

## D2.6

Version: 2.0

Date: 2010-12-30

Dissemination status: PU

Document reference: D2.6



# Implementation of an Integrated Prototype Handling Interactive User Specified Compliance Requests in a Compliance Language

Project acronym: COMPAS

Project name: Compliance-driven Models, Languages, and Architectures for Services

Call and Contract: FP7-ICT-2007-1

Grant agreement no.: 215175

Project Duration: 01.02.2008 – 28.02.2011 (36 months)

Co-ordinator: TUV Technische Universitaet Wien (AT)

Partners: CWI Stichting Centrum voor Wiskunde en Informatica (NL)

UCBL Université Claude Bernard Lyon 1 (FR)

USTUTT Universitaet Stuttgart (DE)

TILBURG UNIVERSITY Stichting Katholieke Universiteit Brabant (NL)

UNITN Università degli Studi di Trento (IT)

TARC-PL Telcordia Poland (PL)

THALES Thales Services SAS (FR)

PWC Pricewaterhousecoopers Accountants N.V. (NL)

This project is supported by funding from the Information Society Technologies Programme under the 7th Research Framework Programme of the European Union.





Project no. 215175

**COMPAS**

**Compliance-driven Models, Languages, and Architectures for Services**

Specific Targeted Research Project

Information Society Technologies

Start date of project: 2008-02-01      Duration: 36 months

## **D2.6 Implementation of an Integrated Prototype Handling Interactive User Specified Compliance Requests in a Compliance Language**

Revision 2.0

Due date of deliverable: 2010-12-31

Actual submission date: 2010-12-30

Organisation name of lead partner for this deliverable:

TILBURG UNIVERSITY, Stichting Katholieke Universiteit Brabant (NL)

Contributing partner(s):

TUV, Vienna University of Technology (AT)

UNITN, Universita degli Studi di Trento (IT)

USTUTT, Universitaet Stuttgart (DE)

CWI, Stichting Centrum voor Wiskunde en Informatica (NL)

UCBL, Université Claude Bernard Lyon 1 (FR)

<b>Project funded by the European Commission within the Seventh Framework Programme</b>		
<b>Dissemination Level</b>		
<b>PU</b>	Public	X
<b>PP</b>	Restricted to other programme participants (including the Commission Services)	
<b>RE</b>	Restricted to a group specified by the consortium (including the Commission Services)	
<b>CO</b>	Confidential, only for members of the consortium (including the Commission Services)	

## History chart

Issue	Date	Changed page(s)	Cause of change	Implemented by
1.1.1	2010-11-22	All sections	New document	TILBURG, TUV, UNITN, USTUTT, CWI, UCBL
1.2	2010-11-27	Section 2	Updates	TILBURG
1.3	2010-11-30	Section 3.2	Internal review	TILBURG
1.4	2010-12-06	All sections	Partner reviews (RV-UCBL-D2.6 and RV-UNITN_D2.6rev2760)	UCBL, UNITN, TILBURG
2.0	2010-12-30		Approval	TUV

## Authorisation

No.	Action	Company/Name	Date
1	Prepared	TILBURG	2010-12-10
2	Approved	TUV	2010-12-30
3	Released	TUV	2010-12-30

Disclaimer: The information in this document is subject to change without notice. Company or product names mentioned in this document may be trademarks or registered trademarks of their respective companies.

### All rights reserved.

The document is proprietary of the COMPAS consortium members. No copying or distributing, in any form or by any means, is allowed without the prior written agreement of the owner of the property rights.

This document reflects only the authors' view. The European Community is not liable for any use that may be made of the information contained herein.

## Contents

1. Introduction .....	6
1.1. Purpose and scope .....	6
1.2. Document overview .....	7
1.3. Definitions and glossary .....	7
1.4. Abbreviations and acronyms .....	7
2. Integration within the COMPAS Architecture .....	7
3. CRLT Components – Overview .....	9
3.1. Conceptual Model (of the CRR) .....	10
3.2. Compliance Requirements Manager (CompRM) Component .....	11
3.2.1. Manage Compliance Requirements .....	12
3.2.2. Search Compliance Requirements .....	13
3.2.3. Manage Compliance Source .....	13
3.2.4. Manage Compliance Risk .....	14
3.2.5. Manage Controls .....	15
3.2.6. Manage Compliance Rules .....	16
3.3. Rule Modeller .....	16
3.3.1. Implementation .....	17
3.4. Design-time Compliance Request Manager (DCRM) .....	19
4. Installation of the Components .....	23
4.1. Web-based Components .....	23
4.2. Rule Modeller Installation .....	23
5. Reference documents .....	25
5.1. Internal documents .....	25
5.2. External documents .....	26

## List of figures

Figure 1	Overall COMPAS architecture .....	8
Figure 2	CRLT Main Page .....	10
Figure 3	Conceptual model of the CRR .....	11
Figure 4	Main page of the CompRM .....	12
Figure 5	User Interface for Managing Compliance Requirements .....	12
Figure 6	User Interface for Searching Compliance Requirements .....	13
Figure 7	User Interface for Managing Compliance Sources.....	14
Figure 8	User Interface for Managing Compliance Risks.....	14
Figure 9	User Interface for Managing Controls .....	15
Figure 10	User Interface for Managing Compliance Rules .....	16
Figure 11	Rule Modeller User Interface .....	18
Figure 12	DCRM - Formulating Compliance Requests.....	20
Figure 13	DCRM - Viewing Compliance Requests.....	21
Figure 14	DCRM - Presenting Compliance Check Results .....	22
Figure 15	DCRM - Presenting the Details of the Check Results.....	23
Figure 16	Rule Modeller installation wizard.....	24
Figure 17	Rule Modeller - New Project Window .....	24
Figure 18	Rule Modeller - Domain-Specific Language Designer Wizard.....	25

## Abstract

This deliverable presents the final version of the integrated prototype D2.6 (Compliance Request Language Tools - CRLT). The initial version of the prototype was released in M23. The deliverable describes how the CRLT prototype is integrated with the COMPAS Architecture and presents briefly its main features, design and implementation details. In resume, the CRLT supports the management of the compliance requirements stored in the Compliance Requirements Repository (CRR) and the design-time compliance verification of compliance targets.

# 1. Introduction

## 1.1. Purpose and scope

The major goal of work package 2 (WP2) is developing a language for user requests and provider constraints, and a language of regulation compliance concerns, as examples of high-level compliance languages based on the concepts from WP1. This mainly involves the design of compliance language modules and supporting infrastructure.

Deliverable 2.1 (M6) presents an overview of the state-of-the-art in compliance languages, particularly focusing on languages for regulatory and legislative provisions. Mandatory features that should exist in a compliance language are also reported. One of the main findings of this study is that the compliance requirements should be based on a formal foundation of a logical language to pave the way for automatic reasoning techniques that assist in verifying business process compliance particularly during design-time phase of the business process lifecycle.

Deliverable 2.2 (M11) examines the languages that can be used as the basic building blocks for a comprehensive Compliance Request Language (CRL) and reports a comparative analysis between these candidate formalisms (also published in [ETH+10a] & [ETH+10b]). For this deliverable [D2.2], we analyzed a wide range of compliance legislations and frameworks and studied a variety of relevant works on the specification of compliance requirements. We identified structural patterns of frequently recurring (compliance) requirements imposed on the business processes.

Deliverable 2.3 (M18) presents a meta-model for the CRL and proposes extensions to address the limitations identified in D2.2. The CRL integrates property specification patterns, Linear Temporal Logic (LTL) formalism and higher-level compliance concepts for the specification of business process compliance requirements for design-time. D2.3 also presents the main features and architecture of a software environment for the CRL, which is further elaborated by the successive deliverable D2.6.

Deliverable 2.6 (M23) presents the *initial* version of the prototype [M23] for the run-time environment (Compliance Request Language Tools – CRLT) together with the Compliance Requirements Repository (CRR), where data maintained by the CRLT resides.

Deliverable 2.4 (M30) presents the ‘Rule Modeller’ – the standalone component of the CRLT- that allows the compliance expert to visually represent a particular compliance constraint by using property specification patterns and business process constructs (operands), and to automatically generate Linear Temporal Logic (LTL) formulas derived from these pattern-based expressions.

This prototype deliverable D2.6 (M35) presents the final version of the integrated prototype for CRLT<sup>1</sup>. The CRLT comprises components (including the Rule Modeller) that allows the definition and management of the compliance requirements; handling of interactive user specified compliance requests in a compliance language (design-time verification of the compliance targets), and the design of visual representations of compliance requirements using patterns for the automated generation of formal compliance rules.

## 1.2. Document overview

The remainder of this document is structured as follows: Section 2 describes the integration between the CRLT and the other components of the COMPAS Architecture defined in [DA.1]. Section 3 briefly describes the underlying conceptual model of the CRLT and presents the main features of the CRLT components by going through their user interfaces. Finally, section 4 presents the installation procedure of the components.

## 1.3. Definitions and glossary

The most important terminology concerning the COMPAS project is listed on the public COMPAS Web-Site [D7.1] available at <http://www.compas-ict.eu> section terminology. This helps to make the overall COMPAS approach more comprehensive for the general public.

