DA.1

Version:	2.0
Date:	2009-06-08
Dissemination status:	PU
Document reference:	DA.1

COMPAS Architectural Walkthroughs and Evaluation Metrics

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 Universität 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	Universita 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

DA.1 COMPAS Architectural Walkthroughs and Evaluation Metrics

Revision 2.0

Due date of deliverable: 2009-01-31

First submission date: 2009-01-30

Latest submission date: 2009-06-08

Organisation name of lead partner for this deliverable:

TUV Technische Universität Wien, AT

Contributing partner(s):

CWI Stichting Centrum voor Wiskunde en Informatica, NL

PWC, Pricewaterhousecoopers Accountants N.V., NL

TARC-PL Telcordia Poland, PL

THALES, Thales Services SAS, FR

TILBURG UNIVERSITY, Stichting Katholieke Universiteit Brabant, NL

UNITN Universita degli Studi di Trento, IT

USTUTT Universitaet Stuttgart, DE

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

History chart

Issue	Date	Changed page(s)	Cause of change	Implemented by
0.1	2008-11-27	Table of contents, case study walkthrough, initial metrics	New document	TUV
0.2	2008-12-02	Overall architecture figure, technology mapping table	Extensions	TUV
0.3	2008-12-17	Section 3.5	Research metrics for WP3 added	CWI
		Overall architecture figure change, 2.2 and 3.2 update	Extensions	TARC-PL
		Section 3.6, typo fixes	Extensions	USTUTT
0.4	2008-12-18	Whole document	Merging and revising	TUV
0.5	2008-12-18	Figure 1	Update	TARC-PL
		Section 2.1	Extensions	PwC
		Section 3.7	Extensions	TRENTO
0.6	2008-12-28	Whole document	Revising	TUV
0.9	2009-01-20	Document's name,	Changing document's name, updating related cross-references,	TUV
0.95	2009-01-21	2.2,3.1	Modifications, Fill in section 3.1	THALES
0.96	2009-01-22	3.2	Revising	TARC-PL
0.97	2009-01-26	Fig. 1	Revising	TUV, TILBURG
	2009-01-26	3.6	Revising	USTUTT
0.98	2009-01-27	Section 3.7	Revising	UNITN
1.00	2009-01-28	Whole document	Finalizing	TUV
1.10	2009-05-08	Whole document	Revise according to reviewers' comments	TUV
2.0	2009-06-08		Approve & Release	TUV

Authorisation

No.	Action	Company/Name	Date
1	Prepared	TUV	2008-12-15
2	Approved	TUV	2009-01-28
3	Released	TUV	2009-01-28
4	Prepared	TUV	2009-05-08
5	Approved	TUV	2009-06-08
6	Released	TUV	2009-06-08

Contents

List of figures

Abstract

This deliverable introduces a pragmatic view of COMPAS architecture introduced in [D1.1] in terms of architectural instantiation and walkthroughs in the course of two use case scenarios in COMPAS (cf. [D6.1]). Finally, the Evaluation Metrics, introduced in D6.1 will be concretised and explained for this architecture – especially in terms of tangible results for the research work performed in COMPAS.

1. Introduction

1.1. Purpose and scope

A prototype of a business compliance software framework based on the model driven development paradigm is presented in [D1.1]. The purpose of this deliverable is, on the one hand, to illustrate the usability of the architecture of this framework, and on the other hand, to elaborate evaluation metrics for COMPAS work packages.

1.2. Document overview

The deliverable is organised as following. Section 2 gives an overview of the whole COMPAS architecture along with details on instantiating the architecture in the course of two industry use cases. Evaluation metrics are subsequently presented in Section 3.

1.3. Abbreviations and acronyms

BPEL	Business Process Execution Language
DSL	Domain Specific Language
ESB	Enterprise Service Bus
ETL	Extract, Transform and Load
MDSD	Model-Driven Software Development
QoS	Quality of Service
VbMF	View-based Modelling Framework

2. Architecture Instantiation and Walkthrough for the Use Cases

In this section, we align COMPAS architecture (cf. [D1.1]) that provides prototypes, tools, and technologies, with the use cases provided by industry partners in order to demonstrate the pragmatic use of COMPAS contributions. For the sake of simplicity and illustrative purpose, the components of the detail architecture [D1.1] are adequately grouped according to the relevance of their functionality (see Figure 1). In next subsequent sections, the architecture, shown in Figure 1, is instantiated in the course of two COMPAS use cases (cf. [D6.1]).

