "true">                               (E)

    <task name="The Control MinimumNumberOfSuppliers has been violated. "
          swimlane = “PD“/>

    <transition name="" to="PRCreation"></transition>                  (F)
  </task-node>
```

<end-state name="end1"></end-state>
</process-definition>

8.4.3 Integration of Business Process Instances with ICR-Execution

For the task of synchronization of the ICR-Execution (phase 3 of the approach), we use the functionality provided by the jBPM Engine implementing the command software design pattern [Gamma et al., 1995]. This functionality is implemented in a synchronization component that will be described in section 8.4.3.1.

The execution course of a business process instance is decided (based on the occurrence of a control violation detected in ICR-Execution) using a decision component. This component will be introduced in section 8.4.3.2. The interplay of the synchronization component and the decision component during the compliance enforcement phase will be given in section 8.4.3.3.

8.4.3.1 Synchronization-Component

jBPM provides the possibility of registering (during design-time) an *ActionHandler* – Implementation to each node-class (activity) of a jPDL-Process definition. The implementation of the ActionHandler-interface can invoke additional custom functionality. We introduced this jBPM functionality in section 8.2.3.1.2.

Our implementation of the *ActionHandler*-Interface is called *ICRSynchronizer* and it obtains a reference to the ICR-Execution (in terms of obtaining a reference to the rule engine's working memory). The current instance of the business process is provided automatically by the jBPM Process Execution Engine to the handler (input parameter of the execute-method *ExecutionContext*). Based on the current instance, necessary data is collected and encapsulated in the business process context. The instance and its context are then asserted to the working memory as facts. Since we use the shadow fact functionality, each update of the *CE* corresponding to the facts previously asserted to the working memory on the business process execution layer will automatically be updated in the working memory as well.

In order to assure a continuous synchronization of the ICR-Execution, each node in the jPDL process definition which has an ActionHandler attached to it, (a node which potentially contains some business logic), is automatically equipped with an *ICRSynchronizer-ActionHander*, (a custom extension to the jBPM jPDL-Modeler-Tool). To avoid performance drawbacks, the implementation of *ICRSynchronizer* keeps a reference to the working memory and checks if the current business process context is modified by the current business process instance. This means that only if there exist completely new business document instances or transition instances (*PATH*-parameter of business process context, see Definition 4.16) in a business process instance, will the working memory be updated by these new facts through *ICRSynchronizer*. With this mechanism the resource overhead of synchronizing the ICR-Execution is reduced by a reduction of the calls from the business process execution to the ICR-Execution.
The method we just described requires no extension or change in the internal implementation of the process execution engine and is completely loosely coupled.

8.4.3.2 Decision Handling Component

The execution course of a business process instance is automatically determined based on the facts in the ICR-Execution. In the case of a certain control violation, a fact will be asserted in the RHS of the rule which detected the control violation. The relevant part of a rule for achieving this behavior looks as follows:

Listing 9.8:

```
rule "control<X> for business process <bpdId> "
    when
        …
        $control : Control ( name = <X>, bpd=<bpdId> )                (1)
        $pi : ProcessInstance( bpdId = <bpdId>, id = $control.piId )  (2)
    then
        …
        $control . setViolationStatus ( "Violated", $pi ) ;           (3)
end
```

In line (1) of the rule in Listing 9.8, the working memory will be retrieved to check whether there exists a control instance (fact) for the business process identified by *bpdId*, for which the control definition exists. A default control fact was previously added to the working memory, through assertion of the business process context, by the *ICRSynchronizer*. Recall that the business process context contains the business documents and their instantiations in a business process instance. Since a control itself is treated in a business process as a business document, it is available as well in business process context and accordingly in the working memory. In line (2) of the rule the process instance of the current control instance is retrieved from the working memory. The process instance written in variable *$pi* is required in the RHS of the rule (line 3), in order to set the *VIOLATION*- status of the control for that process instance on the value "*Violated*" in case the LHS of the rule detects a control violation.

During business process adaptation (phase 2) of the approach, a decision-node was added in the jPDL process definition (see *MinNumberSupplierCheck*-Decision Node in the XML-definition of the adapted jPDL inListing 9.7). This decision-node is equipped with a *DecisionHandler*-Implementation (*ControlEvaluator*). Our implementation of this *DecisionHandler* establishes a connection to the ICR-Execution and queries it (Drools Query, see section 8.2.3.3.2) in order to determine an instance of a control with *VIOLATION*-status having the value "*Violated*". *ControlEvaluator* knows for which instance control definition must search in the working memory, because the name of the control-name is set as a parameter in the XML-configuration-part of the *ControlEvaluator* in the jPDL definition (*variableName*-node in XML-definition,see Listing **9.7**). In this way the implementation of a separate *DecisionHandler*-implementation for each control definition is not required.

However, when a violated control exists in the working memory, *ControlEvaluator* forces the process execution (the next transition) to the transition which was set during the business process adaptation phase (recovery action). If no violated control fact exists, *ControlEvaluator* will take the originally designed transition in the current business process instance.

A summary of the steps described in this and the previous sub-section is provided in the next sub-section.

8.4.3.3 Sequence of Component Interactions

This section summarizes the implementation of the approach we developed for the integration of business process instances (run in a business process execution engine) and control instances (run in the ICR-Execution). It visualizes the sequence of interactions between the parties involved in Figure 62. They are:

- A business process instance (*BPInstance)*
- *ICRSynchronizer* as described in section 8.4.3.1
- *ControlEvaluator* as described in section 8.4.3.2 and

- A control in the ICR-Execution

We now describe the interactions (as marked in Figure 62):

1. *readControlParameters() :* As the business process instance is instantiated, it reads the required control names on each jPDL-node configured (*variableName*-parameter in decision-node, see Listing 9.7, area marked with B)

2. *setControlParametersInContext() :* The control name is set in the context. Now the context contains all controls on each node.

3. *execute(context)* : When the business process instance is executed, the *execute*-method of the *ICRSynchronizer* of each node containing business logic is invoked.

3.1. *isSynchronizationRequired(context) :* *ICRSynchronizer* checks whether the received context contains new business document or transition instances. Only then will it synchronize with the ICR-Execution.

3.1.1. *assert (facts)* : *ICRSynchronizer* fetches the context and asserts the necessary parameters from the business process instance as facts into the ICR-Execution

3.1.1.1*: setControlStatus (Violation)* : As the working memory is changed by *ICRSynchronizer*, the RETE-algorithm of the underlying Drools rule engine processes all controls and updates the agenda. In the case of a control violation, the control's *VIOLATION* status will be set to "*Violated*" in the RHS of the rule representing a control for the current business process instance.

4. *updateShadow(facts):* This is a continuously occuring interaction. Each updated CE in a business process instance will be reflected on its according shadow fact in the ICR-Execution.

4.1. *setControlStatus(Violation)* : The same as 3.1.1.1, with the difference that the ICR-Execution is not updated by *ICRSynchronizer*.

5. *decide (context) :* As the business process instance token has arrived at a jPDL-decision-node, its *ControlEvaluator* is invoked

5.1. *readControl (context) :* *ControlEvaluator* reads the *control name* from the context (set in step 2).

5.2. *query(controlName) :* *ControlEvaluator* queries the ICR-Execution for checking the existence of a violated control (with name *controlName*). In this case *ControlEvaluator* returns the name of the transition responsible for the recovery action of the control.

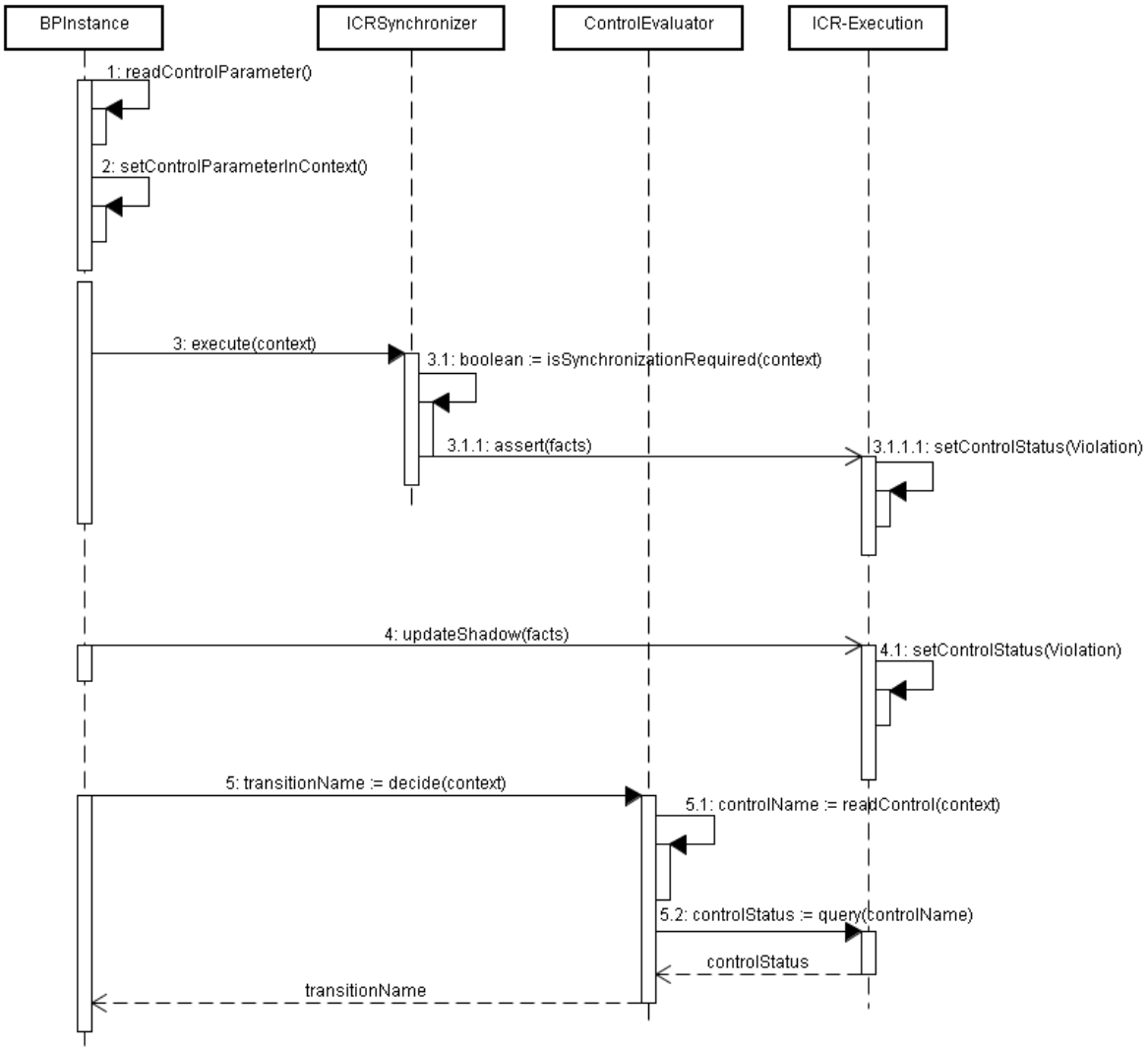

Figure 62 Sequence of interactions between components involved in compliance enforcement phase

8.5 Related Work

8.5.1 On Adaptive Workflows

We consider the research done in area of adaptive workflows as related to the approach presented in this chapter. There is significant amount of research done in the area of adaptive workflows, such as AgentWork [Müller et al., 2004] or AdeptFlex [Reichert et al., 1998]. In these approaches instances of business process are adapted during the execution of business processes, whenever a prescribed failure occurs. One could argue for the application of these approaches to the area of business process compliance. In the case of a control violation, the business process instance would be adapted dynamically during the execution of a business process and the instance could continue to process compliantly. Theoretically, these approaches would then have two advantages ours lacks:

1. Phase 2 of our approach, the business process adaptation phase, which takes place in parallel to the business process design phase, would happen later, during the business process execution phase.
2. There would be no need for the concept of forward error recovery processed by a business user or a process expert.

However, the concept of adaptive workflows during execution time of business processes can not be applied to the domain of business process compliance for two reasons, one from a technical perspective and the other from a business perspective, relating to the way in which internal controls compliance is certified. We discuss:

From a technical point of view, in the context of dynamic adaptive workflows, all approaches known to us apply the algorithms for dynamic adaptation of instances on an execution infrastructure implemented specifically for the approach. These concepts and their implementations do not represent a universal result, applicable in any technical environment. For example, in our experimental implementation on a jBoss jBPM-based process execution infrastructure, the dynamic adaptation of a business process instance was not realizable in that infrastructure during the execution of business processes. The main reason is that, although we were able to adapt a business process instance on the java object level, it was not possible to persist the adapted instance in the database of jBPM, if the business process instance goes into a wait state and is waiting to be processed at a later point. The reason the process cannot be persisted is that most commercial and popular open source process engines and ERP systems check a business process instance to verify whether it fulfills its original model before the instance is persisted. In those cases where the instance does not satisfy the original model, a runtime exception is thrown by the engine. The implementation of the dynamic instance adaptation in most cases then requires significant modifications to the core engine of the infrastructure in order to realize the approach. However this is not the objective of our research. We argue that in most cases a predefined business process execution environment or an ERP infrastructure will exist. The challenge is to couple the compliance validation framework with the already existing infrastructure, as was shown in our case with the ICR. Our approach has a universal character and does not depend on a specific business process execution infrastructure to be realized.