## 1.4. Abbreviations and acronyms

CompRM	Compliance Requirements Manager
CRL	Compliance Request Language
CRLT	Compliance Request Language Tool
CRR	Compliance Requirements Repository
DCRM	Design-time Compliance Request Manager
LTL	Linear Temporal Logic
WP	Work package

## 2. Integration within the COMPAS Architecture

Figure 1 depicts the COMPAS Architecture that provides relevant prototypes, tools, and technologies, with the use cases provided by the industry partners in order to demonstrate the pragmatic use of COMPAS contributions. The components enclosed in the figure represents CRLT's context together with the CRR.

CRR stores compliance requirements in various levels of abstraction and is integrated with the 'Model Repository', where models including the definitions of compliance targets; such as, business processes, business process activities, and web services, reside. The Model Repository is also integrated with the 'Process Fragments Repository' (cf. [D4.4]). Repositories can be accessed through a set of web services maintained by WP1.

---

<sup>1</sup> Compliance Request Language Tools (CRLT) web site: <http://eriss.uvt.nl/compas/>

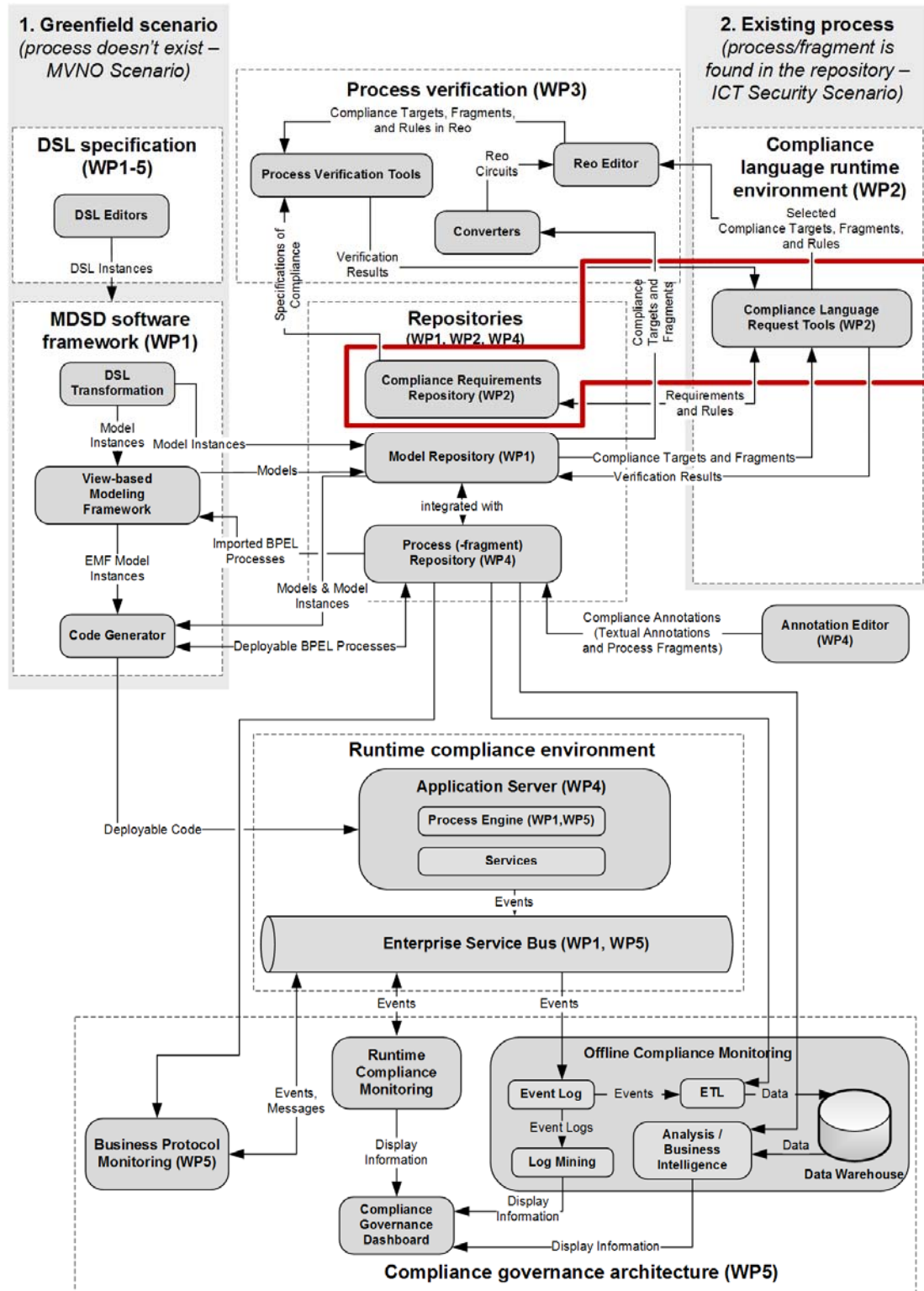


Figure 1 Overall COMPAS architecture

CRLT is used to populate CRR with compliance requirements and related concepts (such as compliance sources, risks, controls and rules<sup>2</sup>) in line with the COMPAS unified conceptual model and to associate them with the compliance targets that reside in the Model Repository (WP1). CRLT integrates with the CRR to retrieve necessary elements to build graphical pattern-based expressions of controls and to forward back the formal compliance rules. For design-time compliance verification, CRLT enables formulating compliance request by integrating with MORSE and CRR to retrieve compliance targets that are subject to design-time compliance verification (mainly end-to-end business processes) against selected compliance requirements. In doing so, it also integrates with the *Process Verification Tools* from WP3. It forwards the compliance target specifications and the formal compliance rules to the Process Verification Tools and retrieves back the static verification results. The results are analysed and presented to the end-user (business process expert) together with their root-causes and stored for future reference. (A brief overview of the design-time compliance management approach summarized here is presented in [ETH+10c]. The details of the overall approach focusing on its integration with the process fragments are published as a joint paper [STK+10])

### 3. CRLT Components – Overview

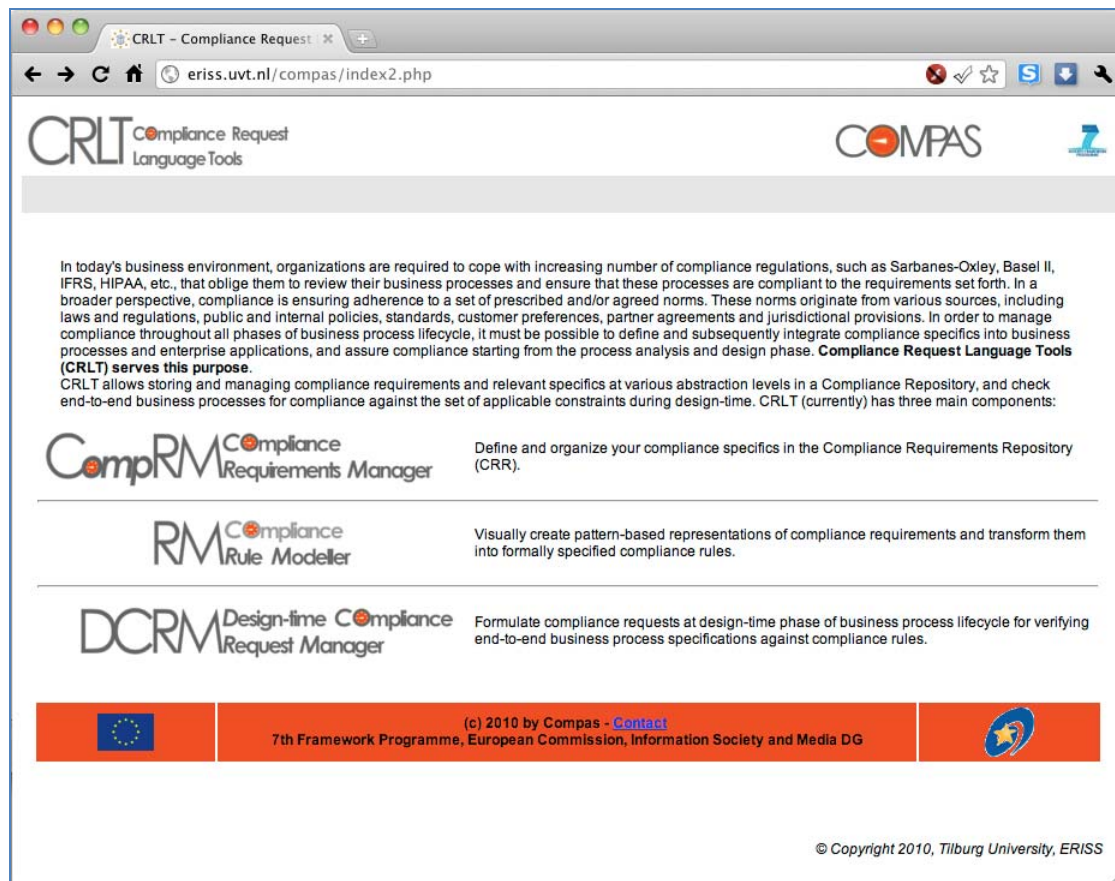
The internalization and definition of compliance requirements and relevant concepts, establishing their internal and external relationships with business processes, and managing the evolution of these requirements as a separate entity in a repository is one of the key elements of business process compliance. CRLT in the COMPAS Architecture is developed mainly to serve this purpose. CRLT has three main components:

- *Compliance Requirements Manager (CompRM)* is a web-based application that allows compliance and business experts to define, store and manage compliance requirements and relevant concepts (sources, risks, etc.) in the CRR.
- *Rule Modeler* is a standalone component of the CRLT that enables compliance experts to visually create pattern-based representations of controls and transform these expressions into compliance rules (as formally represented logical formulas).
- *Design-time Compliance Request Manager (DCRM)* is a web-based application for formulating compliance requests at design-time phase of business process compliance for verifying compliance targets (mainly business process specifications) against relevant compliance requirements. It integrates with WP1 and WP3 prototypes and presents the analysis of the verification results to the business and/or compliance experts to aid the resolution in case of violations.