Detailed functionality, status and corresponding WP responsibility of these components are then provided in Table 1. Every row of Table 1 describes detail of one component in the COMPAS architecture. The first column is the component's name. Next, the second column introduces functionality of these components, respectively. Each component might have one or more interactions with others. These interactions are clarified in the third column in which the corresponding component is aligned with expected inputs from its adjacent components. The forth column mentions specific technologies or tools used to realise the component as well as their status in context of COMPAS project, such as **New** (i.e., develop from scratch), **Extend** (i.e., extend existing tools or technologies), and **Use** (i.e., use existing tools or technologies without extending). The last column names the WPs involving in the component.

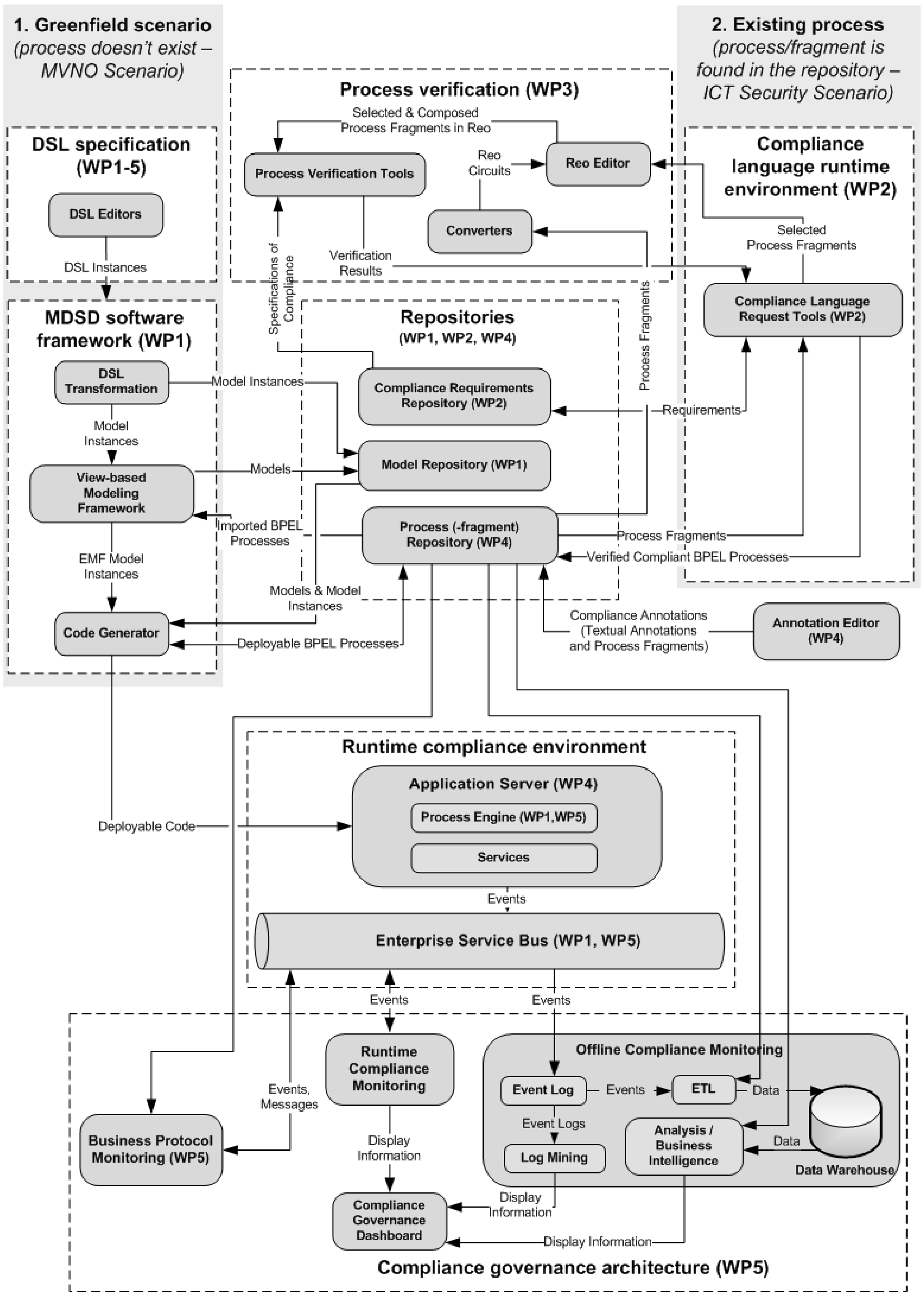
1. Greenfield scenario
(process doesn't exist – MVNO Scenario)
2. Existing process
(process/fragment is found in the repository – ICT Security Scenario)
DSL specification (WP1-5)
DSL Editors
DSL Instances
MDSD software framework (WP1)
DSL Transformation
Model Instances
View-based Modeling Framework
EMF Model Instances
Code Generator
Process verification (WP3)
Selected & Composed Process Fragments in Reo
Process Verification Tools
Reo Circuits
Reo Editor
Converters
Verification Results
Specifications of Compliance
Process Fragments
Compliance language runtime environment (WP2)
Selected Process Fragments
Compliance Language Request Tools (WP2)
Repositories (WP1, WP2, WP4)
Model Instances
Compliance Requirements Repository (WP2)
Requirements
Models
Model Repository (WP1)
Imported BPEL Processes
Process (-fragment) Repository (WP4)
Process Fragments
Verified Compliant BPEL Processes
Models & Model Instances
Deployable BPEL Processes
Compliance Annotations (Textual Annotations and Process Fragments)
Annotation Editor (WP4)
Runtime compliance environment
Application Server (WP4)
Process Engine (WP1,WP5)
Services
Deployable Code
Events
Enterprise Service Bus (WP1, WP5)
Events
Events
Events, Messages
Business Protocol Monitoring (WP5)
Runtime Compliance Monitoring
Display Information
Compliance Governance Dashboard
Offline Compliance Monitoring
Event Log
Events
ETL
Data
Event Logs
Log Mining
Analysis / Business Intelligence
Data
Data Warehouse
Display Information
Display Information
Compliance governance architecture (WP5)