From a business perspective, i.e. from the point of view of compliance certification, in typical internal controls compliance projects, external auditors check whether the enterprise has documented its business processes (business process definitions) and its controls and whether the business processes work (execute) as designed in their definitions and controls. In the case of

dynamic adaptive business processes, it is per concept of the approach not guaranteed that business process instances will execute as designed, because they can take any other execution course as implemented in the runtime adaptation algorithms in use. This is a significant contradiction to the methodology required for assuring the business process compliance desired by official bodies. In our case, an adapted business process after phase 2 of the approach becomes the business process definition that will be presented to external auditors. Using the compliance enforcement phase we are able to prove that business process instances behave as described in the adapted business process definition. This is due to the fact that the recovery actions defined in the control definitions are implemented in the adapted business process definition during phase 2. A special case in this context is a selected recovery action of type *instantiate*. In that case, the recovery action is not reflected in the adapted business process definition, but in the RHS of a rule in our implementation. However in this case the implementation of the recovery action is stored in the ICR (and not on the business process layer), and the consequence of its invocation is again reflected in a pre-designed business process definition (recovery business process), which will be instantiated in order to correct the conditions in the system that cause a relevant business process to violate a control. The recovery business process is again in the focus of compliance certification (business process layer) and can be presented to certification bodies.

8.5.2 On Industrial Solutions

Software providers offer solutions for the problems encountered when tackling the management of compliance requirements (such as SOX) which are related to our work. They can be divided into two categories: 1) software which supports the realization of detective controls, and 2) software which supports the realization of preventive controls.

8.5.2.1 Commercial Tools supporting Detective Controls

The providers in the first category originate from the business intelligence and analytics, enterprise reporting, and data warehousing/mining areas. Some of the prominent software companies here are SAS, COGNOS, Business Objects, and MicroStrategy. Based on collected information produced in an enterprise, the reliability of financial statements can be supported by detecting the occurrence of certain patterns of possible control violations in the produced data. Periodic reports can also be generated about different transactions in an enterprise in alignment with internal controls. Those reports can then be either manually or automatically processed to identify certain control deficiencies in the operations and financial statements of an enterprise.

The requirement for realizing detective controls is gathering the necessary compliance data produced during business process executions. Based on the collected information, they can then be audited and possible control violations and deficiencies which have already taken place can be discovered. The specific compliance-related information collected is usually called *Audit logs*, which are useful for post-checking the enforcement and effectiveness of controls. The audit logs do not only have to be collected, they also have to be managed. In [Ramanathan et al., 2007], IBM sets out the requirements of a service for collecting and managing audit logs, which is called *audit service.* The audit service can be used by a given IBM product, such as Tivoli or Access Manager for e-business, or others, to enhance its auditing capabilities. The service can also potentially be used by any IBM application implementing operative business processes. Each product using the audit service has to produce the audit logs sent to the audit service in a certain format, called Common Base Event (CBE) [CBE101]. The audit service then stores the audit logs received in a relational database called an audit database that is tailored for the storing of large

volumes of data and also provides utilities that help with the life cycle (reporting, archiving and restoration, etc.) of audit logs. The content of the audit database can then be used in a query or to generate reports as a basis for detective controls. Reporting facilities such as IBM DB2 Alphablox or other business intelligence solutions, some of them mentioned above, can then be used to analyze the data. The main technical shortcoming of the audit service provided by IBM is that it can only be used by IBM products, i.e. only IBM products can invoke the service. It also represents a manual approach insofar as that the source system representing the operative business process application has to gather the necessary compliance data manually, map it to the log format required (CBE) and then invoke the service in the productive code. Thus during execution of the processes the data can be tracked by the audit service to the audit database. Because there is no clear conceptual separation of compliance and business process design, it is the responsibility of the programmer of the source system to ensure the auditability of a business system.

8.5.2.2 Commercial Tools supporting Preventive Controls

The industry supporting preventive controls does so generally from two different angles: 1) Business Rule engines and 2) Workflow/Business Process engine providers.

8.5.2.2.1 Business Rule Engines

Business Rule product providers such as ILog [ILOG] or Corticon [CORTICON] provide a generic rule framework to express conditions on a target system (in our case business processes). The general architecture common to most business rule providers is depicted in Figure 63: Through an editor, the business rules are entered into a business rules repository. The business rules can then be processed by an engine. These components are developed by the business rule product provider and deployed at a customer enterprise. However the software is not usable for business process compliance directly out of the box, because the rule engines are kept generic. They have to be provided with some data, which they can then process according to the algorithms that the rule engine implements. However, what we have depicted in Figure 63 as a "Target System" are business process instances in an enterprise. Through an adapter component, which has to be implemented at the customer enterprise, the data produced during business process executions will be sent to the business rule engine. In this approach the controls have to be modeled and implemented manually on already existing business processes, since there is no clear formalization and conceptual separation of the controls from business processes. The introduction and integration phases of the traditional rule engines at enterprises currently represent significant overhead because they are mostly decoupled from business process management models and infrastructures.

Although we use a rule-based approach to implement the approach presented in this chapter as well, the core difference is that we built our approach on well-defined models of controls and business processes, which is not the case with a plain rule engine provided by business rule engine providers.

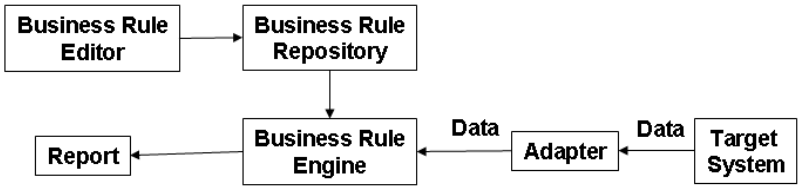

Figure 63 General Architecture of Business Rule Engines

8.5.2.2.2 Business Process/Workflow Providers

In the area of Business Process/Workflow Providers there are two approaches: 1) controls are implemented inside the processes during the design of those processes, or 2) there are solutions provided which offer only the documentation of existing controls without considering their application on process designs or checking their effectiveness during the execution of the processes.

One major provider in the first category is ARIS [ARIS]. We introduce and discuss ARIS solutions for compliance as a representative of software providers in the area of business process/workflow.

The ARIS core products that can be used for business process compliance are: ARIS Business Architect and ARIS Audit Manager. According to [Klueckmann, 2007], ARIS considers the business process design phase as the basis for business process compliance. The business process design phase is supported by the tool ARIS Business Architect. Here processes are designed with the process modeling methodology of event-driven process chains (EPCs). Using the method of EPC, the events in a company which lead to the initiation of certain functions, which in turn set off other events, can be visualized. The individual function can be related to the operational organizational units. The controls are part of process modeling in EPCs. Thus the controls in EPCs are not conceptually separated from the business process design phase and the modeling artifact in business process modeling. Therefore with this approach the reusability of process models and the controls disappears, since the controls are "hard wired" into the process models. After modeling the business process as an EPC in ARIS Business Architect, the identified risks in a business process can be designed into the EPC. This is also done with help of the tool ARIS Business Architect. The risks represent the check-points on controls previously designed in an EPC. The controls are then synchronized into the ARIS Audit Manager, where a testing process of the controls takes place. The testing process supported by ARIS Audit Manager represents a one-time testing of the controls by connecting to operative IT systems of an enterprise (ERP, CRM, SRM, etc), where the actual technical implementation of a business processes resides and is enacted (BP Execution phase). The test of controls is done by a separate test workflow modeled in the ARIS environment, which starts by automatically requesting assigned testing routines and ends with a sign-off by management and the preparation of test results for external audits. Once tests have been closed by the ARIS system or the user, they can no longer be changed. The test workflow is documented and locked. It is important to note that this is not a continuous monitoring of the business processes as required for business process compliance (Monitoring component in COSO, see section 3.1.2.2.5). The problem is that the control effectiveness can only be determined based on the state of business process instances. After successful testing of a control in ARIS Audit Manager, the tested controls are considered as effective. This approach represents a manual process. Since the real business processes run in different systems, not in ARIS products (external ERP system for instance), compliance in the running system is not guaranteed. For example, after successful testing of a control, the controls (which actually should be checked during business process executions) can be reset or changed by a technical or business process expert in the operative back end systems (the business process implementation). Thus the test results previously stored in ARIS Audit Manager do not necessarily represent the real effectiveness situation of the controls. The main weakness is based on the disconnection between the business process design phase and the controls design phase in ARIS Business Architect and on the fact that the real execution and effectiveness of the controls take place in different operative systems, outside the ARIS Tool set. The main building blocks of the business process compliance solution provided by ARIS are represented in Figure 64.

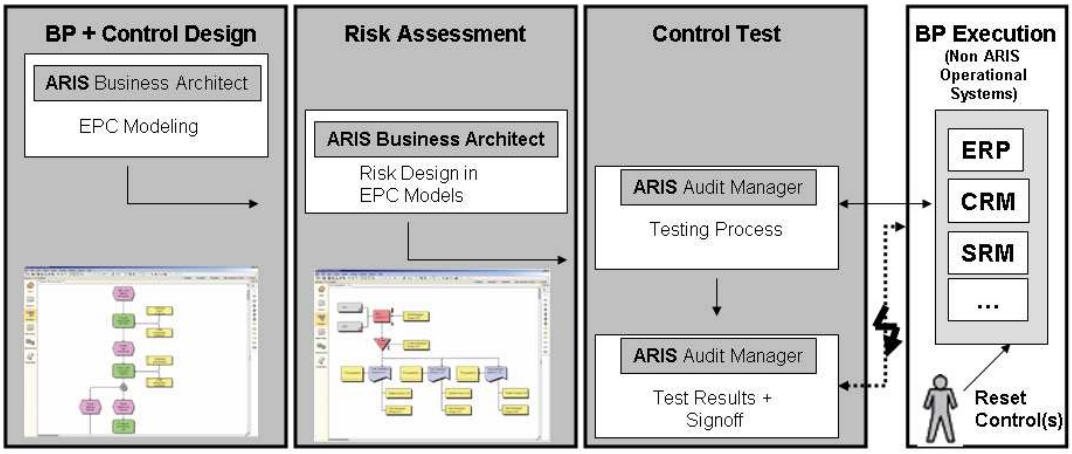

Figure 64 Interplay of ARIS tools for achieving business process compliance

8.5.2.3 Compliance solutions provided by big players in the software market

Big players in the software market also provide solutions for ensuring internal controls compliance. With reference to [Agrawal et al., 2006] the software products basically address the information and communication element of the COSO control framework. When it comes to design, and to assuring the effectiveness of the controls, they rely on the manual implementation of control activities and monitoring requirements set by COSO. [Agrawal et al., 2006] lists that, for example, IBM's Workplace for Business Controls and Microsoft's Solution Accelerator for Sarbanes-Oxley provide central content repositories with controlled access to company financial data. But these solutions represent a more or less manual assistance for compliance-responsible persons in an organization to document the risk assessments and control policies. Further, with the help of these software products the control responsibility can be assigned (delegated) to employees in the enterprise and those assigned employees are then responsible for implementation and monitoring of control effectiveness. Thus the task of assuring the controls effectiveness remains unspecified and manual, because the assigned control owners must manually verify whether each control has been implemented and assessors must likewise indicate whether each control has been effective.

A higher level of automation in business process compliance is provided by Oracle's Internal Controls Manager, which offers conventional workflow modeling capabilities. Virsa, recently acquired by SAP, provides through its product Continuous Compliance suite, some concrete controls in the area of security and access controls to IT systems. However, [Agrawal et al., 2006] comes to the conclusion that despite the existence of a wide variety of professional software solutions on the market, "a considerable opportunity exists to develop new technologies that further automate the most labor-intensive internal control processes". They basically argue that the opportunity is related to the low degree of automation in business process compliance. A sound conceptual separation of business processes and the internal controls process on the design level serves as a basis for bringing a higher level of adaptability, reusability, and usability to the models, and these needs are not actually being addressed by industrial solutions in this area.

8.5.3 Application of Formal Ontologies for Business Process Compliance Automation

According to [Studer et al., 1998] "an ontology is a formal, explicit specification of a shared conceptualization. A conceptualization refers to an abstract model of some phenomenon in the

world by having identified the relevant concepts of that phenomenon. Explicit means that the type of concepts used, and the constraints on their use are explicitly defined. [...] 'Formal' refers to the fact that the ontology should be machine readable, which excludes natural language. Shared reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group."

Based on formal ontologies the mechanism of *logical reasoning* can be applied. With logical reasoning the following tasks can be achieved:

- *Consistency checks* of a model
- *Subsumption reasoning*, which determines hierarchies of concepts (or their instances) existing in the ontology
- *Rule* processing

Regulation is one of the domains which apply extensive requirements on the formal modeling of the domain due to the complex normative knowledge existing in and between different related regulations. Thus one could claim that business process compliance could greatly benefit from an ontology driven information system based on the formalization of text contained and referenced in and between regulations [Gangemi et al., 2005].

Further, another thread of research exists in the development of domain specific ontologies in the area of financial accounting and costing. One of the main motivations of regulations such as SOX and requiring public companies to document and implement internal controls is to assure the control objective "financial statements". Thus using a clear, precise specification on financial transactions and accounting practices in terms of an ontology could provide support to companies in order to help them achieve this control objective, including the "operations" control objective. In this context, there exists a set of business domain ontologies such as TOVE [Fox, 1992], REA Accounting Ontology [Geerts et al., 1999], Business Model Ontology [Osterwalder, 2004], Enterprise Ontology [Ushold et al., 1998], or E3 Value Ontology [Gordijn et al., 2001]. The developers of the business domain ontologies listed above have different scientific backgrounds, which is reflected by the domain-specific knowledge captured in those ontologies and can also be noted through the level of formalization provided. While the creators of TOVE and of the Enterprise Ontology come from the artificial intelligence community, with a precise understanding of the ontology engineering process and the formalization there, the other ontologies are more focused on the business level. The E3-value ontology is not formalized at all and according to [Gordijn, 2002], a formalization of the ontology is not required because of the communication focus of the ontology.