Figure 2 shows the main page of the CRLT web site that introduces the toolset and directs users to its components.

---

<sup>2</sup> Please refer to the terminology section on the public COMPAS Web-Site [D7.1] available at <http://www.compas-ict.eu> for the definition of the terms (such as compliance target, compliance requirements, compliance rules, compliance expert, business process owner, and etc.) used in this document.



**Figure 2 CRLT Main Page**

In below sections, first, we describe the underlying conceptual model on which the CRLT & CRR is implemented. Then, we briefly describe the CRLT components listed above.

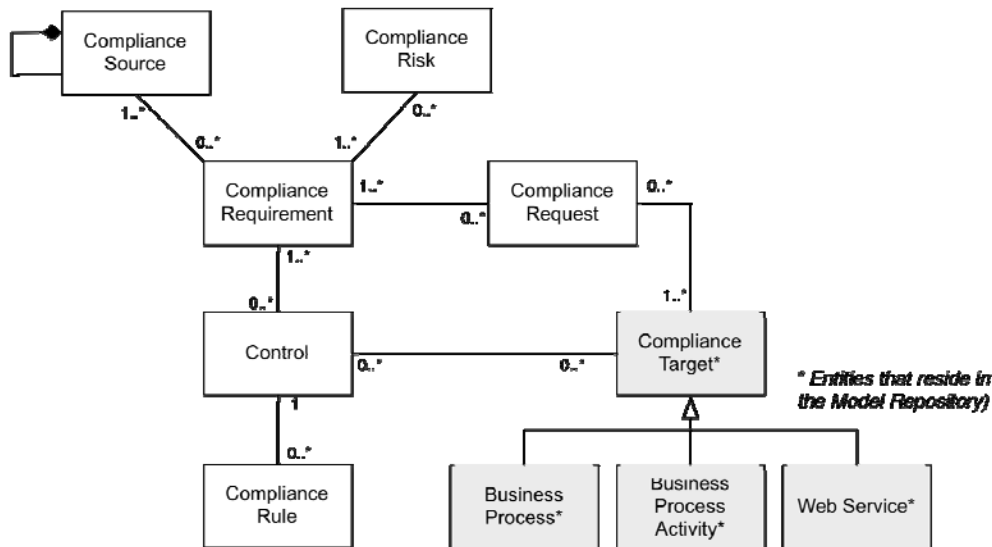
### 3.1. Conceptual Model (of the CRR)

The conceptual model of the compliance specifics (requirements, sources, risks, controls, etc.) maintained in the CRR and managed through CRLT is based on the COMPAS unified conceptual model<sup>3</sup>. Figure 3 presents these concepts and their relationships.

A *Compliance requirement* is a constraint or assertion that results from the interpretation of the *compliance sources*, such as laws, regulations, policies, standards, contracts, etc. Failure to meet these requirements increases the likelihood of a *compliance risk* to materialize, which in turn might impair the organization's business model, reputation and financial condition. To mitigate these risks and ensure that compliance requirements are satisfied, the organization defines *controls*. A control describes the restraining or directing influence to check, verify or enforce rules to satisfy one or more compliance requirements. A *Compliance rule* is an operative definition of a compliance requirement, which formally describes a control. A *Compliance target* is a generic specification (such as a business process) representing the 'object' of compliance requirements. A user (compliance or business expert) can issue a *compliance request* to check whether a set of compliance targets conforms to a set of

<sup>3</sup> COMPAS unified conceptual model is available at the COMPAS Web-Site [D7.1]: <http://www.compas-ict.eu/terminology>

applicable compliance requirements. The purpose of a compliance request is to identify if and how a process can or should be changed to make it (more) compliant.



**Figure 3 Conceptual model of the CRR**

Please refer to the previous version of this deliverable D2.6 [M23] for detailed descriptions of relevant attributes of the concepts. The terminology section on the public COMPAS Web-Site [D7.1] (available at <http://www.compas-ict.eu/terminology>) also provides the definitions and examples of the concepts described here.

### 3.2. Compliance Requirements Manager (CompRM) Component

Organizations go through an internalization process, which mainly involves the analysis of compliance sources (such as regulations, laws, standards, internal/external policies, etc.) and assessment of risks to the achievement of the objectives and constraints mandated by these sources to identify the controls to mitigate the risks. *Compliance Requirements Manager (CompRM)* component of the CRLT allows business and compliance experts to manage these requirements and relevant concepts in a structured way.

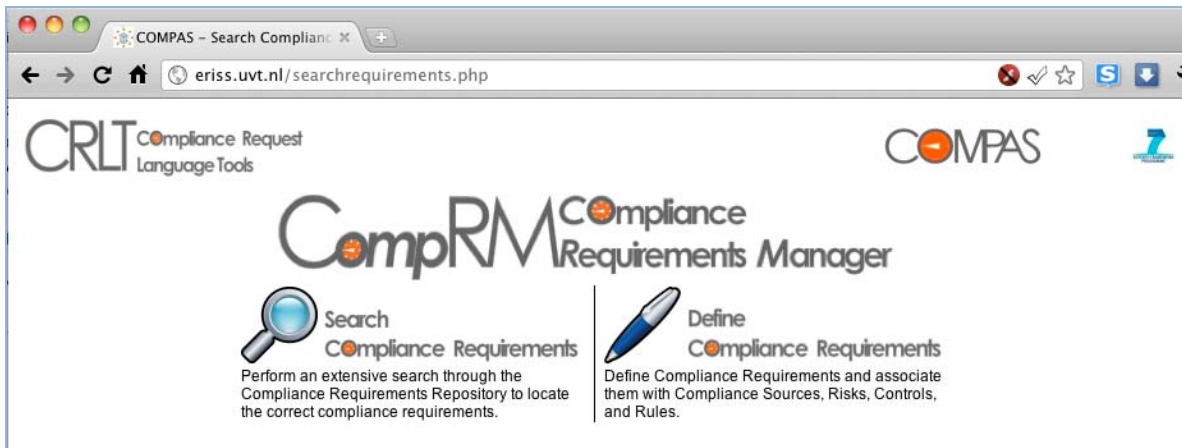
CompRM prototype component is implemented as a web application using PHP<sup>4</sup> as the main scripting language. It can be reached through <http://eriss.uvt.nl/compas><sup>5</sup>. Oracle database (ver.9i) is used for the CRR.

Figure 4 shows the main page of the CompRM component. Users can search compliance requirements that are already defined in the repository or starts defining a requirement from scratch by clicking 'Define compliance Requirements' link.

<sup>4</sup> <http://www.php.net/>

<sup>5</sup> The source code of the CRLT is packed in two accompanying files:

- 1- Source code of the web site: D2.6\_M35\_Source-Code\_Web-Site.zip
- 2- Setup files and the source code for the Rule Modeller: D2.6\_M35\_Source-Code\_Rule\_modeler\_v05\_031210.rar