Figure 1 Overall COMPAS architecture

Component	Functionality description	Relationships to other components	Tools and technologies	Status	Work-Package
Analysis / Business Intelligence	The reasoning mechanisms and algorithms should analyse the root causes of deviations of the process constellations and fragments from the desired compliance concern targets, as well as it automatically identifies such deviations. A meta-model and DSL for compliance to security policies will be developed in this task to show-case and validate the results of this task for a technical governance concern, which will be analysed using the business process intelligence suite.	**Data Warehouse** Analysis/Business Intelligence component acts as both producer and consumer of data: it analyses data in the warehouse and stores the results back into the warehouse. **Process (-fragment) Repository** The Repository provides the Analysis/Business Intelligence component with Processes (or -fragments) that comprise relevant annotated compliance meta-data in order to support compliance-oriented analysis and interpretations.	BusinessObjects, SpagoBI	Extend	WP5
Annotation Editor	The annotation editor is a text-based editor that allows annotating artefact with other artefact. In the case of COMPAS we foresee the annotation of processes with both textual annotations and with process fragment, as well as the annotation of process fragments with textual annotations.		Text Editor	New	WP4
Application Servers	The application server is the runtime environment for components such as the process engine and the services. It is often also referred to as container.	**Code Generators** Deployable artefacts (e.g., generated and annotated processes, services, runtime configurations, etc.,) are deployed to	Apache Tomcat, Axis2 support	Extend	WP4

		process engines and application servers for execution.			
Business Protocol Monitoring	Architecture for runtime monitoring of Web Services conversations. It provides basic events as detection of compliance violation at messages level.	**Process (-fragment) Repository** The Business Process Protocol Monitoring component retrieves BPEL processes and BPEL process fragments in order to monitor business process execution. **ESB** Business Protocol Monitoring component listens to messages exchanged between Services through the ESB and publishes some events related to compliance checking into the ESB.	MS Visual Studio/ Eclipse plug-in	New	WP5
Code Generator	The Code Generator takes as inputs the modelling artefacts validated via the Model Validator, and a number of transformation templates used to produce. Then, it might perform model validations against required constraints (if any). Finally, schematic code and configurations are generated. The generated schematic code might be augmented with individual code specialized for specific business logics, particular platform features, etc.	**Model Repository** The Code Generators takes models and model-instances from the Repositories as well as templates needed for transforming model-instances to code. **Process (-fragment) Repository** The Code generator retrieves non-executable BPEL processes from the Process (-fragment) Repository and generates execution information, e.g.	openArchitectureWare	Extend	WP1

		WSDL-file, deployment descriptor etc. and stores the executable BPEL process in the Process (-fragment) Repository. **View-based Modelling Framework** View model and view instances in VbMF can be directly used by the Code Generator for generating code.			
Compliance Governance Dashboard	The Dashboards are a user friendly graphical web based visualisation of compliance information, particularly compliance violations of business process.	**Runtime Compliance Monitoring** Displays the visualised monitoring results to the user. **Log Mining** The Dashboard is fed up with analysed and interpreted results from Log Mining to display to the user. **Analysis/Business Intelligence** The analysis/business intelligence component will compute compliance indicators based on the data in the warehouse and identify (where possible) root causes of violations. Such results are displayed in the dashboard	Graphical Web UI Dashboard	New	WP5