The assumption is that modeling the enterprise, that is modeling its business processes according to the (formal) model proposed in a "suitable" ontology, would have benefits. In [Spyns et al., 2002] the possible benefits of formalizing the business domain ontologies and their application are recognized and discussed. Additionally, having the compliance requirements (lets say SOX) for a business company formalized according to the concepts and properties provided by a "suitable" legal ontology could be used to automate achieving business process compliance supported by logical reasoning possibility that can be utilized on top of formalized ontologies. The option of using such a method is can be analyzed in the following manner:

For the discussion on the applicability of ontologies in business process compliance we leave out the point of the expressivity of the formalism used, because it is off-topic for the discussion. We assume for the discussion an ideal constellation where we have a highly expressive language for describing the ontologies, so that we can very precisely and formally express and capture the content (text) of a regulation (lets say e.g. SOX 404). Let's further assume that we also have an ideal formalized business domain ontology, which reflects all necessary aspects and layers in an enterprise including its business processes and the way they are designed. Based on that ontology

we could capture the execution semantics of a business process instance. So far we have two different ontologies, one for modeling the current regulation (let's call it *RegOnto* here) and another one for modeling the business processes in companies (we call it *BPOnto* here). The question would be how and in which constellation we could use these two ontologies to better support business process compliance.

A possible method could be the following, as presented in Figure 65:

1. Develop, based on *RegOnto*, a formalized model of the currently considered regulations.
2. Design your enterprise, including its business processes, according to *BPOnto*.
3. Build a repository in which semantically enriched execution facts, according to the *BPOnto* ontology, are collected.
4. Use logical reasoning to determine whether the semantic instances of a business process instance as prepared during step three "satisfy" the *RegOnto* representation of the current regulation.

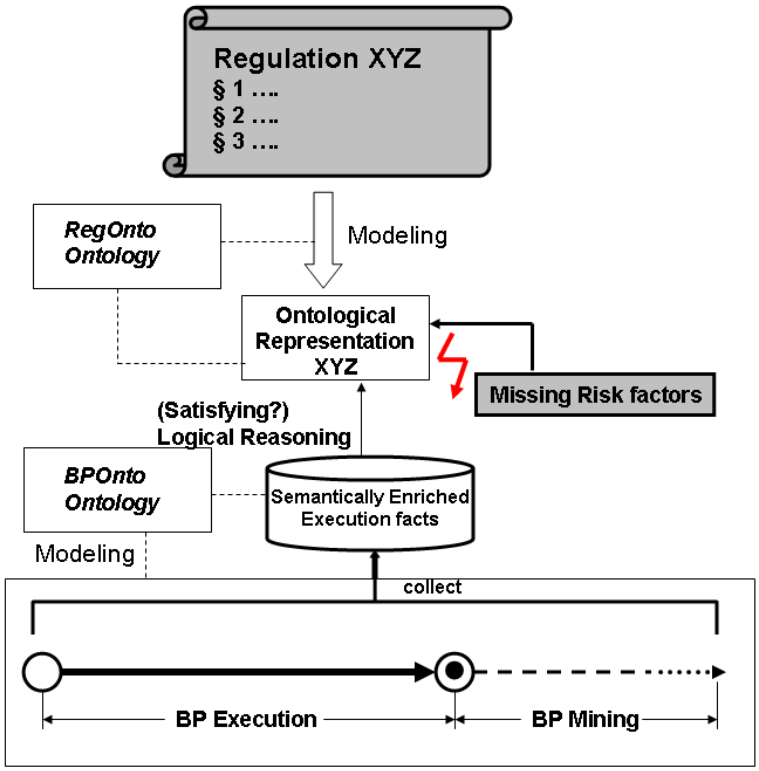

Figure 65 Possible role of legal and business ontologies in business process compliance

It is important to remark that although in our assumed world, a 1:1 mapping of a regulation including all the references to other related regulations in the case of business process compliance in an ontology would be theoretically possible, the resulting ontology is not as straightforwardly usable as described in the above approach. The reasons are as follows:

Recall the internal controls process as described by COSO (and required by SOX 404). The descriptions and the requirements in those regulations and frameworks are not "grounded" on a business level. By not grounded, we mean that it does not tell a company how exactly to ensure compliance, it does not specify which accounts, which business processes, and which controls have to be included in the internal controls project. This is due to the fact that each enterprise is unique in the way it works internally (internal factors) and how it is influenced by the external factors (such as market situation, politics, competitors etc.). We exemplified such a situation in

chapter 2, where each use-case company required a completely different set of controls for same business process. The crucial differentiating factor for each business is the notion of **risk.** The regulation does not tell an enterprise what constitutes a risk for that enterprise, and what therefore has to be addressed by its internal controls system. The enterprise itself has to assess the risk, possibly by consulting domain experts in the area of enterprise risk management (ERM). This is what we call the *interpretation of the regulation*, i.e. what it means for a company. Interpretation means to assess the risks and derive necessary consequences for the enterprise in terms of controls. The two variable factors are risks and their consequences: We mention again that risk is enterprise-specific, which means that the occurrence of a certain situation may represent a risk for an enterprise, while the occurrence of the same situational constellation may not be a risk for another enterprise operating in a different environment (internal and external factors). At the same time, the consequence of a risk is enterprise specific as well in terms of how to handle a certain risk. Thus the regulations in the area of enterprise risk management require enterprise-specific interpretation. The modeling of the regulation text, as it is given in an ontology, (see Figure 65), is not sufficient for business process compliance, because the "interpretation"-step is necessary. The reflection of the interpretation of the regulation will lead to a partially enterprise-specific ontological model of the same regulation. This is due to the fact that each business domain (transportation vs. high tech companies for example) use their own terms, have their own business processes and have their own risk factors, which leads to partially different conceptual models being present in an ontology for each enterprise (see Figure 66).

To the best of our knowledge there is currently no research that addresses the problem of interpretation of regulation and how enterprise-specific risks may be related to the ontological modeling of regulations. According to [Spyns et al., 2002] and [Ushold et al., 1996], important ontological quality factors are: reusability, reliability, shareability, portability, and interoperability. We see in the context of business process compliance the factors of reusability and shareability of ontologies used in the approach described above as not fully satisfied. The reason is that in each enterprise, significant conceptual modeling effort has to be made on top of a given regulation ontology in order to integrate the relevant concepts of risk for that enterprise in that ontology.

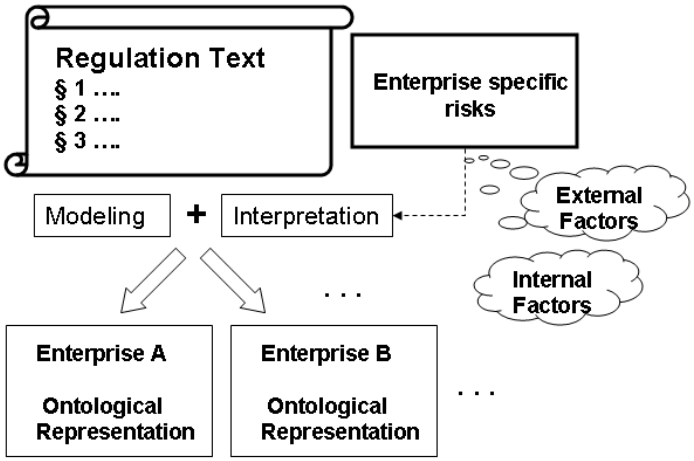

Figure 66 Different set of required concepts for identical regulations resulting in two different ontologies

8.6 Conclusion

This chapter presented the description and the implementation of an approach for detecting control violations during business process executions. Such an approach enables us to realize business process compliance in a preventive manner. The system realizing the approach, called ICR, consists of 2 parts: ICR-Design and ICR-Execution. ICR-Design contains a set of controls and ICR-Execution is responsible for ensuring that the business process instances which violate controls in ICR-Design will not pass. ICR builds on models that were provided in chapters 4, 6 and 7.

In ICR-Design, controls according to Definition 6.11 are designed on business process models according to Definition 4.2, possibility through the instantiation of a control pattern according to Definition 7.4. The controls will be deployed by their storage in ICR-Design and by adapting the business process model according to recovery actions in the control. This adaptation step will compile a set of new transitions in the business process model. These transitions will ensure that a control-violating business process instance behave according to the required recovery actions of the control. The integration of a business process instance with a control is then achieved by the continuous monitoring of a business process instance by ICR-Execution. This is achieved by evaluating the current state of the business process instance (contained in business process context, see section 4.2.2.2) against the control conditions of a control specifying in what circumstances its triggering event can be considered as having occurred.

9 Assessment

The contributions in this thesis are assessed according to the following two aspects:

1. *Complexity* of the system proposed and its
2. *Completeness*

In the following we introduce the role of *complexity* and *completeness* assessment in the context of this thesis:

In software engineering, several definitions have been given to describe the meaning of complexity. [Fenton, 1991] defines complexity as the amount of resources required for a problem's solution. [Curtis, 1980] states that complexity is a characteristic of the software interface which influences the resources another system will expend or commit while interacting with the software. [Card et al., 1988] define relative system maintenance complexity as the sum of structural complexity and data complexity divided by the number of modules changed. For business process management, [Cardoso, 2006a], defines process model complexity as "The degree to which a process is difficult to analyze, understand or explain. It may be characterized by the number and intricacy of activity interfaces, transitions, conditional and parallel branches, the existence of loops, data-flow, control flow, roles, activity categories, the types of data structures, and other process characteristics." Applied to our context complexity, is related to the following aspects:

1. Design complexity and
2. Execution complexity.

Design complexity affects the following aspect: How complex is it to provide a model of business process compliance)? In order to answer this question we divide the system realizing business process compliance into the following sub-systems:

i) System of business process models, as they have to be represented according to the definition of BPD

ii) Controls according to the Definition 6.11; controls will be designed separately from business processes.

The key question asks how much effort would be required to design such a system. The assessment of the design complexity of business processes will be discussed in section 9.1.1.1 and the complexity of control modeling will be discussed in 9.1.1.2.

Execution complexity is in our approach related to the execution phase of business processes, as described in chapter 8. More specifically, execution complexity is about monitoring business process instances and reacting to control violations. The key question in this context is: "Is it feasible, from a performance point-of-view, to assure the control effectiveness through the integration of the component ICR-Execution during business process execution?" We assess the performance of the system according to the additional *time complexity* brought into the system through ICR-Execution. The existence of such a component is not allowed to influence the runtime performance of the system, i.e. the time taken to fulfill a certain part of a business process, in such a way that an end user of the system, i.e. a business user, is hindered in the pursuit of the business objective intended by the business process. Space complexity of execution is not subject to our assessment because there are usually *large scale enterprise systems* responsible for implementing the business processes we are dealing with. Thus providing additional hardware plays, from our point of view, a minor role in assessing the feasibility of approaches today. To summarize, we consider the time complexity of business process execution as the crucial factor for assuring the feasibility of the ICR-Execution. The optimal way to evaluate the time complexity of the approach is through its technical deployment and

performance measurements in real production environments. This was not possible for this thesis, due to mostly political reasons: in order to make it this approach possible, a business or company must first provide a critical set of controls resulting from a risk assessment for that enterprise. By providing such information, a business company would indirectly expose the risks with which the business enterprise is faced. Companies are not usually willing to provide such information, not even to their internal employees. Thus we were forced to take the approach of *estimating* the time complexity of the system. It is in this way that we show that the implementation of our approach will have no negative influence on the efficiency of a system for managing the business processes. Technical performance will not be affected. The time complexity aspect will be discussed in section 9.1.2.

There exist several definitions for *completeness*. We consider completeness from an expressiveness point of view in general and refer to the definition provided by Wikipedia for language completeness [WikiCompleteness]: "A language is *expressively complete* if it can express the subject matter for which it is intended." Applying this definition to our context, the "*language"* would be the model of control (Definition 6.11) and its different patterns presented in chapter 7. The term *"subject matter"* in the definition is the possible set of internal controls in a company. Whether and how far is it possible to reflect different kinds of controls in the control model proposed should be assessed. We assume that a control that can be captured by the model can be used in the compliance validation during the execution as described in chapter 8. The key question to be answered by the completeness assessment is whether, and how well, building on top of the proposed models is able to provide the necessary set of controls according to the models that are the basis for the automation of business process compliance design. Automation in this context is related to the question of whether, and how much, human interaction will still be required for defining the controls on top of the models. This completeness assessment is covered in section 9.2.

9.1 Complexity

In the following we first discuss the design complexity of business process models and controls, and then provide the estimation model for the execution complexity of business process instances in a system which includes ICR-Execution.

9.1.1 Modeling Complexity

9.1.1.1 Complexity of Business Process Modeling

The effort required to model a business processdepends highly the complexity of the business process to be designed. To determine the complexity of a business process model, the BPM community relies mostly on research results coming from the software engineering community, where a significant amount of research has been done regarding the complexity of software programs.

[Cardoso, 2006b] extends the work of [van der Aalst et al., 2005c] by stating that the complexity of a business process model can be determined from four perspectives. These perspectives are:

- Activity complexity: This view on complexity simply calculates the number of activities in a process model. This metric was inspired by lines-of-code (LOC) metric used with a significant success rate in software engineering [Jones, 1986].