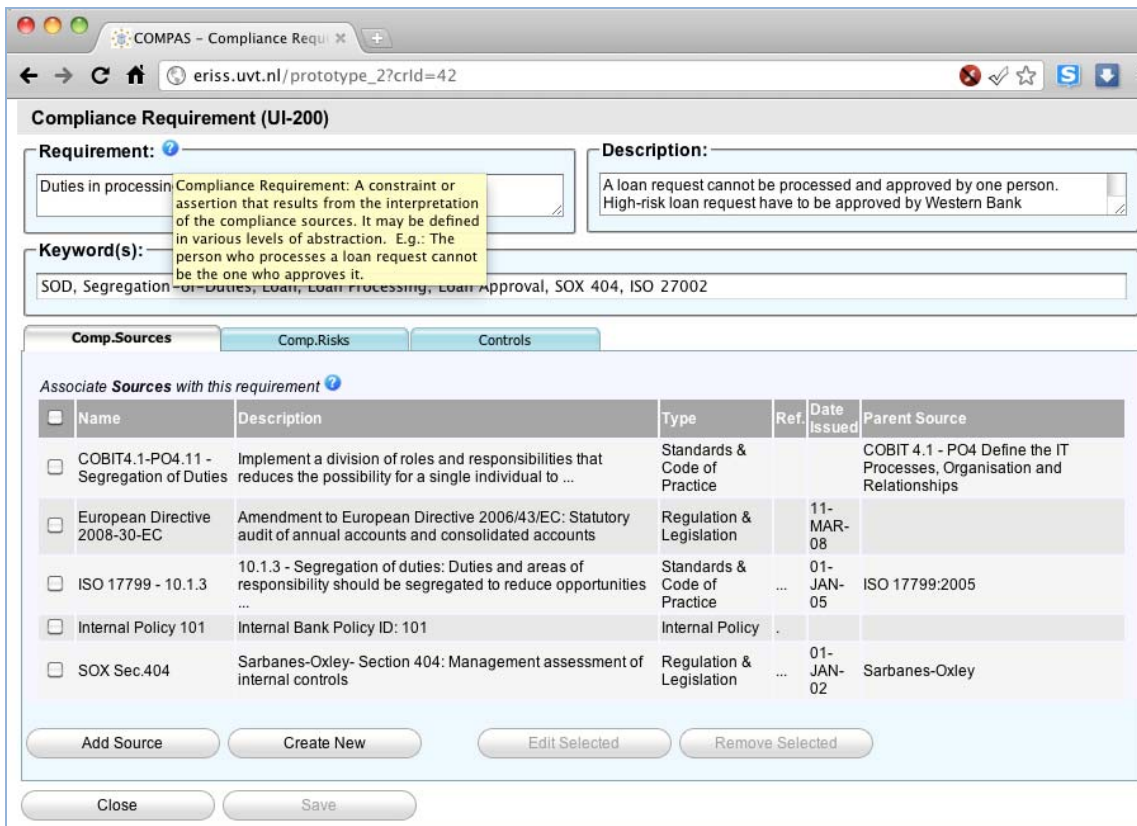


**Figure 4 Main page of the CompRM**

In the previous version of this prototype deliverable ([D2.6]-M23), we describe the functionalities and the design of CRLT components in detail. In this deliverable we briefly describe the main features by going through key user interfaces. The following paragraphs summarise these features. Please refer to [D2.6]-M23 for the details regarding the requirements and the internal design of the components.

### 3.2.1. Manage Compliance Requirements

CompRM component allows users to manage (define and maintain) compliance requirements mainly through the interface presented in Figure 5.



**Figure 5 User Interface for Managing Compliance Requirements**

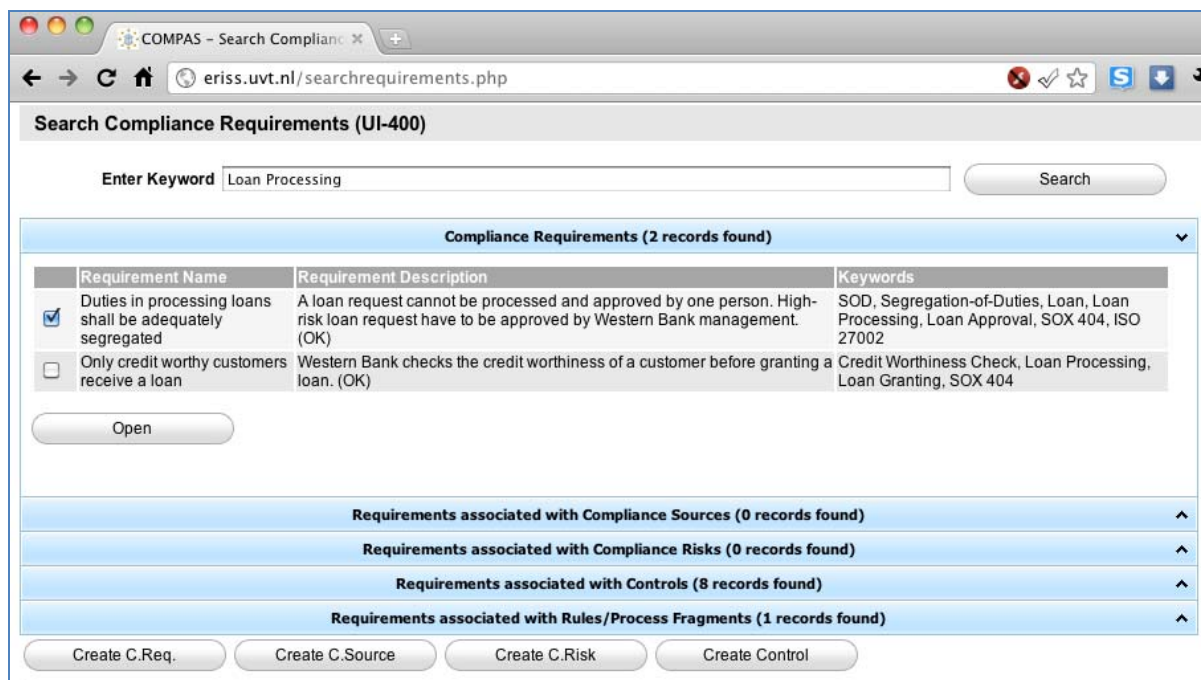
In addition to the name and the description of the compliance requirements, relevant keywords can be defined to facilitate searching the requirements defined in the CRR. The main interface contains separate tabs to associate a compliance requirement with relevant compliance sources, risks and controls. Each tab has a set of buttons to:

- Search and associate a concept (compliance source, risk and control) with the requirement (through ‘Add Source/Risk/Control’ button)
- Create a concept and associate it with the requirement (through ‘Create New’ button)
- Edit an associated concept (through ‘Edit Selected’ button)
- Remove the association of the concept (through ‘Remove Selected’ button)

As shown in Figure 5, the user interfaces in CompRM (and in CRLT in general) are enhanced with ‘help icons’ located close to key attributes, which provide brief information and remarks regarding the concepts to guide the user in the definition process.

### 3.2.2. Search Compliance Requirements

In order to be able to locate certain requirements available in the CRR for maintenance and/or reuse purposes, it should be possible to search in the repository through all concepts relevant to the requirements. Figure 6 presents the interface for running a search in the CRR. The search button (located at the upper-right corner of Figure 6) initiates the search through all components (sources, controls, etc.) in the CRR as well as their attributes, and retrieves and lists back the compliance requirements that are in relation to the keywords searched.



**Figure 6 User Interface for Searching Compliance Requirements**

Selecting a compliance requirement and clicking ‘Open’ will open the interface given in Figure 5 with selected requirement’s information filled in.

### 3.2.3. Manage Compliance Source

CompRM tool enables compliance sources and their sections to be managed through the interface displayed in Figure 7. Name and type attributes are mandatory for compliance

sources. A compliance source can be: a regulation or legislation; a standard or code of practice; a business contract or an agreement; or policies internal to the organization. A section or a part of a source is created as a separate source and associated with its pattern through browsing available sources in the CRR.

**Figure 7 User Interface for Managing Compliance Sources**

### 3.2.4. Manage Compliance Risk

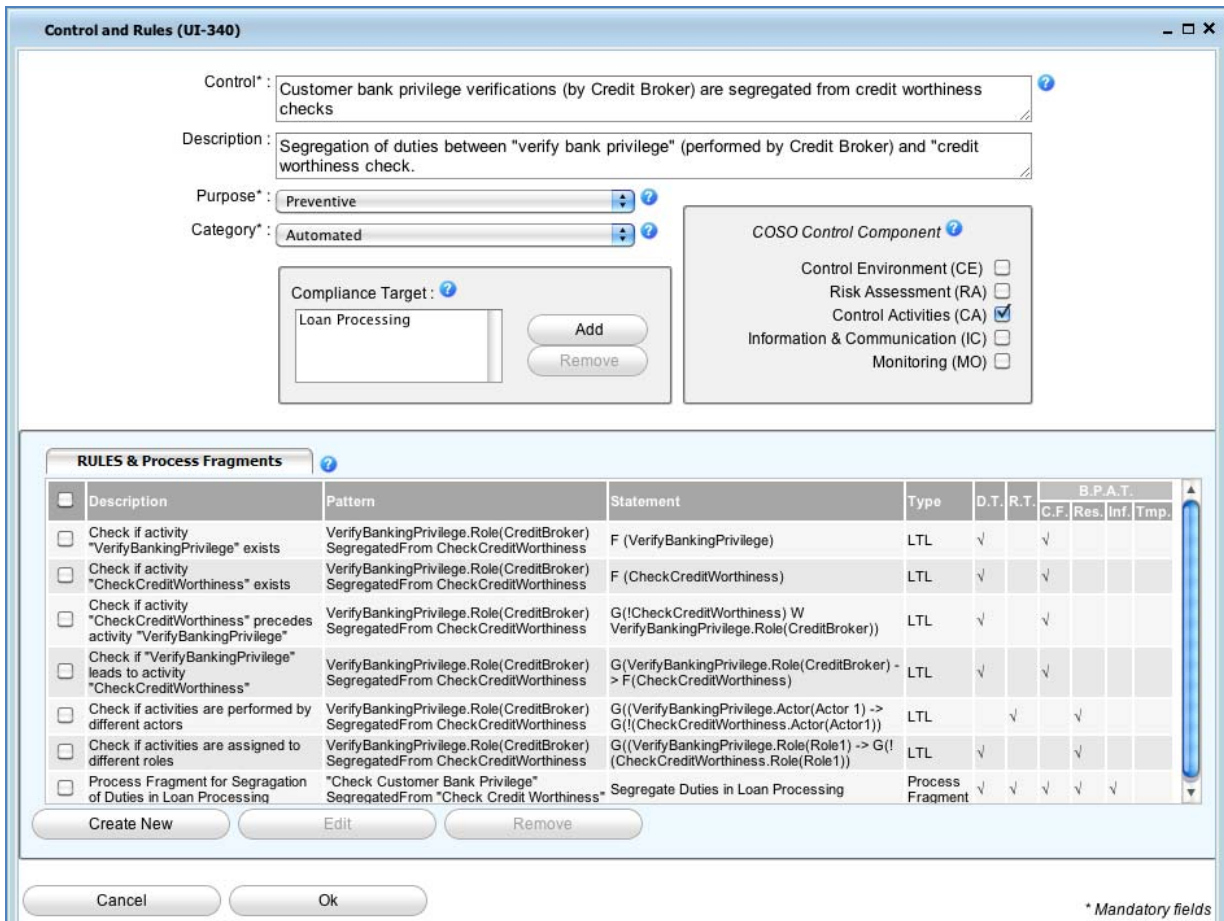
A compliance risks is the probability of occurrence of an event that might influence the achievement of certain goals, which in turn might impair the organization's business model, reputation and/or financial condition. A risk is usually measured by a combination of impact and probability of occurrence. Figure 8 presents the user interface for managing risks and their attributes.