Compliance Request Language Tools	Compliance Request Language (CRL) is a proposed formal language that can be utilised for the specification and representation of compliance requirements and introduce an initial specification of compliance language, along with its associated constructs and operators (extracted from D2.2). CRL Tools aim at supporting the user effectively making use of the CRL language.	**Process (-fragment) Repository** The Compliance Request Language Tools retrieve BPEL process fragments and BPEL processes stored in the Process (-fragment) Repository by querying the Process (-fragment) Repository employing a request language defined and specified in WP2. Moreover the Compliance Request Language Tools store verified compliant BPEL processes and process fragment compositions in the Process (-fragment) Repository. **Compliance Requirements Repository** The Compliance Requirement Repository stores and organises compliance requirements at various abstractions levels (in terms of goals, policies and rules) and allows the reusability of the compliance constraints (extracted from D2.2) **Process Verification Tools** Verified process models are stored in the Process (-fragment) Repository and can be queried by means of the Compliance Request Language Tools.	Graphical LTL tools	Extend	WP2

		Compliance Governance Dashboard Feedbacks from the Dashboard are needed by the Compliance Request Languages Tools to help the user to adjust relevant compliance requirements and specifications.			
Converters	Three converters, namely, BPMN2Reo, UML2Reo and BPEL2Reo, are Eclipse plug-ins used to convert business process models to Reo circuits, and subsequently, to constraint automata, for their formal analysis and compliance verification.	**Process (-fragment) Repositories** BPEL process fragments are retrieved from the Process (-fragment) repository are automatically converted to Reo process models for their further composition, refinement, verification and compliance analysis using Reo Editor and Process Verification Tools.	Reo, Eclipse	New	WP3
Data Warehouse	A Data warehouse is an integrated, non-volatile, historical, subject-oriented data collection, aimed at supporting decision-making processes. Particularly, in COMPAS the DW stores compliance and process related data.	**ETL** The data warehouse is the destination of the data processed by the ETL procedures. It stores transformed events in form of process, activity, and service instance facts.	Project-specific data warehouse model DBMS	New Use	WP5
DSL Editor	System requirements and compliance concerns are represented in terms of Domain-Specific Languages (DSLs). DSLs describe knowledge via a graphical or textual syntax. Therefore, DSL editors are necessary for, and often used by stakeholders to manipulate the requirements.		Eclipse-based editors, XML Editors	Extend	WP1
DSL	DSL transformations take as inputs the DSLs	**DSL Editor**	Frag, XML	Extend	WP1

Transformation	defined by using DSL editors, and then, interpret and transform them into modelling artefacts such as models, model instances and/or relevant constraints (if any).	DSL instances are passed from DSL Editors to the DSL Transformation	Parsers, Parser Generators		
ESB (Enterprise Service Bus)	The Enterprise Service Bus (ESB) is a unified communication channel between the components. Besides that the ESB provides a publish/subscribe mechanism for events that are published via this channel. In COMPAS the ESB is employed for publish subscribe mechanism for events, without implementing additional functionality.	**Application Servers** The runtime components such as the process engine executing processes, and the services describe their current status by generating and emitting events. Those events can be published via the ESB in order to inform any interested component, such as the Event Log (described below) for example. **Business Protocol Monitoring** Described above **Runtime Compliance Monitoring** Runtime Compliance Monitoring is an event subscriber of the ESB. Relevant events published by the ESB are exploited by the Runtime Compliance Monitoring for online detection of compliance violations.	ServiceMix, ActiveMQ	Use	WP2,4,5
ETL (Extract, Transform, and Load)	The ETL (extract, transform and load) processes are responsible for the extraction of the data from the Event log (possibly from Audit trail), transforming them according to	**Process (fragment) Repository** Process models will be used during ETL for the identification of executed process	Project-specific ETL procedures	New	WP5

	the DW data model.	instances out of logged events. Process fragment models can be used to check whether a specific process instance obeyed to a given fragment or not. **Event Log** The event log represents the actual data is processed by the ETL procedures. Event will be aggregated in order to reconstruct process, activity, and service instances.	Talend (or similar)	Use	
Event Log	An Event log can be seen as a set of past events typically ordered chronologically by the timestamps. Depending on the implementation the event log normally provides an interface to retrieve a certain sub-set of those events that occurred within a given interval. In the architecture of COMPAS the events are emitted from any component in form of messages, which can be delivered by the ESB that again provides publish/subscribe functionality for any component that is interested in a certain type or source of event.	**ESB** The Event Log uses the Publish-Subscribe mechanism that the ESB provides, for subscribing to, and retrieving any event that is required for further processing, such as for compliance analysis in the Data Warehouse.	DBMS	New	WP5
Log Mining	Log mining refers to the activity of extracting implicit knowledge from log repository. For instance, the actual business protocol of the process can be retrieved, allowing a better understanding of service and clients behaviour. In the architecture of COMPAS, knowledge from log mining is reported	**Event Log** The Log Mining uses subsets of events provided via the Event Log's interfaces to reason and produce relevant knowledge that are displayed to the user via the Dashboard.	Java, Matlab	New	WP5