- Control-flow complexity**:** The control-flow perspective describes activities, and their ordering, through different constructors, which permit the flow of execution control. The control-flow complexity of a process is closely related to its activity complexityperspective. While the control-flow complexity can be very low, its activity complexity can be very high. For example, a process that has a thousand activities may have a very low control-flow complexity (if it is sequential), whereas its activity complexity is very high.
- Data-flow complexity: This perspective reflects the complexity of documents and other data objects that flow between activities. The data-flow complexity of a process increases with the complexity of its data structures, the number of formal parameters of activities, and the mappings between activities' data [Reijers et al., 2004].
- Resource complexity: The resource perspective provides an organizational structure anchor to the business process in the form of human and device roles responsible for executing activities [van der Aalst et al., 2005c].

The above complexity perspectives on a business process model are reflected in the proposed model of a business process in terms of BPD: the activity complexity is measured by the number of activities according to Definition 4.12 in a BPD. Control-flow and data-complexity are reflected by the transitions (see Definition 4.14) of business documents and activities in a BPD. Resource complexity is influenced by the number of users involved in a BPD and by their roles (see section 4.2.2.3) in that BPD. There are complexity metrics of business process models for data flow and control-flow complexity. They are listed in the following and can also be taken as reference metrics to measure the complexity of a BPD:

According to [Cardoso, 2005], a data-flow complexity metric can be composed of several sub-metrics, including: data complexity, interface complexity, and interface integration complexity. While the first two sub-metrics are related to static data aspects, the third metric is more dynamic in nature and focuses on data dependencies between the different activities of a process. In the case of our model of BPD, this means that if a business document is composed of basic data types, it will have lower complexity than one which is composed of business documents that are a composition of other business document types. Further [Gruhn et al., 2006] propose seven measurement metrics for the complexity of the control-flow in a business process model, which are: Number of Activities, Control Flow Complexity (CFC), Max. / Min. nesting depth, Number of handles, Cognitive weight, (Anti) Patterns for BPM, Fan-in / Fan out. For a detailed discussion of these measurement metrics, please refer to [Gruhn et al., 2006].

To the best of our knowledge there exists almost no research about the calculation of the effort needed to model a business process. However, we propose a modified calculation approach, borrowed from ontology engineering called ONTOCOM [Bontas et al., 2006]. Ther modified approach based on ONTOCOM can be used to estimate the cost of modeling a business process according to BPD. We believe that the creation of a business process model can be treated as an ontology engineering problem, given a constellation where a standard software provider offers different repositories of business documents, activities etc, on top of which a customer enterprise can build its business processes. In this case the different entities in these repositories and their relationships can be seen as an analogy to the concepts and the properties between them defined in the ontology (TBox). The business process model built on top of entities in such a repository then represents an instantiation of the ontology (which can be treated as the ABox).

Based on this argumentation we propose below the following adapted formula for calculating the effort (in person-months PM) required for modeling a business process:

$$PM = A^* | CES |^{\alpha} * \prod cd_i$$

,where

- The cardinality of controlled entities CES = USERS ∪ ROLES ∪ TRANSITIONS ∪ BDS (business documents) in a business process represents one of the modeling effort factors. The required controls in a business process are excluded here, because they will be separately designed.
- Parameter α is defined in the same way as in the ONTOCOM-model, in that it controls for the possibility of non-linear behavior of the model with respect to the number of controlled entities.
- *A* is a constant according to the ONTOCOM-model, which represents a baseline multiplicative calibration constant in person months
- *cd*s are different *cost drive*r factors having a rating level that expresses their impact on development effort.

The critical factor for calculating the PM in the context of business process compliance is the number of entities, which are affected by the controls, i.e. the size of the set CES. The cost drivers in the area of ontology engineering can be domain analysis complexity, implementation complexity, support tools etc. A detailed description of different cost drivers in the area of ontology engineering and their rating levels can be found in [Bontas et al., 2006]. From our point of view these cost drivers will mostly hold for business process modeling as well.

As a concluding statement regarding the modeling complexity, we argue that because our proposed model of BPD adds no additional complexity perspective to the complexity of a business process model, its modeling method and the resulting models would have the same complexity and require the same effort as other mature process modeling approaches such as EPCs, BPMN etc.

9.1.1.2 Complexity of Control Modeling

The task of identifying the necessary controls in order to mitigate the existing risks on business processes remains manual (see section 4.1.2). In the following, we discuss the complexity of modeling the controls for a business process. In order to achieve this task a compliance expert must be assisted by supporting the following functionalities:

1. Model a control
2. Apply the modeled control to a business process model.

The first functionality relates to the modeling of a control and is supported by providing a precise model of the control (see Definition 6.11). The management of controls, i.e. a desired modification of an existing control, is highly flexible. This is achieved through the parameterization of the control model that enables the modification of its different attributes. The factors influencing the complexity of a control are the number of its attributes and the complexity of each of its attributes. Through a strict model-driven design of controls, the maintenance of controls, which is closely related to the challenge of *maintaining compliance* as described in section 2.4.5, is greatly simplified.

The second functionality can be treated generally as an annotation, i.e. a business process model is annotated with the necessary controls. It can be treated as an annotation due to the fact that an annotation approach must be capable of supporting or helping in answering the questions:

‘what to annotate”, “where to annotate” and “how to annotate”, and can be applied to our case in following way:

1. *What* to annotate in a business process with the control: The question of what to annotate in a business process is answered by a compliance expert. The approach supports him in the selection of an identified controlled entity in a business process and helps him to equip it with a control.
2. *Where* in a business process to annotate the control: This is the scope of the control codified in its *event*- part as defined in Definition 6.11.
3. *How* to annotate the control: The technical annotation of a business process is then automated as described by the business process model adaptation in 8.3.2. Through the concept of business process model adaptation, an existing business process model designed according to BPD is extended by the necessary artifacts required in order to monitor a business process instance and react any control violation. This is an automated approach and requires no manual interaction in order to adapt the process model. All that is required is to identify the business process, specify the control for it and set the recovery actions (see section 6.2.3) that should be invoked in case that control is violated.

Providing a quantitative number regarding the modeling and annotation effort required for controls in business processes is heavily dependent on the level of technical and compliance expertise of the person using and applying the models. In order to determine an approximate value of effort for annotating the business processes with controls, the evaluation results of [Handschuh, 2005] can be used as an orientation, although the nature of annotation problem in [Handschuh, 2005] is different from the case of business process compliance. [Handschuh, 2005] determined the effort required for the semantic annotation of web sites with semantic information. The results achieved therein did show that, in that specific context, the effort for the manual annotation of web sites with semantic information is almost within reasonable or feasible limits. In that work, evaluation was in a context where experiments were carried out by students without deep technical and logical knowledge, using the annotation approach. We argue that these results can be applied to designing controls in business processes, because the approach can be treated as an annotation. We further believe that the efforts required in our context would be even lesser, for the following reasons:

i) Through usage of configurable patterns (see pattern specific parameters in section 7.4), the required level of technical knowledge is significantly lowered
ii) The level of required knowledge of logic is minimal because of the use of business level control patterns, which hide the technical complexity of the underlying formalism
iii) The number of concepts and the relationships between them in the domain model of business process compliance is relatively limited
iv) The set of possible annotations in a business process is limited through the extensible set of proposed control patterns
v) The target users of our approach are experts in their domain, namely compliance experts in a company, whereas in the context of semantic annotation of web sites the set of target users is potentially open.

9.1.2 Execution Complexity

Regarding the integration of ICR-Execution into the execution of a business process, in the following we discuss and validate the performance of a system with regard to its real world feasibility. The integration of ICR-Execution implies that during the execution of a business

process, additional steps have to be enacted inside a process part, in the scope of a control. The full set of possible additional required steps is visualized in Figure 67, which yields worst-case time complexity results during execution:

1. Synchronization
    a. Collect data
    b. Obtain reference
    c. Write
2. rule processing
    a. working memory update
    b. query backend systems
    c. RETE
    d. Control Violation Fact Creation
3. Recovery Action Handling
    a. instantiate rbp
    b. Business process instance recovery

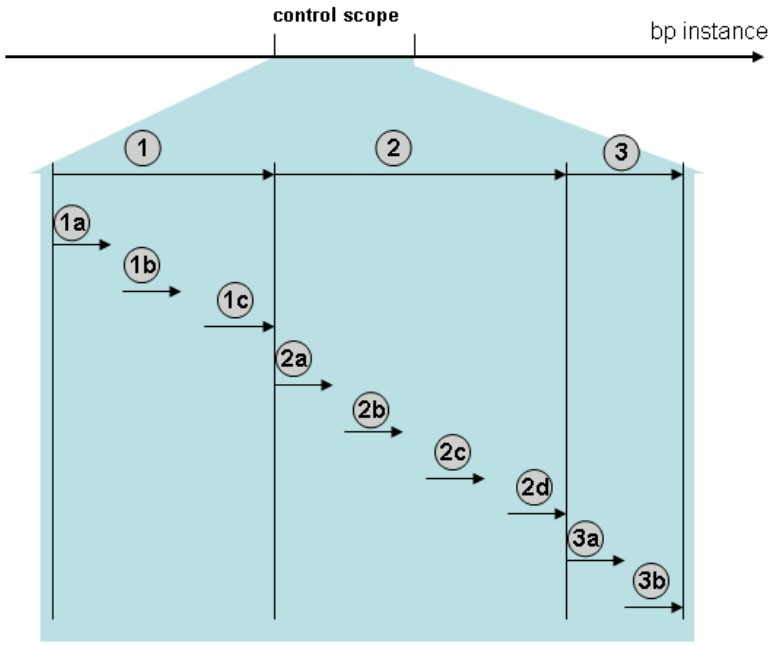

Figure 67 Full set of possible additional steps required in compliance validation

These additional steps only exist to assure the compliance of business processes in a preventive manner. They would not exist if the controls would have been handled in a detective manner, which is the usual state of practice. In addition, these steps are not necessary for achieving the business objectives of a process. These additional steps exist for assuring the control objective of a business process and thus are not allowed to influence the execution of business processes in such a manner that from a technical point of view the achievement of a business objective of a process will be negatively influenced. The technical factor that has to be taken into account is the additional time complexity of the system added to the system through ICR-Execution.

9.1.2.1 Effort estimation

Step 1a (Collect Data) is about replicating the operational data, which is produced during business process execution and which is necessary for control evaluation. This is copied from the business process instance and context into a Map-Data type, which will be asserted to ICR-Execution as facts. Since this operation is basically a copy-operation without any computational effort we estimate its time complexity as linear O(n) depending on the number *n* of data items which are copied.

Step 1b (Obtain reference) is about establishing a connection to the working memory of the rule engine. The connection will be cached in the context of the business process instance. We estimate the time complexity of this operation as a constant O(c) = 1 in the best and worst case. Best case would be if the reference is cached in context, worst case would be if the reference has to be obtained by connecting to the working memory.

Step 1c (Write) is responsible for sending the set of collected data items copied during step 1a to ICR, i.e. its working memory. Although the real time efforts for this operation depend on several factors, such as network connectivity and the distribution model of the engines (rule and BPM), we estimate the time complexity of this step as linear O(m) depending on size m of data sent to ICR-Execution.

Step 2a (Update Working Memory) updates the existing facts in the working memory and creates new ones. The time cost for this operation depends on the current size of the working memory, i.e. the number l of facts in it and the number k of received facts from business process instance. The time complexity of this operation is O(l) + O(k) linear.

Step 2b (Query Backend Systems) The time complexity of this step depends on the form and number of the rules representing the controls, whether they require data from operational backend systems in order to evaluate a rule. The processing of such backend queries depends on the technical environment parameters, i.e. the performance of the database of the backend systems responding to such queries. We assume that in most cases in practice the database of such backend systems is relational. Performance of processing queries in relational backend systems depends on technical factors such as the usage of inidices in the database, form of the queries (Selection, Projection type of joins etc.), the preferment design of database schema and its normalization, size of operational data to be retrieved or whether a distributed query processing is possible etc. Assuming j as the cardinality of the relation in database, it is well known [Özsu et al., 1999] that the complexity of relational operation containing a simple select without using joins in the selection is linear O(j), a join-operation has the complexity of O(j * log j) and the worst case would be a query with a cartesian product operation that has a complexity of $O(j^2)$. However, in practice, implementations of the state of the art relational databases offer a very good performance and query optimization techniques such that we expect a linear time complexity as well in practice for query processing. We further argue that this kind of query operations would be invoked during business process execution anyways, in order to fulfill the business objectives of a business process. In the case of the application of a detective nature of controls, the query of the backend systems in order to evaluate the controls would be done manually, by a compliance expert.

Step 2c (RETE) This step represents the most critical phase regarding the time complexity. Beside the technical environment, i.e. the number and size of CPUs used for running the rule engine infrastructure, the processing of this step depends on 2 factors: i) number of rules and ii) the form of their LHS. According to [Forgy 1979] and [Albert, 2006], RETE requires in worst case a linear time complexity in order to compute the set of satisfied rules. Further, different production rule engines have their custom implementations of RETE algorithm, which optimizes

the pattern matching algorithm. Figure 68 refers to a benchmark provided by ILog Rule engine regarding the time needed for executing the rules. We do not expect that the number of rules (i.e. the controls in a company) in a real application scenario would exceed 4000. As shown in Figure 68, the time effort for processing such a rule base remains almost linear.