**Figure 8 User Interface for Managing Compliance Risks**

Both attributes –probability of occurrence and impact- are selected among a scale from ‘very low’ to ‘very high’.

### 3.2.5. Manage Controls

Controls define how a compliance requirement is handled to ensure their effective implementation. They are typically concrete and organization-specific. An example of a control is presented in Figure 9.



**Figure 9 User Interface for Managing Controls**

The *purpose* of a control can be *preventive*, aiming to detect violations before they occur; or *detective*, involving after-the-fact reporting of the violations occurred. A control can be *automated* so that it can be used for automated verification; or *manual*, which requires human intervention in the form of inspections/audits for assessing compliance.

Controls can be relevant to certain control components. The abbreviations CE, RA, CA, IC and MO on the user interface (in Figure 9) represent the Control Components that are defined in COSO Internal Control Framework [COSO94] as the components that should be in place to achieve an effective internal control system. A control is relevant to one or more of the following components of an internal control framework:

- CE: Control Environment
- RA: Risk Assessment
- CA: Control Activities
- IC: Information and Communication
- MO: Monitoring

The relationship between a control and a compliance target is established through the interface given in Figure 9. By pressing ‘Add’ button, the user is provided a list of

compliance targets available in the model repository so that the control can be associated with one or more of the compliance targets listed.

The tab located in the below part of the user interface lists the corresponding formal compliance rules and process fragments that implement the control under consideration. These rules are essentially generated through the Rule Modeller component (described in Section 3.3) and transferred to CRR, from where they are retrieved by the CompRM and listed in the user interface in Figure 9. However, through this user interface users are also allowed to edit or remove the rules that have been generated by the Rule Modeller. New rules can also be created manually and associated with the control through this interface. In the next section we describe the user interface through which the new rules are created and edited.

### 3.2.6. Manage Compliance Rules

Compliance rules represent the operative definitions of controls, which also involve formal specifications. Figure 10 shows the user interface for managing compliance rules. A compliance rule has a (formal) rule statement or name attribute representing the rule specification; a pattern-based representation from which it has been generated (please refer in Section 3.3 for more details); a type attribute indicating the formalism or means used to specify the rule (such as LTL, ESPER, process fragment, etc.); the scope attribute indicating the phase at which the rule will be checked; and finally BP aspect type attribute signifying the aspect or concern to which the rule belongs to.

As discussed in above section, although the compliance rules are principally generated through the Rule Modeller component described in the next section, they can be edited through the CompRM interface given in Figure 10.

Figure 10 User Interface for Managing Compliance Rules

## 3.3. Rule Modeller

In order to bring about automated assurance of business processes to compliance requirements for the entire business process lifecycle, it is necessary to formalize those requirements that

can be represented with a formal logical language. This enables compliance targets and their executions to be verified, controlled and analysed against these rules.

Formal models of such requirements provide an accurate and unambiguous specification facilitating a lifetime assurance of business process-based systems. Although the need for formal models are clearly acknowledged by several studies, the pre-knowledge required for their use and the complexity of the formalisms remained as significant obstacles for their widely adoption by compliance and business experts. Using *patterns* is considered as a means to support hiding the complexity of formalisms and facilitate their specifications. In D2.4 [M30] we introduced a set of *patterns* for defining abstract representations of controls and based on these representations generating corresponding formal compliance rules that are particularly applicable for design-time verification. Patterns are high-level abstractions of frequently used logical formulas, which help non-technical users to represent desired properties and constraints without going into the lower-level and complex details of the underlying formal language. In addition to original patterns in [DAC98], we developed and integrated a new breed of *patterns* to capture frequently recurring requirements particularly in the *compliance context*.

We also developed a mapping scheme from patterns to a corresponding formal language (LTL in our case) to facilitate the automated transformation of pattern-based representations of controls into a set of logical formulas as compliance rules. The mapping also takes *implicit logical rules* into consideration to facilitate the analysis of violations and to provide better insight into their root-causes and remedies for their avoidance or recovery. The details regarding the generation of implicit rules and the analysis of root-causes of violations are described in detail in [ETH+10d].

Rule Modeller allows compliance experts to create graphical representations of the controls in a drag-and-drop fashion using the patterns that are integrated into the application. By implementing the mapping rules into the Rule Modeller, we also allow automated transformation of these graphical expressions into compliance rules (as formally represented logical formulas).

Pattern-based expressions of controls comprise not only patterns but also business process elements (such as activities, roles, data objects, etc.) that reside in the repositories. For example, the expression “CreateOrder *LeadsTo* ApproveOrder” has two operands (business process activities in this case) connected via the *LeadsTo* pattern. In order to ease the construction of the expressions, the Rule Modeller integrates with the repositories to retrieve these specifics and lists them to the user for selection during the construction. For example, the user has to select among the activities available in the repository to construct the sample expression given above. Once the compliance rules are generated based on these expressions, they are transferred to the CRR as compliance rules.

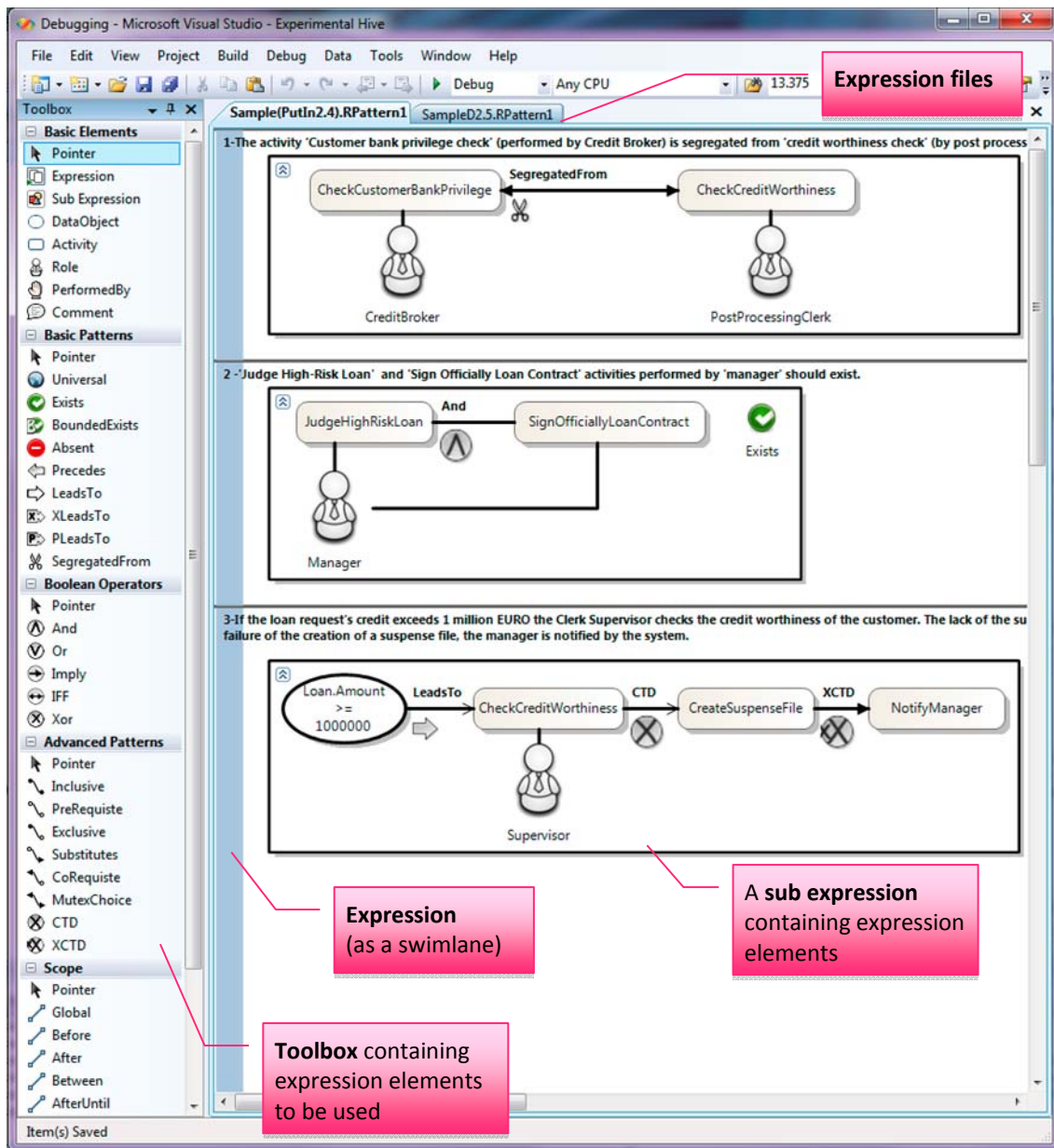
### 3.3.1. Implementation

Rule Modeller is developed as a standalone application in Microsoft Visual Studio<sup>6</sup> development environment using C# programming language. It works on Windows