	through the monitoring dashboard.				
Process Engine	The process engine is the component that executes processes by navigating through the steps defined in the process model, based on the current parameters of the running process instance. A process engine describes the current status of the execution of processes by generating and emitting execution events. During execution of a business process instance the engine stores additional execution data in the audit trail e.g., incoming purchase order, and emits events to the ESB, which is used for reliable messaging and Publish-Subscribe mechanism for event messages		Apache ODE	Use	WP1,5
Process Verification Tools	Process Verification tools include a Reo animation plug-in and a Vereofy model checker. Reo animation plug-in is a tool that generates flash animated simulations of formal business process models. The plug-in depicts the process that was previously shown in the Reo editor in the animation view. The parts of the process highlighted red represent synchronous data flow. Tokens move along these synchronous regions. On the left side there is a list of possible animations for this connector and the attached writers and readers. Reo validation plug-in is a tool that performs model checking over coordination models	**Reo Editor** Process Verification Tools take as input format constraint automata automatically generated from Reo process models. **Compliance Requirement Repository** Compliance requirements are expressed as logic formulas and used as input for the model checker.	Eclipse plug-in	Extend	WP3

	represented as constraint automata. This model checker uses a symbolic model and LTL logic as property specification formats.				
Reo Editor	The Reo editor is an Eclipse plug-in that enables business process modelling by simple drawing operations and serves as a bridge to a number of other tools that can be either invoked from the context menus or directly interact with it. Formal business process models are stored using an XML format and can be further verified and transformed to service compositions by wiring appropriate web services.	**Converters** Reo process models can be automatically obtained from BPMN/UML diagrams (green field scenario) or BPEL process fragments with the help of corresponding converters.	Reo, Eclipse	Extend	WP3
Repositories	The Repositories provides means for registering, persisting, and versioning modelling artefacts in order to enhance reusability and collaborative development. There are three main types of repositories including Compliance Requirement Repository, Model Repository and Process (-fragment) Repositories. • Compliance Requirement Repository • Process (-fragment) Repository. • Model Repository	**DSL Transformation** Model instances produced by the DSL Transformation are stored in the Model Repository for later use. **Compliance Request Language Tools** The Compliance Request Language Tools retrieve BPEL process fragments and BPEL processes by querying the Process (-fragment) Repository employing a request language defined and specified in WP2. Moreover the Compliance Request Language Tools store verified compliant BPEL processes and process fragment	DBMS as backend	New New New	WP2 WP4 WP1

		compositions in the Process (-fragment) Repository. **Annotation Editor** The Annotation Editor is employed for annotating BPEL processes with textual annotation as well as process fragments for defining and specifying compliance constraints. Besides the BPEL processes and BPEL process fragments are annotated with meta data, e.g. information about application domain. For details see [D4.1] **Runtime Compliance Monitoring** The Runtime Compliance Monitoring component retrieves BPEL processes and BPEL process fragments in order to do near-real time monitoring of business process execution. **Code Generator** Schematic process code generated from the Code Generator can also be stored in the Process (-fragment) Repository. Conversely, processes and process fragments in the repository can be queried			

		and re-used within the Code Generator.			
Runtime Compliance Monitoring	Software architecture for online detection of compliance violations. It's based on complex event processing concepts and provides immediate notification of detected violations.	**Process (-fragment) Repository** The Runtime Compliance Monitoring component retrieves BPEL processes and BPEL process fragments stored in the Process (-fragment) Repository in order to do near-real time monitoring of business process execution. **ESB** Described above	CEP engine, Eclipse	New	WP5
View-based Modelling Framework	View-based Modelling Framework acts as a modelling foundation for representing different process concerns by exploiting the concept of architectural views. Process concerns, such as the control-flow, service invocations, data handling, etc., are modelling artefacts. These artefacts might be bound to some modelling constraints, or be associated with meta-data of some compliance concerns.	**DSL Transformation** The DSL transformation parses the DSL instances and produces model instances that can be manipulated by the View-based Modelling Framework. **Process (-fragment) Repository** The View-based Modelling Framework imports BPEL processes from the Process (-fragment) Repository, transforms them into EMF-models and stores these models in the Model Repository.	EMF, Frag	Extend	WP1

Table 1 Mapping of COMPAS components into prototypes, tools and/or technologies

2.1. Use Case Selection and Details

In the COMPAS meeting on November 24-25, 2008, in Lyon, the COMPAS partners agreed to change the role of partner PwC. Instead of providing a use case scenario, PwC will change its role in WP6. From the perspective that PwC is a subject matter expert in the field of compliance and auditing PwC's role is twofold:

- On the one hand PwC will support WP6 partners TARC-PL and THALES in defining the compliance requirements and the evaluation metrics for their use case scenarios.
- On the other hand PwC will have a profound role in evaluating the pilot of the use case scenarios based on the defined evaluation metrics and compliance requirements. In other words PwC will help in evaluating whether COMPAS will be able to assess the compliance requirements and report on this according to the evaluation metrics.