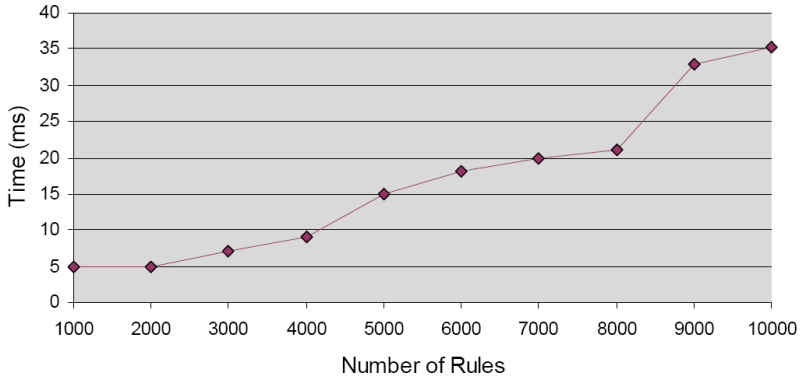

Figure 68 Performance of Ilog Rule Execution [JRULES]

Step 2d (Control Violation fact creation) In the case of a control violation, during this step a fact will be generated in the working memory signaling the control violation. It is obvious that the time complexity of this operation is constant.

Step 3a (Instantiate rbp) During this step an instance of the business process responsible for eliminating the conditions which make a control violated is generated. Here in parallel necessary data for processing the instance will also be provided. While we estimate the time complexity of the instantiation itself as constant, the time complexity of providing data to the instance depends on the number p of data items set in the instance. Thus we consider the time complexity of this step as $O(1) + O(p) = O(p)$.

Step 3b (Business Process Instance Recovery) This step is related to the application of all other types of the selected recovery action model during control design phase (as described in section 8.3.3). The time complexity of this step depends on the complexity of the process definition that has to be recovered. The most time-consuming recovery action would be a *rollback* and we estimate its complexity as linear, depending on the number of transitions existing in a process definition.

The whole complexity brought to the system by the above steps will be discussed in the following sub-section

9.1.2.2 Discussion

None of the steps oulined above requires a time complexity worst than linear. Thus we consider that the integration of ICR-Execution during process of business process instance adds a linear time complexity to the system, where the most expensive operation is the rule execution in the rule engine by RETE algorithm (Step 2c) depending on a number j of rules. This is the general observation in several applications using a production rule engine. However it is important to assess the additional complexity brought into the system while considering the nature of operative business processes, which are the focus of business process compliance. Typical operative business processes such as sales processing, purchasing or human resource management require several days to be processed. The typical paradigm is that certain roles or users involved in a business process receive tasks in their task lists, which they then must process. This is very similar to e-mail processing. Here a business user does not immediately

process a task (as he would not immediately process an e-mail received), but rather checks his task list and processes the tasks in FIFO or priority based manner. For these reasons, the additional time required for processing the controls in ICR-Execution can be ignored and does not significantly influence the fulfillment of the business objectives of a business process in the context of how they are processed by business users.

9.2 Completeness Assessment of Control Model

The completeness aspect will be assessed empirically. We assess the completeness of the control model by reporting our analysis of a large set of controls and the way we could capture them in our model according to our approach.

COSO provides in [COSO92] an evaluation framework designed to assist an evaluator in completing the "Risk Assessment and Control Activities" in a company. The controls covered in the COSO's Reference Manual are based on a generic model of a business enterprise. The generic business model depicts major activities in an enterprise in terms of its business processes, and is organized in levels, from a "high level" view of an enterprise to increasingly more detailed "low level" views.

The complete set of controls proposed within this framework contains 504 controls. We used this framework to assess how far we are able to reflect the required set of controls in an enterprise based on our model. The result of our assessment is shown in Table 10. The table is organized in the following way: The first row shows the name of the process in the enterprise, the second row the total number of controls proposed for that process by COSO, the third row named "Possible" shows the number (and percentage) of controls, which we are able to design and check according to our model and then to approach in a preventive way, the number of controls in the fourth row (Interpretation) require further human interpretation to be reflected in the model, thus they are not immediately applicable in the approach. The fifth row shows how many controls we were unable to capture and check.

Table 10 Assessment results for possibility of mapping the COSO's control set on the control model

Process area	**Total No. of Controls**	**Possible**	**Interpretation**	**Not Possible**
Inbound Activities	39	27 (~69%)	5 (~12%)	7 (~18%)
Operations	33	18 (~55,5%)	3 (~9%)	12 (~36%)
Outbound Activities	34	24 (~70,5%)	3 (~9%)	7 (~20,5%)
Marketing and Sales	29	14 (~48%)	2 (~7%)	13 (~49%)
Service	19	10 (~53%)	5 (~26%)	4 (~21%)

Human Resources	33	14 (~42%)	8 (~24%)	11 (~33%)
Technology Development	12	6 (~50%)	2 (~17%)	4 (~33%)
Procurement	40	37 (~92,5%)	1 (~2,5%)	2 (~5%)
Process Accounts Payable	18	18 (100%)	0 (0%)	0 (0%)
Process Accounts Receivable	17	16 (~94%)	1 (~6%)	0 (0%)
Process Funds	46	38 (~83%)	2 (~4%)	6 (~13%)
Process Fixed Assets	13	8 (~61,5%)	2 (~15,5%)	3 (~23%)
Analyze and Reconcile	3	3 (100%)	0 (0%)	0 (0%)
Process Benefits and Retiree	20	12 (~60%)	1 (~5%)	7 (~35%)
Process Payroll	22	14 (~64%)	2 (~9%)	6 (~27%)
Process Tax	10	5 (50%)	0 (0%)	5 (50%)
Process product costs	15	12 (80%)	3 (~20%)	0 (0%)
Process Financial Management and Reporting	11	5 (~45%)	3 (~27%)	3 (~27%)
Manage the enterprise	15	4 (~27%)	3 (20%)	8 (~53%)
Manage external Relations	6	0 (0%)	0 (0%)	6 (100%)
Manage Administrative Services	3	0 (0%)	0 (0%)	3 (100%)
Manage IT	24	15 (62,5%)	4 (~17%)	5 (~21%)
Manage Risks	15	1 (~7%)	3 (~20%)	11 (~73%)
Manage Legal Affairs	13	5 (~38%)	3 (~23%)	5 (~38%)
Planning	14	4 (~28,5%)	6 (~43%)	4 (~28,5%)
All (25)	**504**	**~ 55 %**	**~ 12,5 %**	**32,5 %**

Some examples of controls in the table above which are in the row named “Interpretation” are:

- “Monitor *production problems* related to unavailable materials and parts” in the process area “Inbound Activities”
- “Evaluate *adequacy* of production capacity” in the process area “Operations”

- "Monitor *adequacy* of staffing, their overtime and workloads" in the process area "Service" etc.

The problem with the above control descriptions is that the model does not know the notion of "problem" or "adequacy". In order to be reflected in our control model, a compliance expert has to describe what a "problem" or "adequate" means for a specific enterprise in quantitative terms. This is the reason why such controls have to be "interpreted" before they can be designed in a business process.

Some examples of controls in Table 10 in the "Not Possible" row are:

- "Institute and monitor code of conduct" or "Maintain physical security of purchase orders" in the process area "Procurement"
- "Personnel report suspected violations of laws, regulations or company policies" or "Human resource personnel are subject to periodic training regarding legal and regulatory requirements in the process area "Human Resources"
- "Periodically evaluate direction and priorities set by senior management to make certain they are still valid" in the process area "Manage the enterprise" etc.

The above control descriptions cannot be captured by our model and consequently cannot be automatically monitored, because they involve employee behavior and do not interact with IT systems. These controls can only be verified manually and are closed to COSO component "Control environment" as described in section 3.1.2.2.1.

It is interesting to remark that some process areas such as "Processing Accounts payable", "Processing Accounts receivable" or "Procurement" seem to be very well suited to use in our approach (high percentage of possible controls). These are according to COSO "low level" areas, which are in closed interactions with IT systems. At the same time, we count in "high-level" process areas such as "Manage the enterprise", "Manage external relationships" or "Manage Administrative Services" very few controls which are well-suited to our approach.

However, the results of the completeness assessment show that more than half of the controls (55%) can be represented by our model and their effectiveness can be assured automatically in a preventive way by our proposed approach. We are satisfied with this number, it shows that using a model-driven approach will significantly reduce the manual efforts required for compliance management of internal controls for business processes.

The fact that a significant number of the controls require a further interpretation to be used in our approach (12,5 %) and even that 32,5% of the controls are not automatable at all shows that the human factor still plays a very important role in business process compliance. Business process compliance and the required internal controls cannot be completely automated, we conclude that the roles of compliance experts and control testers will not be rendered obsolete.

9.3 Summary

In this chapter we assessed the added value of using models to design and manage controls for business process compliance. To this purpose we discussed the modeling complexity of business processes and controls in a step-by-step manner.

We also showed that the integration of ICR-Execution will not negatively influence the transactional response time of business processes (Execution complexity). Negative influence is interpreted in this context as resulting in a situation in which the business objectives of a business process can not be satisfied due to low performance of the whole system. We showed that this was not the case by estimating the time complexity of the additional steps required during execution of a business process in order to realize the approach implemented by ICR-Execution,

which according to our estimation seems to be linear. However we concluded that this additional time will not be of any consequence due the nature of operative business processes, which often take several days to be processed by business users.

The completeness of the control model and its patterns were assessed in a scenario-driven fashion using the proposed controls on 25 different process areas proposed by COSO evaluation framework.

10 Conclusion and Outlook

In this thesis, we tackled the problem of high costs and effort for achieving the compliance of business processes to regulations in the area of Enterprise Risk Management (ERM). Common to these regulations (such as the Sarbanes Oxley Act – SOX) are requirements on the presence of effective internal controls at companies. The current shortcomings faced by companies in this respect are the low level of automation in the translation of compliance requirements into a set of internal controls and assuring the effectiveness of these controls during the execution of business processes. The high cost of business process compliance is due to the fact that in many organizations a large number of the steps in designing and testing controls on business processes are manual.

In section 10.1, we briefly summarize the contents of this work and accentuate its main contributions. Subsequently, an outlook on possible future work is addressed by discussing some open research questions in section 10.2.

10.1 Summary of Contributions

Most companies rely on standard software providers to deliver software solutions on top of which they build the companies' business processes. These companies require this standard software to provide mechanisms which yield a higher level of adaptability, reusability and usability of internal controls on their business processes. The aspects adaptability, reusability and usability of internal controls are related to the design time of business process compliance, namely, modeling the controls in the business processes. Another requirement from customer companies on their standard software providers is to assure the automatic detection of possible control violations or to prevent possible non-compliant executions of business processes. Meeting this requirement depends on having an approach for continuous monitoring of compliance at companies. An automated approach for monitoring business process compliance provides companies with a means to transform manual steps and automate them as system level controls. Automation of controls and monitoring the compliance of business process executions to them saves those costs associated with performing the controls and improves the reliability of the controls because the level of human interaction required for assuring their reliability is minimized.

The basis for the contributions of this thesis to reducing the high effort of modeling compliance and assuring compliant executions of business processes is a strict model-driven approach to business process compliance. Using models to describe the necessary artifacts involved in internal controls enables the realization of preventive compliance for business processes. This is achieved by increasing the number of preventive controls that can be automated. By comparison, the usual manually detected controls can only assure post-compliance.

The problem space described above was exemplified in this thesis by elicitation of a set of challenges identified through two use cases. These challenges can be summarized as follows:

Each company has different sets of significant accounts, affected by different relevant business processes containing different kinds of risks that are the focus of internal controls. Risk assessment of business processes is enterprise-specific. This leads to the situation that each company building on top of standard software requires its own enterprise-specific variant of a

certain type of business process (such as sales, purchasing etc.) that has to be provided and supported by standard software providers. Different roles in a company are involved in the design of business processes and the necessary set of controls on them which mitigate the exiting risks in the business process: business process and compliance experts. They have different background knowledge and expertise and different intentions regarding a business process. A business process expert is interested in a business process that achieves the business objectives for which it exists, whereas a compliance expert is interested to assure the compliant behavior of a business process by assuring the effectiveness of controls required for a business process. This leads to the situation that one meets not only a heterogeneous system environment necessary for achieving business process compliance, but also a heterogeneity in the roles involved and their responsibilities in business process compliance. After the necessary set of controls on a relevant business processes are determined, their design has today a manual nature in that they are basically only documented. The lack of precise models of controls, business processes and the existing formal relationships between them hinders a technical link between the documented controls and business process designs and, accordingly, their execution. This missing link leads to the situation that assuring the effectiveness of the controls is mostly manual in nature, using a test-driven approach by a control tester in a company. Furthermore, once a control has been tested, and has been judged to be working properly, it cannot be assumed to work properly in the future, meaning it may become non-effective. The reason is that new business processes, respectively new software versions implementing business processes are continuously deployed in companies, and may affect the effectiveness of controls. Therefore, a continuous monitoring approach built on top of a formal domain model of controls and business processes assures preventive detection of non-compliant behavior of business process executions, minimizing the manual effort that is today required in maintaining compliance.

In order to overcome the above challenges this thesis developed an abstraction layer above business processes, which is responsible for business process compliance. In this layer the controls are formally modeled and evaluated against existing process models and their execution instances. The thesis describes a novel, model-driven approach for the automation of business process compliance through monitoring the effectiveness of controls. This is enabled through the conceptual separation of the design of controls and business processes at a model-level, and a tight integration of controls in the business process instances at the execution-level. In order to address the usability of the models and the approach, this thesis advocated the use of control patterns in the abstraction layer responsible for business process compliance. The control patterns should give compliance experts and business process experts access to specify and design the compliance requirements accordingly. These control patterns are then mapped to formal models that are used by technical experts to implement the control patterns in business processes. The provided model-driven approach in the context of business process compliance has the following added value:

- It enables the usage of formal methods, like inference, for the verification and validation of a business process' compliance to internal controls as required in regulations such as SOX.