---

<sup>6</sup> Microsoft Visual Studio 2008 with Visual Studio 2008 SDK supporting visual domain-specific languages, <http://msdn.microsoft.com/en-us/vstudio>.

environments<sup>7</sup>. Figure 11 presents a screenshot from the Rule Modeller displaying examples of controls represented as graphical pattern-based expressions.



**Figure 11 Rule Modeller User Interface**

On the left side of the interface the toolbox is positioned, which lists the primary elements (constructs) that are used to build the pattern-based expressions. These elements include patterns (basic and advanced), basic expression elements (such as activity, role, relationships, etc.), scope elements, and Boolean operators. The main working field, where the visual pattern-based expressions are constructed, is positioned in the centre of the interface. The expressions are built by dragging-and-dropping necessary expression elements from the toolbox onto the drawing canvas. The application controls the type and direction of the relationships that can be established between each expression element and allows only valid relationships based on a predefined meta-model.

<sup>7</sup> Current version of the software is tested successfully on Windows Vista and Windows 7 operating systems.

Figure 11 shows three examples of controls constructed using the patterns provided on the ‘Toolbox’. Table 1 gives the descriptions of the control examples and the text-based representations of the visual pattern-based expressions shown in Figure 11.

**Table 1 Descriptions of the examples shown in Figure 11**

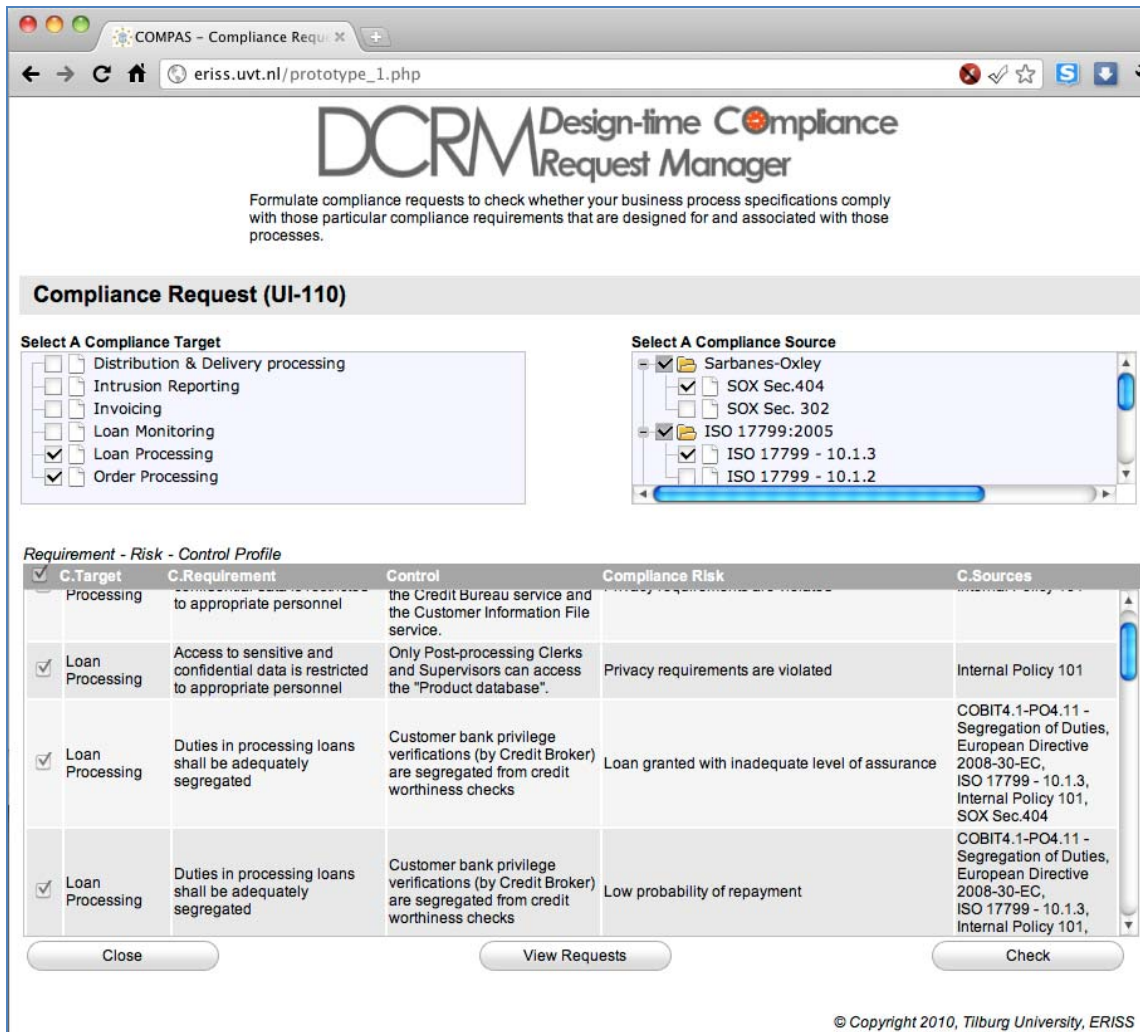
ID	Control	Pattern-based Expression (Text-based)
1	The activity ‘ <i>Customer bank privilege check</i> ’ (performed by the Credit Broker) is segregated from ‘ <i>credit worthiness check</i> ’ (performed by the Post Processing Clerk)	(CheckCustomerBankPrivilege.Role(CreditBroker) SegregatedFrom CheckCreditWorthiness.Role(Post Processing Clerk))
2	The branch office <i>Manager</i> checks whether risks are acceptable (Judge High Risk Loan) and makes the final approval of the request (Sign Officially Loan Contract).	((JudgeHighRiskLoan.Role (‘Manager’)) Exists) AND ((SignOfficiallyLoanContract.Role (‘Manager’)) Exists)
3	If the loan request’s <i>credit exceeds 1 million EURO</i> the Clerk Supervisor <i>checks the credit worthiness</i> of the customer. The lack of the supervisor check immediately creates a suspense file. In case of failure of the creation of a suspense file, the manager is notified by the system.	Loan.Amount >= ‘1M’ LeadsTo CheckCreditWorthiness.Role(Supervisor) XCTD CreateSuspenseFile CTD NotifyManager

Please refer to D2.4 [M30] for the details regarding the design and the main features of the Rule Modeller, the proposed pattern system, and the mapping between these patterns and LTL.

### 3.4. Design-time Compliance Request Manager (DCRM)

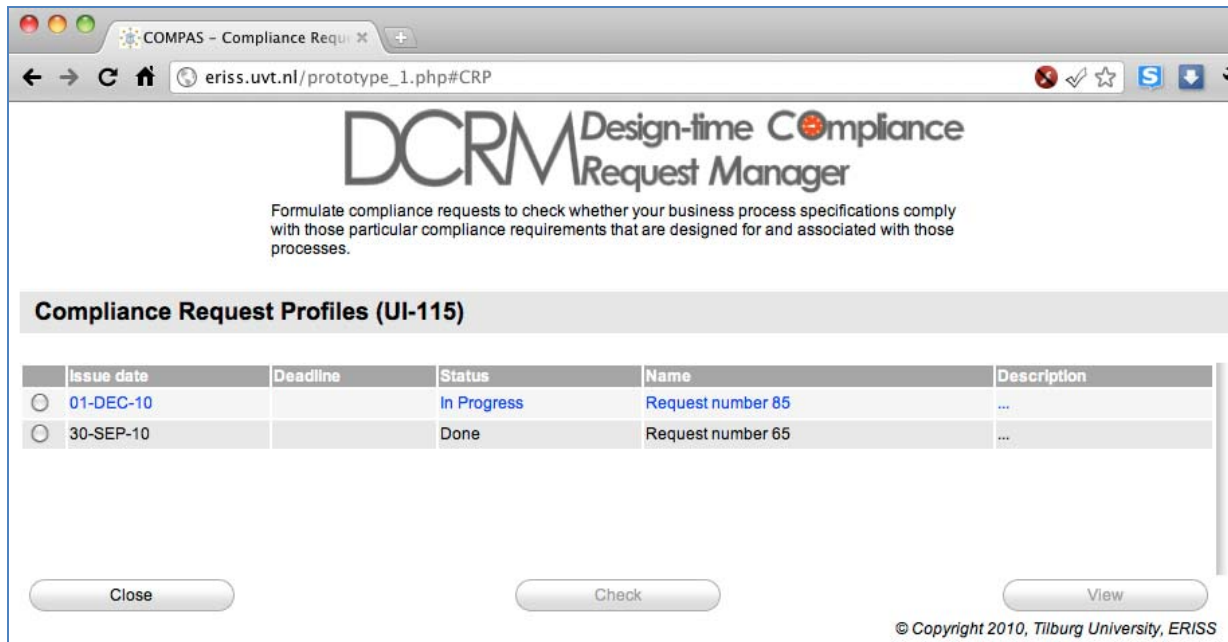
DCRM integrates with other components of the COMPAS Architecture to allow business experts to formulate compliance requests to support the verification of the compliance targets during design-time. A compliance request is issued to verify if a set of compliance targets conforms to a set of applicable compliance requirements. The verification involves the static checking of the compliance targets that reside in the Model Repository (WP1) against a set of compliance requirements (that are stored in CRR) by using the process verification tools implemented in WP3.