The case studies presented in this section are two scenarios which are chosen for implementation. They represent different kinds of compliance concerns:

- Security related compliance concerns (authorisation, privacy, etc.) – ICT Security Scenario
- Licensing related compliance concerns – Mobile Virtual Network Operators Scenario (MVNO)

The specific architecture instantiations and walkthroughs are provided in the following subsections. They refer to the architecture presented in Figure 1 and instantiate two different flows through the diagram. ICT Security Scenario assumes that the processes already exist, while MVNO Scenario is developed from scratch with the use of modelling and DSL transformation tools. Both scenarios are thoroughly monitored in the course of their executions by online, and offline monitoring components such that compliance violations will be readily detected and processed accordingly. Therefore, each of the implementations will serve to test different software frameworks created in the course of the COMPAS project.

The other scenarios presented in [D6.1] will serve for research evaluation only.

2.2. ICT Security Scenario: Architecture Instantiation and Walkthrough

In this section, we provide a description of a possible architecture instantiation and a walkthrough of the COMPAS architecture for the ICT Security Scenario described in [D6.1].

This scenario assumes we could start from either an existing application or an application being assembled from existing software components (coming from achievements to date). The approach here for Thales comes up with a realistic (i.e. relevant from the business perspective) use case in order to assess, from an operational point of view, the COMPAS resulting technology and framework. The rationale is that the perfect Use Case used to assess COMPAS resulting doesn't exist as a whole, but it has to be assembled based on existing software components in the business domain under consideration.

That means, business processes already exist, but they are not yet modelled as BPEL processes or process fragments – or in *COMPAS DSLs*. The first step hence is to model the BPEL processes using the DSLs for process design from the *View-based Modelling Framework* (VbMF) or using a publicly available BPEL editor, such as the Eclipse BPEL editor [EcBPEL]. In the later case, the VbMF import tooling can be used to import the BPEL processes and generate VbMF process models such as the flow view model, high-level view models for collaboration, information concerns, and low-level view models for collaboration, information, transaction, event handling concerns, from it. The process models and model instances are stored in the *Model Repository*. The BPEL processes generated from the model instances are stored in the *Process (-fragment) Repository*.

Next, using the *Annotation Editor* along with the BPEL editor, we can develop process fragments and annotate these fragments with appropriate compliance meta-data to properly enforce compliance with the specific regulations, norms, and standards identified in the ICT Security Scenario. Process fragments can be identified by dividing existing processes into compliance concerns and basic concerns, in case the existing processes already contain compliance concerns. Many of the compliance concerns, identified in the ICT Security Scenario described in [D6.1], have not been implemented as process before. Then we need to implement new process fragments for the compliance concerns.

Both the fragments and the processes that have been modelled can be verified using the *Process Verification Tools*. To perform verification, they must be imported and modelled in the *Reo Editor*.

The compliance metadata model from *VbMF* is used to represent the compliance metadata, such as to which regulations, norms, and standards a fragment or process is compliant. This information is given using the compliance metadata DSL, and it can also be added in the *Annotation Editor* to the process fragments – as textual annotations. This information is also stored in the *Model Repository*.

The *Compliance Requirements Repository* contains further information: the compliance requirements in terms of a goal-oriented model. This information and the set of fragments in the *Process (-fragment) Repository* are used in the *Compliance Request Language Tools* to enable users to find compliant fragments. That is, once fragments have been designed, we could query the repository for compliant fragments before we newly design it. If compliant fragments exist, we can reuse them in our processes. That is, we have to compose the process with these fragments. The composition is a new process that needs to be verified again, before it can be stored in the *Process (-fragment) Repository*. From the *Process (-fragment) Repository* we can also import the composed process into the VbMF models. The *Compliance Request Language Tools* can use the feedback from the verification tools to provide another or a better result in later queries.

At the end of the process modelling, all the models of business processes and fragments, as well as the compliance metadata models are stored in the *Model Repository*.

Some security information, such as access control rights, credentials, or role information are usually not realized in BPEL processes but hosted in backend systems. Such security information needed for representing the ICT Security Scenario is implemented in the Security DSL. From the information in this DSL, we generate services to access the security functions from the BPEL processes.