- Consequently, compliance will be achieved automatically, based on the current state of parameters (instances) of a business process.
- Moreover, the conceptual description of control conditions ensures the flexibility of the approach, i.e. changes to the controls require no manual changes in the design and execution of the original business processes; this ensures relatively effortless maintenance of compliance.

- Finally, through another abstraction layer introduced on top of the compliance definitions, we ensure that non-technical experts can build the required internal controls on top of the domain model provided.

To complement this abstraction layer which uses models of the entities involved in business process compliance, a verification and validation approach was presented: The verification of business process models assures that business processes are built in a compliant manner as required in a formal specification (chapter 5). The validation assures the compliant behavior of business process executions, i.e. the business processes work as described in the formal model of controls (chapter 8).

The requirements for realizing this approach to business process compliance based on verification and validation were the following:

i) Model of a control and its relationship to business processes
ii) An approach for separating the design of controls and business processes
iii) An approach for deploying independently designed controls on business process models
iv) Monitoring of control effectiveness during business process executions
v) A mechanism for handling possible control violations
vi) A cooperative environment for compliance and business process experts to design and manage controls
vii) A common domain terminology in which the involved roles communicate.

The basis for satisfying the above requirements was provided in chapter 4 through a precise formalized description of the entities in business process compliance that are the targets of internal controls (Controlled Entities). The precise model of a control, which is a controlled entity as well, was formally described in chapter 6 in detail. Verification and validation of business process compliance builds on top of this formalized domain model of business process compliance. Furthermore, the pattern-based approach to designing the controls in business processes presented in chapter 7 uses the domain model of business process compliance and the compliance controls provided in chapters 4 and 6.

In addition to the fulfillment of the requirements listed above, the following major contributions to the research questions addressed in this thesis can be identified:

- **Models of Intersection between Business Processes and Internal Controls**

  This contribution is related to the question about the *relationship between business processes and internal controls* that was raised by the first research question. This thesis provides a set of modeling entities for business process design that are subjected to internal controls (Controlled Entities). It was shown what the formal relationships between these controlled entities and a business process at design and execution time are. These entities and their relationships serve as the basis for modeling the controls in business processes and assuring their effectiveness during business process executions. Furthermore, using the models proposed we showed that the level of automation in the design and application of internal controls to business processes can be raised, which partly addresses the second research question "*Can internal controls be automated using a model-driven approach?*"

- **Identification and Application of Controls Patterns to Business Processes**

  With this contribution the third research question about the *usability of a model-driven approach for compliance experts* is tackled. This thesis presents a set of control patterns as the terminology in which the compliance experts speak about the internal controls compliance domain. The control patterns are formalized and their relationships to the models presented in the first contribution are specified. Designing the controls in business processes with the

suggested pattern-based approach reduces the complexity of using the models in this thesis. This improves the usability of the approach.

- **Preventive nature of Business Process Compliance in daily operations**

  While the purpose of the first contribution was to address the second research question concerning the *automation of internal controls in business processes* on a modeling level, with this third contribution the question about the automation of business process compliance is answered on the execution level. The system ICR provides a preventive kind of business process compliance by detecting non-compliant business process instances automatically and reacting to possible control violations as required in the control. The automation is technically realized using a rule-based approach that builds on models that were provided by the first contribution.

The impact of the above contributions for standard software providers are adaptability and reusability of models through providing a model-based control repository that can be rolled out to the different customer companies. The customers can reduce their compliance costs by designing and building pattern-based controls on top of such a control repository. Furthermore, reduction of the manual effort needed to test the controls reduces the compliance costs. Further cost reductions are achieved by the automatic detection of control violations, which is enabled by the compliance validation of business process executions.

In chapter 9 the modeling and execution complexity of the approach was assessed. The complexity was assessed by discussing the modeling and execution complexity of the approach for compliance validation of business process executions. The basic result for the assessment of modeling complexity was that since the models of controlled entities in business process compliance do not add any new modeling dimension in business process models, the modeling efforts, and, therefore, the complexity remains the same compared to other existing business process modeling approaches. The compliance validation of business process executions adds a linear time complexity to the overall system responsible for enacting business processes during runtime. This additional overhead was considered as acceptable given the nature of business processes that are the focus of business process compliance; usually it takes several days for each process step to be completed. The basic result of completeness assessment was that business process compliance, i.e. modeling of internal controls in business processes and assuring the compliant behavior of business process executions, cannot be completely automated using a model-driven approach. This is due to the fact that approximately 32.5 % of the common controls at companies cannot be formally modeled and therefore cannot be automatically monitored at runtime. The necessity of the human factor in terms of compliance experts for assuring business process compliance at companies remains.

10.2 Future Work

There are several directions to extend the results presented in this thesis. The main challenge is to raise the level of automation in control identification, since it is not addressed by the models and approach in this thesis.

Currently our model does not capture the internal syntax and semantics of all the entities involved in business process compliance (see Definition 4.1). For instance the model does not make any statement about the composition of risk and its formalization. In addition, capturing the semantics of the relations *interdepends* and *contradicts* is currently not performed, i.e. it is not possible to detect any contradicting, respectively interdependent controls automatically. The same applies for the adjectives *significant*, *relevant* etc, - i.e. it is not possible to determine

automatically what a significant account is or what a relevant business process in a company is. The advantages of having a precise formal model of the relations mentioned above and the risks in relation to formalized business processes would be the following: As seen in the scenario (see chapter 2), the starting point of each business process compliance project is the risk assessment for the enterprise. Given a well formalized representation of those risks and their semantic relationships to business processes and controls, a collaborative system landscape, which can propose a set of required controls on the business process according to the enterprise specific risk assessment can be provided.

Furthermore, our approach requires the manual selection of a concrete control pattern and its specific design for a business process according to the enterprise-specific compliance needs. A higher level of automation can be brought to the whole approach by building a "Risk Repository" as a starting point of the approach. Through a formal description of business level patterns in a business process, as for instance proposed in [Thom et al., 2007], an automated matching of the available control patterns presented in our work and the existing patterns in a business process can be achieved. Such a pattern matching approach can automatically propose possible control patterns to mitigate the existing risks associated with business processes. Such an approach requires that business level patterns in business processes are annotated with possible types of risks (available in the risk repository).

Another direction of future work would be to consider outsourcing scenarios related to business processes between companies. In outsourcing scenarios an organization uses other external service organizations to perform outsourced services. These services are still part of an organization's overall operations and responsibility and, consequently, need to be considered in the overall internal control process; they are thus subject to business process compliance. In this context [PCAOB04] specifically addresses the service auditor's reports. It states: "*The use of a service organization does not reduce management's responsibility to maintain effective internal control over financial reporting. Rather, the management should evaluate controls at the service organization, as well as related controls at the company, when making its assessment about internal control for financial reporting.*"

In this context the research question would be how to automatically detect possible control violations at a partner company, to which parts of the business process have been outsourced. The challenge would be to effectively and efficiently control and react to potential control violations at a partner company without forcing that company to expose its internal business data which may not be directly related to achieving business process compliance.

References

[Agarwal, 2007]
Agarwal S.: Formal Description of Web Services for Expressive Matchmaking. PhD thesis, University of Karlsruhe (TH), May 2007.

[Agrawal et al., 2006]
Agrawal R., Johnson Ch., Kiernan J., Leymann F.: Taming Compliance with Sarbanes-Oxley Internal Controls Using Database Technology. In: Proceedings of 22. International Conference on Data Technology, p. 92, 2006.

[Albert, 2006]
Albert L.: Average case complexity analysis of RETE pattern-match algorithm and average size of join in Databases. In: Foundations of Software Technology and Theoretical Computer Science, ISBN 978-3-540-52048-1, January 2006.

[Alexander, 1979]
Alexander C.: The Timeless Way of Building. Oxford University Press, 1979.

[Allen, 1984]
Allen J.F.: Towards a General Theory of Action and Time. In: Artificial Intelligence. 23(2):123-154, 1984.

[Ankolekar et al., 2005]
Ankolekar A., Paolucci M., Sycara K.: Towards a Formal Verification of OWL-S Process Models. In: Proc. of the ISWC 2005, Galway, pp. 37-51, Ireland, 2005.

[ARIS]
www.ids-sheer.com

[Baader et al., 2003]
Baader F., Calvanese D., McGuinness D., Nardi D., Patel-Schneider P.F.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2003.

[Bace et al., 2006]
Bace J., Rozwell C.: Understanding the Components of Compliance. Gartner Research paper, ID Number: G00137902, July 2006.

[BaselII08]
http://www.bis.org/publ/bcbsca.htm, Visited 25.03.2008

[Basu et al., 2002]
Basu A., Kumar A.: Workflow Management Issues in e-Business. In: Information Systems Research 13(1), pp. 1–14, 2002.

[Balzer et al., 2004]
Balzer S., Liebig, T., Wagner M:: Pitfalls of OWL-S: a practical semantic web use case. In ACM International Conference on Service Oriented Computing, pp. 289-298, 2004.

[Bernard et al, 2002]
Bernard J.-G., Aubert A. B.: Le risque: un model conceptuel d'integration. Montréal, CIRANO: Centre interuniversitaire de recherche en analyse des organisa-tions, 2002

[BITS05]
BITS Operational Risk Management Working Group: Improving Compliance Efficiencies by Minimizing Redundancy. May 2005, http://www.bitsinfo.org/, Retrievend 25.03.2008.

[Boer et al., 2007]
Boer A., Gordon T. F., van den Berg K., Di Bello M., Förhécz A., Vas R.: Specification of the legal knowledge interchange format. Deliverable 1.1, Estrella. 2007.

[Bontas et al., 2006]
Bontas E.P., Tempich C., Sure Y.: ONTOCOM: A Cost Estimation Model for Ontology Engineering. In: Proceedings of the 5th International Semantic Web Conference (ISWC 2006), volume 4273 of Lecture Notes in Computer Science (LNCS), pp. 625-639. 2006.

[Bowen et al, 1982]
Bowen K.A., Kowalski R.: Amalgamating Language and Metalanguage in Logic Programming. In: Clark e Tarnlund eds., Logic Programming, London: Academic Press, pp. 153-172, 1982.

[BPEL4People]
WS-BPEL Extension for People (BPEL4People), Version 1.0, June 2007. http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/, Retrieved 11.04.2008.

[BRG2000]
The Business Rules Group: Defining Business Rules: What Are They Really? - Final Report, Revision 1.3, July 2000.

[Bubenko et al., 1998]
Bubenko J.A., Brash D., Stirna, J.: Ekd - enterprise knowledge development user guide, 1998.

[Card et al., 1988]
Card D., Agresti W.: Measuring Software Design Complexity. In: Journal of Systems and Software, pp. 185-197, 1988.

[Cardoso, 2005]
Cardoso J.: About the Data-Flow Complexity of Web Processes. In: 6th International Workshop on Business Process Modeling, Devel-opment, and Support: Business Processes and Support Systems: Design for Flexibility. The 17th Conference on Advanced Information Systems Engineering (CAiSE'05), pp. 67-74, 2005.

[Cardoso, 2006a]
Cardoso J.: Process control-flow complexity metric: An empirical validation. In: IEEE International Conference on Services Computing (IEEE SCC 06), Chicago, USA, pp. 167-173, 2006.

[Cardoso, 2006b]
Cardoso J.: Approaches to Compute Workflow Complexity. In: Dagstuhl Seminar, The Role of Business Processes in Service Oriented Architectures. Dagstuhl, Germany, July 2006.

[Casanovas et al., 2006]
Casanovas P., Casellas N., Vallbe J.-J., Poblet M., Benjamins R., Blazquez M., Pena R., Contreras J.: Semantic web: a legal case study. In: Davies, J., Studer, R., and Warren, P., editors, Semantic Web Technologies. Wiley, 2006.

[Casati et al., 1999]
Casati F. Ceri S., Paraboschi S., and Pozzi G.: Specification and implementation of exceptions in workflow management systems. In: ACM Transactions on Database Systems, 24(3):405–451, 1999.

[Casati et al., 2000]
Casati F., Castano S., Fugini M., Mirbel I., Pernici B.: Using Patterns to Design Rules in Workflows. In: IEEE Transactions on Software Engineering 26(8), August 2000.

[CBE101]
D. Ogle, et al., Canonical Situation Data Format: The Common Base Event V1.0.1, http://www.eclipse.org/tptp/platform/documents/resources/cbe101spec/CommonBaseEvent_SituationData_V1.0.1.pdf. Retrieved 01.04.2008

[Chandramouli, 2003]
Chandramouli R.: Business Process Driven Framework for defining an Access Control Service based on Roles and Rules. In: 23rd National Information Systems Security Conference, 2003.

[Charfi et al., 2004]
Charfi A., Mezini M.: Hybrid Web Service Composition: Business Processes Meet Business Rules. In: Proceedings of the 2nd International Conference on Service Oriented Computing, 2004.
[Christian, 1989]
Christian F.: Dependability of Resilient Computers, chapter Exception Handling, pages 68 – 97. Blackwell Scientific Publications, 1989.
[Coplien, 1995]
Coplien, O. J.: A generative development process pattern language. In: O. Coplien, James; Schmidt, D., C. (Ed.), Patttern languages of program design. Readings, MA.: Addison-Wesley. 1995.