Figure 12 presents the user interface used to construct compliance requests. First, DCRM retrieves the compliance targets (business processes in this context) that reside in the Model Repository (through CRR) and lists them on the upper-left section of the interface. The user starts with selecting the creating a request by selecting the compliance targets to be checked. Based on the selection, DCRM automatically retrieves the compliance requirements associated with selected targets and lists them on the below part of the interface. It also selects the sources to which these listed requirements belong. The user can unselect a compliance source to filter out the requirements that belong to that particular source.



**Figure 12 DCRM - Formulating Compliance Requests**

Once the compliance targets and the requirements that constitute the request to be issued are selected, the user can progress to verification by clicking the ‘Check’ button. At this stage, the DCRM packs this request and forwards it to the Process Verification tools (WP3) by calling a set of web services developed for that purpose. The compliance request that have been issued together with their status can be tracked through the interface shown in Figure 13.



**Figure 13 DCRM - Viewing Compliance Requests**

The status of the verification process can be updated by clicking 'Update Status' button (in Figure 13). Once the verification process completes, the verification results are retrieved from the Process Verification Tools (WP3) by calling back relevant web services and the status of the request changes to 'done'. The analysis results of the compliance requests that are fulfilled (i.e., those requests having status 'done') can be presented to the user through the interface given in Figure 14.

For each compliance target that has been checked, the DCRM gives the overall verification result and lists the controls that have been checked and whether they were satisfied or violated. Together with the controls, corresponding compliance requirements, risks and sources are also presented.

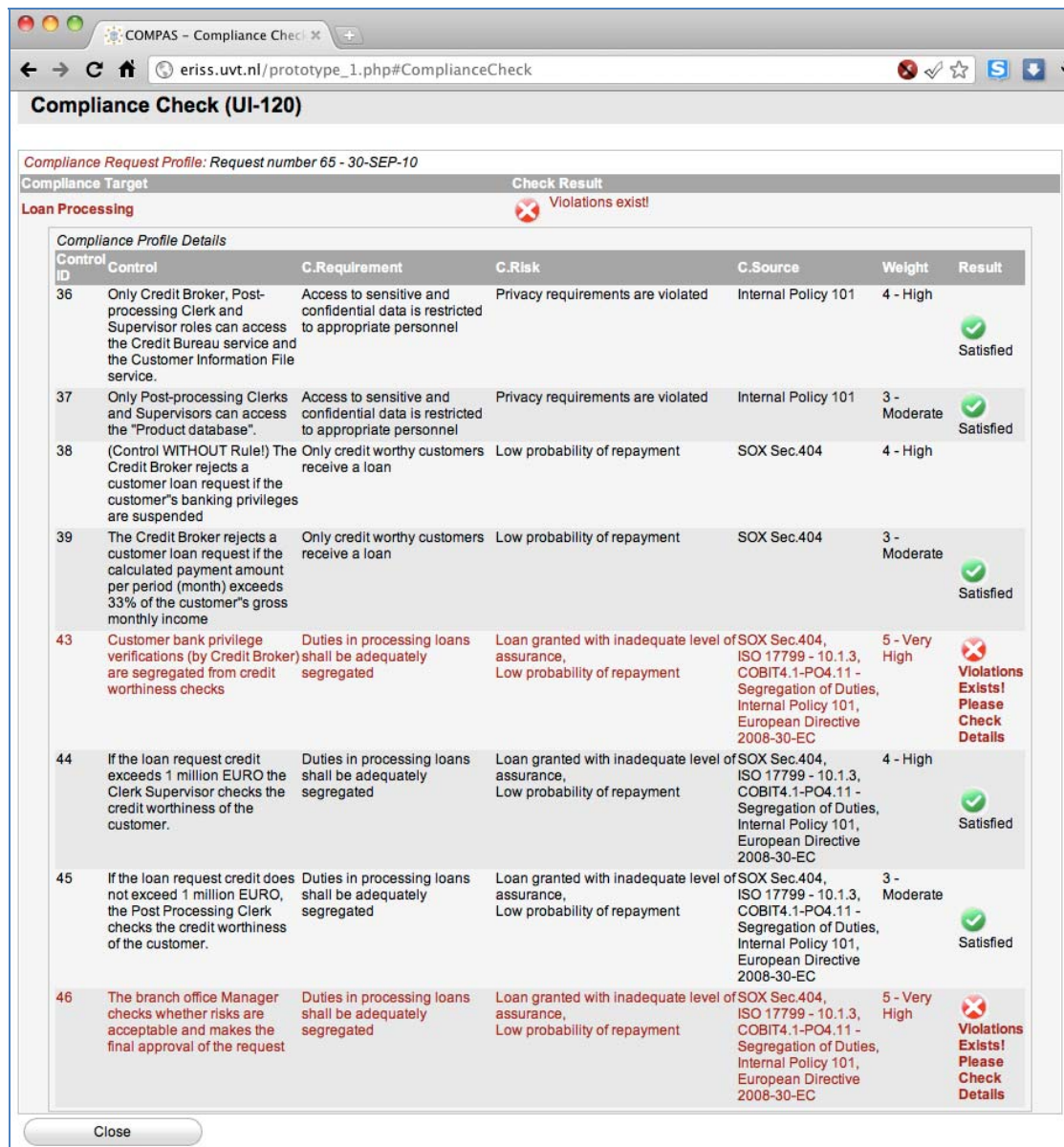


Figure 14 DCRM - Presenting Compliance Check Results

A compliance violation may occur due to a variety of reasons and it is of utmost importance to provide the compliance expert intelligent feedback that reveals the root-causes of these violations and aids their resolution. This feedback should contain a set of rationale explaining the underlying reasons why the violation occurred and what strategies can be used as remedies. The transformation of pattern-based expressions to compliance rules also implicit rule into consideration to make it possible to check and report all possible causes of a violation of a control. Figure 14 gives only an overview of the violations. Clicking on a control in Figure 14 gives detailed information about the compliance rules that have been checked and the remedies or avoidance strategies for those rules that have been violated. Figure 15 depicts this case. The verification results displayed in the screens depicted above are stored in the repository for future reference.

ID	Description.	Statement	Type	Design Time	Run time	Satisfied?	Result Desc. / Remedy
43	Customer bank privilege verifications (by Credit Broker) are segregated from credit worthiness checks						Duties in processing loans shall be adequately segregated Loan granted with inadequate level of assurance, Low probability of repayment SOX Sec.404, 5 - Very High ISO 17799 - 10.1.3, COBIT4.1-PO4.11 - Segregation of Duties, Internal Policy 101, European Directive 2008-30-EC <b>Violations Exists! Please Check Details</b>
38	Check if activity "VerifyBankingPrivilege" exists	F (VerifyBankingPrivilege)	LTL	√		Yes	
39	Check if activity "CheckCreditWorthiness" exists	F (CheckCreditWorthiness)	LTL	√		Yes	
40	Check if activity "CheckCreditWorthiness" precedes activity "VerifyBankingPrivilege"	G(!CheckCreditWorthiness) W VerifyBankingPrivilege.Role(CreditBroker)	LTL	√		Yes	
41	Check if activities are assigned to different roles	G((VerifyBankingPrivilege.Role(Role1) -> G(!CheckCreditWorthiness.Role(Role1)))	LTL	√		No	Check customer bank privilege and check credit worthiness activities should be assigned to different roles.
49	Check if "VerifyBankingPrivilege" leads to activity "CheckCreditWorthiness"	G(VerifyBankingPrivilege.Role(CreditBroker) -> F(CheckCreditWorthiness)	LTL	√		Yes	
50	Check if activities are performed by different actors	G((VerifyBankingPrivilege.Actor(Actor1) -> G(!CheckCreditWorthiness.Actor(Actor1)))	LTL			Not Supported	
44	If the loan request credit exceeds 1 million EURO the Supervisor checks the credit worthiness of the customer.						Loan granted with inadequate level of assurance, Low probability of repayment SOX Sec.404, 4 - High ISO 17799 - 10.1.3, COBIT4.1-PO4.11 - Segregation of Duties, Internal Policy <b>Satisfied</b>

Figure 15 DCRM - Presenting the Details of the Check Results

## 4. Installation of the Components

### 4.1. Web-based Components

Web-based components of the CRLT (CompRM and DCRM) are open to users and can be accessed through 'http://eriss.uvt.nl/compas'. The registration scheme for the site is currently kept 'inactive'. If the installations of the components are deemed necessary the source files of these components supplementing this report can be deployed on any web server<sup>8</sup> with a 'php' processor module (that can be obtained through 'http://www.php.net'). Please refer to the documentation of the web-servers used for the details regarding the deployment of the files.