All models are represented using a high-level, domain-oriented and a low-level, technical view. For implementing the existing processes, only technical experts are needed and they work using the low-level, technical views. The high-level, domain-oriented views can be used to communicate the models with domain experts. This is done to validate that all processes are correctly translated into the COMPAS models, and to model together the newly identified compliance concerns that have not been addressed in the previous system implementation.

Once the first complete system version is modelled, we can use the generator to generate deployable code. This code is deployed on the process engines and application servers. During generation, the generator's transformation templates add event-creating functions to the generated code that are raised for actions such as service is invoked, process is started, process activities is invoked, process has ended, and so on.

All events are sent to the *ESB*, which has a number of event subscribers. First, there are two online monitoring tools. The *Business Protocol Monitoring* allows developers and domain experts to check, whether the events created adhere to the business protocols that are expected to be compliant. If this is not the case, an error message can be raised. Secondly, the *Runtime Compliance Monitoring* allows us to monitor events as they happen, and if compliance is violated, we can report the violation.

In addition to online monitoring, offline monitoring must happen because some compliance concerns require longer event traces and more computational resources to be detected than what is easily possible in an online monitoring (aka real-time scenario). Hence, the *ESB* feeds the *Event Log* with events, which is extracted using an *ETL* component into a *Data Warehouse*, which *Analysis/Business Intelligence Tools* analyse for compliance violations. Finally, *Log Mining* techniques deliver direct information. All this compliance information is visualised in a *Compliance Governance Dashboard*, which shows the users all relevant compliance information.

Using the *Model Repository*, the Dashboard has all metadata and information about the compliance concerns. For instance, it can show the process/service relationships of a compliance concern or the compliance metadata (regulations, norms, rules, policies, standards, etc.) it relates to. Hence, drill down and root cause analysis are supported.

2.3. Mobile Virtual Network Operators Scenario: Architecture Instantiation and Walkthrough

The Mobile Virtual Network Operators Scenario instantiates the COMPAS architecture in a similar way to the ICT Scenario. However there are two major differences - one involves process creation and the other one concerns monitoring.

The Mobile Virtual Network Operators Scenario is a Greenfield scenario. No processes exist. That is, we will first design high-level business processes with the domain experts, which can be modelled using the *Reo Editors* and verified. Next we use the *DSL Transformations* to translate the processes into technical process models using *VbMF*. These are stored in the *Model Repository* and used as a basis for generation.

Again, not all compliance concerns can be expressed in the business processes or process fragments, but some must be implemented as generated services. While the description of the case study in D6.1 presents various possible scenarios including MVNO related services, this scenario will focus mostly on compliance concerns related to Licensing (which also includes QoS). Various licenses will be defined using Licensing DSL described in [D5.2].

Concepts related to authentication or privacy (also mentioned in MVNO scenario description in [D6.1]) will not be implemented, as it's already explicitly covered in ICT Security Scenario.

In contrast to the previous scenario, in which offline monitoring has played an important role – to detect security violations – in this scenario the *Runtime Compliance Monitoring* is most important. That is, violations of various licenses must be monitored while the system runs (e.g. in case when QoS constraints are violated). Proper notifications have to be sent immediately to the *Compliance Governance Dashboard*. Eventual adaptation of the processes would then be performed manually by people responsible for it.

In this scenario, the *Compliance Request Language Tools* are not used – as it is a Greenfield scenario and a basis of process fragments is needed in order to make the use of the *Compliance Request Language Tools*. However, we can validate that after the scenario has been implemented, the *Compliance Request Language Tools* can be used for reuse.

3. Evaluation Metrics

In this section, we present more a detailed view on the metrics for the COMPAS project. In particular, we firstly present *business metrics* for the two case studies described above. That is, these metrics focus on evaluating the COMPAS results from the point of view of the business case of the respective case study. Secondly, we present *research metrics* for all research work packages (WP1-WP5). These metrics focus on evaluating the research results – as far as possible – in a tangible manner.

3.1. Revisited Metrics for the ICT Security Scenario

Thales will assess and value COMPAS Project results in the context of the ICT Security Scenario which has been reported in [D6.1]. Among other application domains (including eGov domain as reported in [DoW]) the final application domain to validate COMPAS results was set to Banking domain due to its Business relevance and its attainability in view of Background and Foreground which are the ones of ThereSIS in the field.

As for metrics Thales will apply to fully assess relevance of the resulting COMPAS compliance technology and framework they are twofold:

- First, we focus on metrics and/or criteria to assess the technical/technological relevance of the work performed (e.g. SOA compliance extensions) through contribution to NESSI, NESSI WGs and NEXOF(-RA). As for the latest we will quantify and qualify the contribution of COMPAS as a NESSI Project to NEXOF(-RA).
- Second, we focus on metrics and/or criteria to assess business relevance of the COMPAS Compliance framework.