[CORTICON]
www.corticon.com, Retrieved 01.04.2008
[COSO92]
Committee of Sponsoring Organizations of the Treadway Commission (COSO), Internal Control – Integrated Framework, 1992.
[COSO-ERM04]
Committee of Sponsoring Organizations of the Treadway Commission (COSO), Enterprise Risk Management - Integrated Framework, 2004.
[Cubera et al., 2007]
WS-BPEL: Web Services Business Process Execution Language v. 2.0, OASIS Standard, April 2007, URL: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf, Retrieved 04.04.2008.
[Curtis, 1980]
Curtis B.: Measurement and Experimentation in Software Engineering". In: Proceedings of the IEEE, Vol. 68, No. 9, IEEE Computer Society Press, Los Alamitos, CA, pp. 1144-1157. 1980.
[Dean et al, 2004]
Dean M., Schreiber G., Bechhofer S., Harmelen F.v., Hendler J., Horrocks I., McGuinness D. L., Patel-Schneider P. F., Stein L. A.: OWL Web Ontology Language Reference. W3C Recommendation. February 2004. URL: http://www.w3.org/TR/owl-semantics/ Retrieved 11.04.2008
[de Beer, 2004]
de Beer H.: The LTL Checker Plugins: A Reference Manual. Eindhoven University of Technology, Eindhoven, 2004.
[Dongen et al., 2005]
Dongen B.F., van der Aalst W.M.P.: A meta model for process mining data. In: J. Castro & E. Tentiento (Eds.), Proceedings of the CAiSE Workshops, part 2 (pp. 309-320). 2005.
[Doyle, 1979]
Doyle J.: A Truth Maintenance System. In: Artificial Intelligence, 12, 1979.
[Drools]
JBoss Rules (Drools), http://www.jboss.com/products/rules, Retrieved 11.04.2008.
[Dumas et al, 2005]
Dumas M., Hofstede A. ter, van der Aalst W.M.P.: Process Aware Information Systems: Bridging People and Software Through Process Technology. Wiley Publishing, 2005.
[Dwyer et al., 1999]
Dwyer M., Avrunin G., Corbett J.: Patterns in Property Specification for Finite-State Verification. In: Proceedings of the 21st International Conference on Software Engineering, pp 411-420, May 1999.

[ERM06]
Overview of Enterprise Risk Management, http://www.casact.org/research/erm/overview.pdf, Retrieved 31.03.2008.
[Fenton, 1991]
Fenton N.: Software Metrics: A Rigorous Approach. Chapman & Hall, London, 1991.
[FDICIA91]
The Federal Deposit Insurance Corporation Improvement Act Of 1991, http://thomas.loc.gov/cgi-bin/query/z?c102:S.543.ENR:, Visited 25.03.2008.
[Forgy 1979]
Forgy C.: On the efficient implementation of production systems. Ph.D. Thesis, Carnegie-Mellon University, 1979.
[Forgy, 1982]
Forgy C.L.: Rete: A fast algorithm for the many pattern / many object pattern match problem. Artificial Intelligence, 1:17–37, 1982.
[Fox, 1992]
Fox M.S.: The TOVE Project: A Common-sense Model of the Enterprise. In: Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Belli, F. and Radermacher, F.J. (Eds.), Lecture Notes in Artificial Intelligence # 604, Berlin: Springer-Verlag, 1992.
[Gailly et al., 2005]
Gailly F., Poels, G.: Towards an OWL-formalization of the Resource Event Agent Business Domain Ontology. In: OPSW Workshop, ISWC, 2005.
[Galler et al, 1995]
Galler J., Scheer A.-W.: Workflow-Projekte: Vom Geschäftsprozessmodell zur unternehmensspezifischen Workflow-Anwendung. In: Information Management, 10(1):20–27, 1995.
[Gamma et al., 1995]
Gamma E., Helm R., Johnson R. , Vlissides J.: Design Patterns: Element of Reusable Object Oriented Software. Addison-Wesley, 1995
[Gangemi et al., 2005]
Gangemi A., Sagri M.T., Tiscornia D.: A constructive framework for legal ontologies. In: Benjamins, R., Casanovas, P., Breuker, J., Gangemi, A., eds.: Law and the Semantic Web: Legal Ontologies, Methodologies, Legal Information Retrieval, and Applications, 2005.
[Geerts et al., 1999]
Geerts G. , McCarthy W. E.: An Accounting Object Infrastructure For Knowledge-Based Enterprise Models. In: IEEE Intelligent Systems & Their Applications, July-August 1999.
[Georgakopoulos et al, 1995]
Georgakopoulos D., Hornick M., Sheth A.: An Overview of Workflow Management From Process Modeling to Workflow Automation Infrastructure. In: Distributed and Parallel Database 3, p. 119–153, 1995.
[Giannakopoulou et al., 2001]
Giannakopoulou D., Havelund K.: Automata-Based Verification of Temporal Properties on Running Programs. In: Proceedings of the 16th IEEE International Conference on Automated Software Engineering (ASE'01), pages 412–416. IEEE Computer Society Press, Providence, 2001.
[Giblin et al., 2006]
Giblin C., Mueller S., Pfitzmann B.: From regulatory policies to event monitoring rules: Towards model driven compliance automation. In: IBM Research Report. Zuerich Research Laboratory. 2006.
[Goedertier et al., 2006]
Goedertier S., Vanthienen J.: Designing Compliant Business Processes with Obligations and Permissions. In : J. Eder, S. Dustdar et al. (Eds.) BPM 2006

Workshops, LNCS, 2006.
[Gordijn et al., 2001]
Gordijn J., Akkermans H.:. E3-value: Design and Evaluation of e-Business Models. In: IEEE Intelligent Systems, Vol. 16(4) 2001.
[Gordijn, 2002]
Gordijn J.: Value based requirements engineering: Exploring innovative e-commerce ideas. PhD thesis, Vrije Universiteit Amsterdam, 2002.
[Governatori et al., 2006]
Governatori G., Milosevic Z., Sadiq S.: Compliance checking between business processes and business contracts. In: Proceedings of 10th IEEE Conference on Enterprise Distributed Object Computing (EDOC2006), Hong Kong, 16-20 Oct 2006.
[Governatori et al., 2005]
Governatori G., Milosevic Z.: Dealing with contract violations: formalism and domain specific language. In: Processings of the 9th International Enterprise Distributed Object Computing Conference (EDOC 2005), September 2005.
[Green, 2002]
Green S.: Managers's Guide to the Sarbanes Oxley Act. Hoboken, N.J.: Wiley, 2004.
[Gruhn et al., 2006]
Gruhn V., Laue R.: Complexity metrics for business process models. In: 9th international conference on business information systems (BIS 2006), vol. 85 of Lecture Notes in Informatics, pp. 1-12, 2006
[GLBA99]
Gramm-Leach-Bliley Financial Services Modernization Act, Pub. L. No. 106-102, 113 Stat. 1338 (November 1999).
[Hagerty, 2007]
Hagerty J.: SOX Spending for 2006. AMR Research, Boston USA. Nov 29, 2007.
[Hammer, 2004]
Hammer M.: Deep Change: How Operational Innovation can transform your Company, Havard Business Review, pp. 84-93, 2004.
[Handschuh, 2005]
Handschuh S.: Creating ontology-based metadata by annotation for the semantic web. Dissertation Fakultät für Wirtschaftswissenschaften (Fak. f. Wirtschaftswiss.) Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB), http://digbib.ubka.uni-karlsruhe.de/volltexte/1000003776
[Hartman, 2005]
Hartman T.: The Cost of Being Public in the Era of Sarbanes-Oxley. Foley & Lardner LLP, June 2005
[Heilmann, 2005]
Heilmann H.: Die Integration der Aufbauorganisation in Workflow-Management Systemen. In: Information Engineering, pages 147–165. 1996.
[Herbst, 2000]
Herbst H.: Business Rule-Oriented Conceptual Modeling (Contributions to Management Science). Physica-Verlag Heidelberg, 2000.
[Hoekstra et al., 2007]
Hoekstra R., Breuker J., Bello M.D., Boer Alexander: The LKIF Core ontology of basic legal concepts. In: Pompeu Casanovas, Maria Angela Biasiotti, Enrico Francesconi, and Maria Teresa Sagri, editors, *Proceedings of the Workshop on Legal Ontologies and Artificial Intelligence Techniques (LOAIT 2007)*, June 2007.
[Horrocks et al., 2004]
Horrocks I., Patel-Schneider P.F, Boley H., Tabet S., Grosof, B., Dean M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML. 2004. URL http://www.w3.org/Submission/SWRL/. Retrieved 11.04.2008.

[Horrocks et al., 2004b]
Horrocks I., Patel-Schneider P. F.: A proposal for an owl rules language. In: Proceedings of the 13th international conference on World Wide Web, (WWW 2004), New York, NY, USA, May 17-20, 2004, ACM, 2004, pp. 723-731, 2004.

[ILOG]
www.ilog.com

[JRULES]
ILOG JRules Performance Analysis and Capacity Planning, http://www.ilog.com/corporate/download/index.cfm?filename=jrules_cap_wp.pdf , Retrieved 12.04.2008.

[ITAudit04]
IT Audit Volume 7, October 1, 2004

[ITGI06]
IT Governance Institute, Control Objectives for IT and related Technologies, Version 4.0 (COBIT), 2006.

[JavaBean]
Java Bean Specification, version 1.01, Sun Microsystems, Inc, 2006.

[Jess]
Jess, The rule engine for Java Platform, http://herzberg.ca.sandia.gov/jess/ Retrieved 11.04.2008.

[jBPM]
JBoss jBPM, http://www.jboss.com/products/jbpm, Retrieved 11.04.2008.

[Jones, 1986]
Jones T. C. Programming Productivity. New York, McGraw-Hill, 1986.

[KAON2]
http://kaon2.semanticweb.org/, Retrieved 11.04.2008.

[Kim et al., 2002]
Kim H. M., Fox, M. S.: Using Enterprise Reference Models for Automated ISO 9000 Compliance Evaluation, Proceedings of the 35th Hawaii International Conference on Systems Science. 2002

[Kingston et al., 2003]
Kingston J. and Vandenberghe W.: A comparison of a regulatory ontology with existing legal ontology frameworks. In: Workshop on Regulatory Ontologies and Compliant Regulations (Worm CoRe 2003), Catania, Sicily, November 2003.

[Klueckmann, 2007]
Klueckmann J.: Business Process Design as the Basis for Compliance Management, Enterprise Architecture and Business Rules. ARIS White paper, 2007. http://www.aris-platform.com/sixcms/media.php/2646/ARIS_Expert_Paper_ Business_Process_Design_Klueckmann_2007-03_en.pdf , Retrieved 25.03.2008

[Koschmider et al., 2005]
Koschmider A., Oberweis A.:Ontology Based Business Process Description. In: Proceedings of EMOI - INTEROP'05, Enterprise Modelling and Ontologies for Interoperability, Proceedings of the Open Interop Workshop on Enterprise Modelling and Ontologies for Interoperability, Co-located with CAiSE'05 Conference, Porto (Portugal), June 2005.

[Lee et al., 1990]
Lee P. A., Anderson T.: Fault Tolerance Principles and Practice. In: 3. volume of Dependable Computing and Fault-Tolerant Systems. Springer - Verlag, 2nd edition, 1990

[Lippe et al., 2005]
Lippe S., Greiner U., Sadiq W., Schulz K.: Interoperability Issues in Cross-Organizational Business Processes. In: Proc. of CE2005, Dallas, 2005.

[Li et al., 2005]
Li Z., Han J., Jin Y.: Pattern-Based Specification and Validation of Web Services Interaction Properties. In: Proceedings of ICSOC 2005, pp. 73 - 86, Amsterdam, The Netherlands, 2005.
[Liu et al., 2007]
Liu A. Y., Müller S., Xu K.: A Static Compliance-Checking Framework for Business Process Models. In: IBM Systems Journal 46(2), 2007.
[Lloyd, 1984]
Lloyd J.W.: Logic Programming. Springer Verlag, 1984.
[Markowitz, 1952]
Markowitz, H. M.: Portfolio Selection. In: Journal of Finance 7(1), pp. 77-91, 1952.
[Martin et al. 2004]
Martin D., Burstein M. , Hobbs J.,Lassila O., McDermott .D, McIlraith S. Narayanan S., Paolucci M., Parsia B., Payne T., Sirin E., Srinivasan N., Sycara K.: OWL-S: Semantic Markup for Web Services. W3C Member Submission. November 2004. ⊥ URL http://www.w3.org/Submission/OWL-S/, Retrieved 03.04.2008.
[Matheus et al., 2005]
Matheus C., Kokar M., Baclawski K., Letkowski V: Using SWRL and OWL to Capture Domain Knowledge for a Situation Awareness Application Applied to a Supply Logistics Scenario. In: Proceedings of International Conference on Rules and Rule Markup Languages for the Semantic Web (RuleML-2005), Galway, Ireland, November, 2005.
[McCarthy et al., 1969]
McCarthy J., Hayes P. J. : Some Philosophical Problems from the Standpoint of AI. In: Machine Intelligence 4, Meltzer B. and Michie D. (eds.), Edinburgh, UK: Edinburgh University Press, pp. 463-50, 1969.
[Mernik et al., 2005]
Mernik M., Heering J. Sloane A.M.: When and how to develop domain-specific languages. In: ACM Computing Surveys, 37(4), pp. 316–344, 2005.