### 4.2. Rule Modeller Installation

This section presents a brief description of installation procedure of the Rule Modeller standalone component of the CRLT.

<sup>8</sup> Please refer to [http://en.wikipedia.org/wiki/Web\\_server](http://en.wikipedia.org/wiki/Web_server) for the description and list of common web-servers.

**Software requirement:**

- Windows Vista or Windows 7
- Microsoft Visual Studio 2008 (professional, premium or ultimate editions)
- Microsoft Visual Studio 2008 SDK (Software Development Kit)

Note that you should have administrator credentials in Windows Vista/7 to install the Rule Modeller.

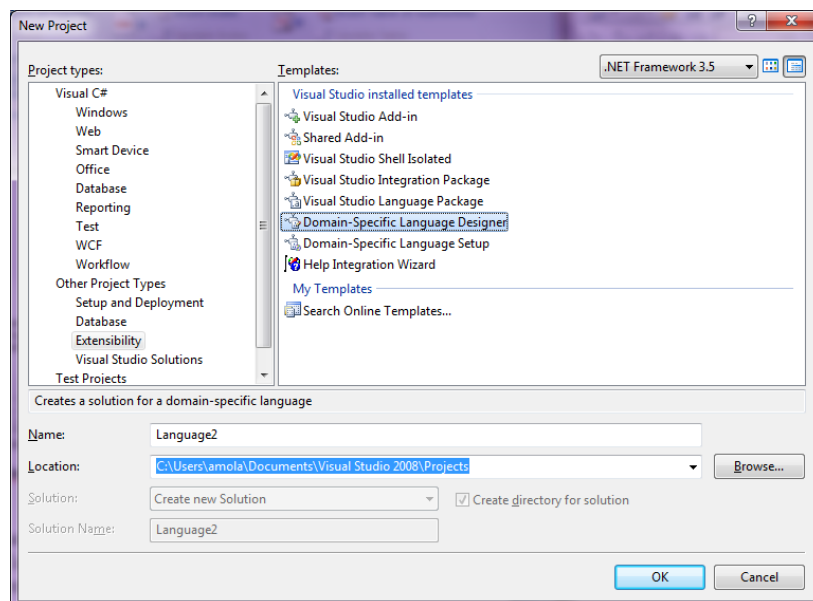
**Installation and Starting Using:**

To install Rule Modeller, click on Rule\_modeler.msi windows installation file located in the main folder where you extracted the prototype. The wizard window shown in Figure 16 opens. Click on 'Next' button, and then follow the installation wizard.



**Figure 16 Rule Modeller installation wizard**

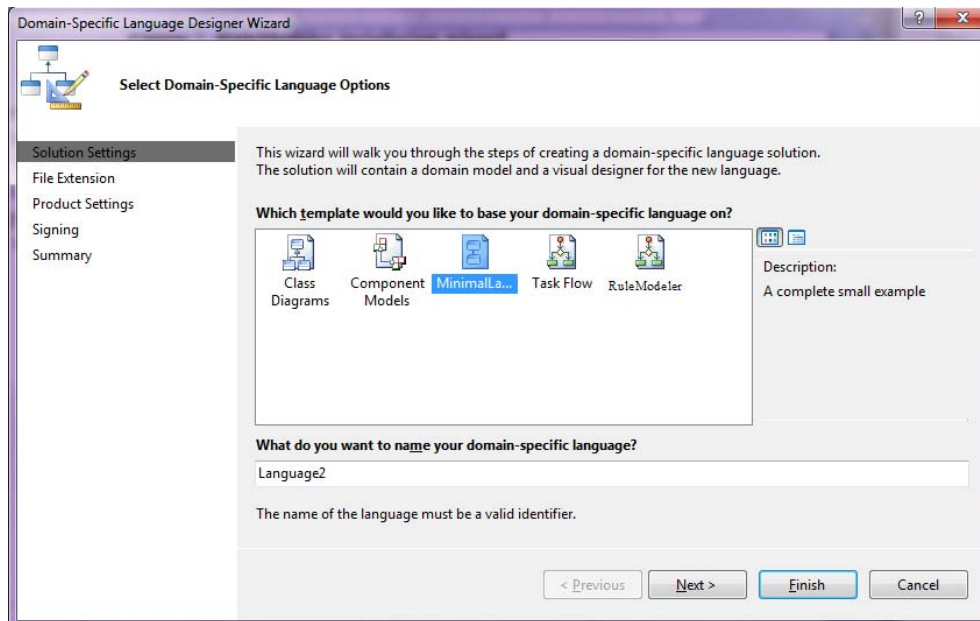
After the installation completes successfully, you can use Rule Modeller as a Domain specific language template, such that it acts as the base of your new project. To do this, open Microsoft Visual Studio, and then click on 'File' menu → New → Project. The 'new project' window shown in Figure 17 appears.



**Figure 17 Rule Modeller - New Project Window**

Click on ‘Other Project Types’ → ‘Extensibility’ → ‘Domain-Specific Language Designer’. Then in the *Name* and *Location* fields, choose a name and location where your new project is stored. Click on ‘Ok’ button. The Domain-Specific Language Designer Wizard appears (Figure 18). Click on ‘Rule Modeller’ in “Which template would you like to base your domain-specific language on?”. Then choose a name for your domain-specific language. Click on ‘Next’ button, and then follow the wizard.

When your new project opens, click on ‘transform all templates’ in the Solution Explorer window, then press on F5 to debug your new project. The Microsoft Visual Studio-Experimental Hive opens, where you can start building new compliance expressions and automatically generate corresponding LTL formulas for verification.



**Figure 18 Rule Modeller - Domain-Specific Language Designer Wizard**

## 5. Reference documents

### 5.1. Internal documents

- [DoW] Description of Work for COMPAS, 2008-02-01.
- [DA.1] COMPAS Architectural Walkthroughs and Evaluation Metrics, 2009-06-08, V2.0
- [D2.1] State-of-the-art in the field of compliance languages, 2008-07-31, V1.0.
- [D2.2] Initial Specification of Compliance Language Constructs and Operators, 2009-06-08, V2.0.
- [D2.3] Design of Compliance Language Run-time Environment and Architecture, 2009-07-31, V1.1.
- [D2.4] Initial implementation of a compliance language, 2010-12-30, V1.0.
- [D2.6] Implementation of an integrated prototype handling interactive user specified compliance requests in a compliance language [M23], 2009-12-17, V1.1.

- [D4.4] Supporting infrastructure – process engine, process artefact repository, process generation tool, 2009-12-22, V1.0.
- [D7.1] “Public Web-Site”, <http://www.compas-ict.eu>

## 5.2. External documents

- [COSO94] COSO Internal Control – Integrated Framework, The Committee of Sponsoring Organizations of the Treadway Commission, 1994.
- [DAC98] M. Dwyer, G. Avrunin, J. Corbett: Property Specification Patterns for Finite-State Verification. *Int. Workshop on Formal Methods on Software Practice*, 1998, pp. 7-15.
- [ETH+10a] A. Elgammal, O. Turetken, W. van den Heuvel, M. Papazoglou: On the Formal Specification of Regulatory Compliance: A Comparative Analysis. *International Performance Assessment and Auditing in Service Computing Workshop, International Conference on Service-Oriented Computing (ICSOC10) Workshops*, USA, 2010.
- [ETH+10b] A. Elgammal, O. Turetken, W-J. van den Heuvel, M. Papazoglou: On the Formal Specification of Business Contracts and Regulatory Compliance. *4th Workshop on FLACOS*. EPTCS, Pisa, Italy, 2010.
- [ETH+10c] A. Elgammal, O. Turetken, W. van den Heuvel, M. Papazoglou: Towards a Comprehensive Design-time Compliance Management: A Roadmap. *15th International Business Information Management Conference (15th IBIMA)*, Egypt, 2010.
- [ETH+10d] A. Elgammal, O. Turetken, W-J. van den Heuvel, M. Papazoglou: Root-Cause Analysis of Design-time Compliance Violations on the basis of Property Patterns. *Proceedings of the 8th International Conference on Service-Oriented Computing (ICSOC10)*, USA, 2010
- [STK+10] D. Schumm, O. Turetken, N. Kokash, A. Elgammal, F. Leymann, W-J. van den Heuvel: Business Process Compliance through Reusable Units of Compliant Processes. *In: Workshop on Engineering SOA and the Web (ESW'10), ICWE 2010 Workshops*, LNCS 6385, pp. 325–337, Springer-Verlag Berlin Heidelberg, 2010.