[MIC]
SAP Management of Internal Controls – MIC, http://www.sap.com/germany/solutions/businesssuite/erp/financials/featuresfunctions/governance.epx, Retrieved 11.04.2008.
[Motik et al., 2004]
Motik B., Sattler U., Studer R.: Query Answering for OWL-DL with Rules. In: Proceedings of The Semantic Web (ISWC2004), Third International Semantic Web Conference, LNCS vol. 3298, pp. 549-563, Japan, November 2004.
[Müller et al., 2004]
Müller R., Greiner U., Rahm E.: AGENTWORK: A Workflow-System Supporting Rule-Based Workflow Adaptation. In: Data and Knowledge Engineering, Elsevier, 2004.
[Müller et al., 2006]
Müller D., Reichert M.U., Herbst J.: Flexibility of Data-driven Process Structures. In: Eder J., Dustdar S. (eds.) Business Process Management Workshops. LNCS, vol. 4103, pp. 179–190. Springer, 2006.
[Namiri et al., 2008]
Namiri K., Stojanovic N.: Towards a Formal Framework for Business Process Compliance. In: Multikonferenz Wirtschaftsinformatik (MKWI2008), Muenchen, Germany, 2008.
[Namiri et al. 2007a]
Namiri K., Stojanovic N.: Pattern-Based Design and Validation of Business Process Compliance. In: Proceedings of OTM Federated Conferences, Cooperated Information Systems (CoopIS2007), S. 59-76, Portugal, 2007.

[Namiri et al., 2007b]
Namiri K., Kuegler M.M., Stojanovic N: A Static Business Level Verification Framework for Cross- Organizational Business Process Models using SWRL. In: 2nd International Workshop Application of Semantic Technologies (AST 2007), in conjunction with Informatik 2007, Bremen, September 2007.

[Namiri et al., 2007c]
Namiri K., Stojanovic N: Applying Semantics to Sarbanes Oxley Internal Controls Compliance. In: 2nd International Workshop Application of Semantic Technologies (AST 2007), in conjunction the Informatik 2007, Bremen, September 2007.

[Namiri et al., 2007d]
Namiri K., Stojanovic N: A Formal Approach for Internal Controls Compliance in Business Processes. In: 8th Workshop on Business Process Modeling, Development, and Support (BPMDS'07) in conjunction with CAiSE'07, Trondheim, Norway, June 2007.

[Narayanan et al., 2002]
Narayanan S., McIlraith S.: Simulation, Verification and automated composition of web services. In: Proc. of the 11th WWW Conference, USA, pp. 77 – 88, 2002

[Neumann et al, 2003]
Neumann S., Probst C., Wernsmann C.: Continuous Process Management. In: J. Becker, M. Kugeler, and M. Rosemann, editors, Process Management: A Guide for the Design of Business Processes, pages 233–250. Springer, Berlin, New York, 2003.

[Nigam et al., 2003]
Nigam A., Caswell N. S.: Business artifacts: An approach to operational specification. In: IBM Systems Journal, 42(3):428–445, 2003.

[Özsu et al., 1999]
Özsu M. T., Valduriez P.: Principles of distributed database systems (2nd ed.), Prentice-Hall, Inc., Upper Saddle River, NJ, 1999

[O'Conner et al., 2007]
O'Connor M. J., Tu S. W., Das A. K., Nyulas C. I., Shankar R. D., Musen M. A.: Efficiently Querying Relational Databases using OWL and SWRL. In: The First International Conference on Web Reasoning and Rule Systems, Innsbruck, Austria, Springer, 2007.

[Osterwalder, 2004]
Osterwalder A.: The Business Model Ontology. A Proposition in a Design Science Approach. PhD-Thesis. University of Lausanne, 2004.

[OWL2004]
http://www.w3.org/TR/owl-features/, Retrieved 11.04.2008.

[PCAOB04]
Public Company Accounting Oversight Board (PCAOB), PCAOB Accounting Standard No. 2 – An Audit of Internal Control Over Financial Reporting Performed in Conjunction with an Audit of Financial Statements, Bylaws and Rules – Standards – AS2, March 2004.

[Peng et al., 2004]
Peng L., Zhong C.: An Extended RBAC Model for Web Services in Business Processes. In: E-Commerce Technology for Dynamic E-Business, IEEE International Conference, pp. 100 – 107, 2004.

[Philipp, 1998]
Matthis P.: Ordnungsmäßige Informationssysteme im Zeitablauf - Umsetzung der GoBS im Informationssystem-Lebenszyklus. In Zeitschrift Wirtschaftsinformatik 4/1998, S. 312-317

[Ramanathan et al., 2007]
Ramanathan J., Cohen R.J., Plassmann E., Ramamoorthy K.: Role of an auditing and reporting service in compliance management. In: IBM System Journal, Volume 46, Number 2, 2007.

[Rasmussen, 2006]
Rasmussen M.: Overcoming Risk and Compliance Myopia. In: Forrseter Research, Risk and Compliance Market Landscape – Series, August 2006.
[RDFS2004]
http://www.w3.org/2001/sw/WebOnt/, Retrieved 11.04.2008.
[RDQL]
RDQL - A Query Language for RDF http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/, Retrieved 11.04.2008.
[Reichert et al., 1998]
Reichert M., Dadam P.: ADEPTflex - Supporting Dynamic Changes of Workflows Without Losing Control. In: Journal of Intelligent Information Systems - Special Issue on Workflow Managament, 10(2), pp. 93-129, 1998.
[Reijers et al., 2004]
Reijers H. A., Vanderfeesten I. T. P.: Cohesion and Coupling Metrics for Workflow Process Design. In: BPM 2004, LNCS 3080, pp. 290-305, 2004
[Robinson, 2005]
Robinson W. N.: Implementing Rule-based Monitors within a Framework for Continuous Requirements Monitoring. In: HICSS'05, Hawaii, USA, 2005.
[Roman et al., 2005]
Roman D., Keller U., Lausen H., Bruijn J.d., Lara R., Stollberg M., Pollers A., Feier C., Bussler C., and Fensel D.: Web Service Modeling Ontology. In: Applied Ontology, 1(1):77 106, 2005.
[Romeike et. al, 2006]
Romeike F., Müller-Reichart, Hein T.: Die Assekuranz am Scheideweg – Ergebnisse der ersten Benchmark-Studie zu Solvency II. In: Zeitschrift für Versicherungswesen, S. 316-321, 10/2006.
[RuleML]
Rule MArkup Language, URL: http://www.ruleml.org/, Retrieved 11.04.2008.
[Russel et al., 2005]
Russell N., van der Aalst W.M.P., ter Hofstede A.H.M.: Workflow Exception Patterns. In: Proceedings of the 18th International Conference on Advanced Information Systems Engineering (CAiSE 06), Luxembourg, 5-6 June 2006.
[Sadiq et al., 2007]
Sadiq S., Governatori G., Namiri K.: Modeling Control Objectives for Business Process Compliance. In: 5th International Conference on Business Process Management (BPM07), S. 149 – 164, Australia, 2007.
[Sandhu et al, 1996]
Sandhu R.S., Coyne E.J., Feinstein H.L., Youman C.E. : Role-Based Access Control Models. In: IEEE Computer 29(2), pp 38-47, IEEE Press, 1996.
[Sartor, 2005]
Sartor G.: Legal Reasoning: A Cognitive Approach to the Law. Springer, 2005.
[Sienou et al., 2006a]
Sienou A., Karduck A. P.: Towards a Framework for Integrating Risk and Business Process Management. In: 12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 06), Saint-Etienne, France, 2006.
[Sienou et al., 2006b]
Sienou A., Karduck A. P., Lamine E., Pingaud H.: Management of Business Processes: The Contribution of Risk Management. In: European Collaborative Electronic Commerce Technology and Research (CollECTeR Europe 2006), Basel, Switzerland, 2006.
[Soffer et al., 2003]
Soffer P.; Golany B.; Dori D.: ERP modeling: a comprehensive approach. In: Information Systems 28(6), pp. 673-690, 2003.

[SOX02]
Pub. L. 107-204. 116 Stat. 754, Sarbanes Oxley Act, 2002.
[SPARQL]
SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/, Retrieved 11.04.2008.
[Spyns et al., 2002]
Spyns P., Meersman R., Jarrar M.: Data modeling versus Ontology engineering. In: SIGMOD record, Special Issue on Semantic Web and Data Management 31 (2002) 12–17, 2002.
[SQWRL]
Semantic Query-Enhanced Web Rule Language (SQWRL], http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL, Retrieved 11.04.2008.
[Stojanovic et al., 2006]
Stojanovic L., Happel H.J.: Ontoprocess - a prototype for semantic business process verification using SWRL rules. In: 3rd European Semantic Web Conference, 2006.
[Striemer et al, 1995]
Striemer R., Deiters W.: Workflow Management - Erfolgreiche Planung und Durchführung von Workflow-Projekten. Arbeitsbericht, Fraunhofer Institut für Software- und Systemtechnik, 1995.
[Studer et al., 1998]
Studer R. ; Benjamins V. R. ; Fensel, D.: Knowledge engineering: Principles and methods. In: Data & Knowledge Engineering Volume 25, Issues 1-2, cf. pp. 161-197, 1998.
[Sun et al., 2005]
Sun S., Zhao J.L., Nunamaker J.: On the Theoretical Foundation for Data Flow Analysis in Workflow Management. In: Americas Conference on Information Systems, 2005.
[Taveter et al., 2001]
Taveter K., Wagner G.: Agent-oriented enterprise modeling based on business rules. In: Proc. of 20th Int. Conf. on Conceptual Modeling (ER2001). LNCS, Springer-Verlag 2001.
[Thom et al., 2007]
Thom L., Iochpe C., Reichert M.: Workflow Patterns for Business Process Modeling. In: B. Pernici, J.A. Gulla (Eds.): Proc. of the CAiSE'06 Workshops (Vol. 1) - 8th Int'l Workshop on Business Process Modeling, Development, and Support (BPMDS'07), Trondheim, Norway. pp. 349-358, June 2007.
[Thomas et al., 1997]
Thomas R.K., Sandhu R.S: Task Based Access Controls. In: 11. International Conference in Database Security, pp. 166 – 181, 1997
[Ushold et al., 1996]
Ushold, M., Gruninger, M.: Ontologies: Principles, Methods and Applications. In: The Knowledge Engineering Review 11, pp. 93–136, 1996.
[Ushold et al., 1998]
Uschold M., King M., Moralee S., Zorgios Y.: The Enterprise Ontology. In: The Knowledge Engineer Review 13(1), 1998.
[Valente et al., 1999]
Valente, A., Breuker, J.A. and Brouver, P.W.: Legal Modeling and automated reasoning with ON-LINE. In: International Journal of Human Computer Studies, 51, p. 1079–1126, 1999.
[van der Aalst, 1997]
van der Aalst, W.M.P.: The application of Petri nets to workflow. In: Journal of Circuits, Systems and Computes 7(1), pp. 21–66, 1997.

[van der Aalst et al, 2002a]
van der Aalst W.M.P., van Hee K.: Workflow Management: Models, Methods, and Systems.In: The MIT Press, 2002.
[van der Aalst, 2005]
van der Aalst W. M. P.: Pi Calculus Versus Petri Nets: Let Us Eat 'Humble Pie' Rather Than Further Inflate the 'Pi Hype'. In: BPTrends 3, No. 5, pp. 1–11, 2005.
[van der Aalst et al.,2005a]
van der Aalst W. M. P., de Beer H. T., van Dongen, Boudenwijn F.: Process Mining and Verification of Properties: An Approach Based on Temporal Logic. In: Proceedings of OTM Federated Conferences, Cooperated Information Systems (CoopIS), S.130-147, 2005
[van der Aalst et al., 2005b]
van der Aalst W.M.P., Weske M., Gruenbauer. Case handling: a new paradigm for business process support. In: Data and Knowledge Engineering, 53:129–162, 2005.
[van der Aalst et al., 2005c]
van der Aalst W.M.P., Hofstede A. H. M. t. : YAWL: Yet Another Workflow Language. In: Information Systems Frontiers 30(4), pp. 245-275, 2005.
[Wagner, 2002]
Wagner G.: How to design a general rule markup language? In: XML Technologien fuer das Semantic Web - XSW 2002, Proceedings zum Workshop, Berlin, Juni 2002. (2002)
[Wang et al, 2005]
Wang J., Kumar A.: A framework for document-driven workflow systems. In: International Conference on Business Process Management (BPM), pp 285–301, 2005.
[WikiCompleteness]
http://en.wikipedia.org/wiki/Completeness, Retrieved 11.04.2008
[Woods et al., 2006]
Woods D., Mattern T.: Enterprise SOA. O'Reilly, April 2006.
[XPDL]
XML Process Definition Language (XPDL), Workflow Management Coalition (WfMC) Specification, http://www.wfmc.org/standards/xpdl.htm , Retrieved 11.04.2008.
[Xu et al., 2006]
Xu K., Liu Y., Wu C.: BPSL Modeler–Visual Notation Language for Intuitive Business Property Reasoning. In: Proceedings of the 5th International Workshop on Graph Transformation and Visual Modeling Techniques, Vienna, Austria, pp. 205–214, 2006.
[zur Muehlen, 2004]
zur Muehlen M.: Workflow-based Process Controlling. Foundation, Design, and Implementation of Workflow-driven Process Information Systems., volume 6 of Advances in Information Systems and Management Science. Logos, Berlin, 2004.
[zur Muehlen et al., 2005]
zur Muehlen M., Rosemann M.: Integrating Risks in Business Process Models. In: 16th Australasian Conference on Information Systems, Sydney, Australia, 2005