As for concrete examples of metrics and/or criteria we may use we'd like to stress the following ones (this in complement of the ones already stated in [D6.1]), some of them being shared with other WPs:

- Level of reference reached (in terms of Reference Model and Reference Architecture)
- Usability of the SOA Compliance Framework/Platform
- Level of automation offered
- Level of expertise required
- Level of reporting offered (online and offline)
- Level of expressiveness of Compliance Request language,
- Level of genericity of the Security DSL
- Level of standardization reached

- Level of interoperability with other SOA platforms/frameworks
- Level of extensibility offered
- Level of relevance of Proof-of-Concepts (focus being on validation of Architectural choices and patterns and their relevance from a technical/business/societal perspective
- Level of re-use offered through the various repositories
- Level of ROI offered

3.2. Revisited Metrics for the Mobile Virtual Network Operators Scenario

In comparison to the metrics listed in [D6.1] for MVNO Scenario, the quantitative metrics used for evaluation of implemented system prototype will include:

- number of compliance violations which occurred,
- number of successful detections of violations in COMPAS framework,
- statistics related to the effectiveness of the compliance violation detection,
- average time required to detect a compliance violation (real-time monitoring).

Specific test cases will be prepared for each type of compliance violation identified in the scenario for the purpose of assessing these measures.

Qualitative metrics for the scenario might also be included and could reflect the added value of COMPAS in detection of the compliance violations related to licensing.

3.3. Research Metrics for WP1

WP1 is mainly concerned with the model-driven integration architecture of COMPAS, as well as means for modelling and expressing the models in DSLs. Unfortunately, research questions related to the quality of a model or of an integration solution can hardly be expressed in terms of quantifiable metrics. We hence will use a qualitative approach to assess the WP1 results. The basic metrics for WP1 are related to the realisation of the COMPAS integration infrastructure:

- Can the main integration solutions (model-driven generator, transformations, and DSL tools) successfully be realised in terms of prototypes? We need to check if we can deliver all prototypes described in the WP1 description (see [DoW]).
- Can the models, meta-models, and model transformations described in the WP1 description (see [DoW]) be realised? Can they express the process-driven SOA runtime technologies used in COMPAS, such as BPEL, Web Services, etc.?
- Can a DSL tooling be developed and well integrated with the COMPAS integration solutions, as well as the models, meta-models, and model transformations?

Next, we need to check, whether the hypothesis that this integration infrastructure is usable for compliance concerns is valid:

- Can all compliance concerns identified in WP6 case studies [D6.1] prototypically be depicted using models and DSLs?
- Can all compliance concern models be integrated with the models developed in WP1?

- Can all compliance concerns identified in WP6 be mapped to runtime components for representing and monitoring them?

All metrics mentioned so far can be seen as binary metrics: Either it has been achieved or not. In addition, we can perform a comparison to the related work to assess the quality of the solution in comparison to related work.

3.4. Research Metrics for WP2

The main concern of WP2 is to develop a language for service user requests and service provider constraints, and a language of regulation compliance concerns including the supporting infrastructure that helps users formulate service requests and relevant requirements and identifying suitable matching service. WP2 is responsible for: (i) developing compliance language request tools where compliance concerns can be formally specified at various levels of abstraction to accommodate different stakeholders' needs, (ii) and the development of compliance requirement repository, where compliance requirements can be stored, organized and managed at various levels of abstraction. According to these goals, the contribution of WP2 can be assessed with the means of the following metrics:

1. Compliance language request tools:
   - *Formality:* Are the compliance language request tools formally-based paving the way for further automatic analysis, reasoning and validation techniques?
   - *Expressiveness:* Is the compliance request language expressive enough to capture the intricate semantics of compliance requirements. More specifically, can all compliance requirements identified in WP6 case studies [D6.1] be formally represented using the compliance language request tools?
   - *Usability:* Are the compliance language request tools easy to be used and understood by users?
   - *Consistency checking:* Do the compliance language request tools provide a mechanism for checking the consistency between a set of selected compliance rules?
   - *Declarative*: Is the compliance request language declarative?
   - *Generic*: Can the compliance request language capture the semantic of compliance concerns covering the basic categories of compliance concerns (control flow, locative, informational, temporal and resource) identified in D2.1?
   - *Monotonic*: Do the compliance language tools support the specification of non-monotonic rules?

2. Compliance requirements repository:
   - Can compliance requirements be organized at various levels of abstractions accommodating the needs of different stakeholders, where the relationships between these levels are managed and maintained?
   - Can the user easily browse and query the repository to locate compliance concerns at various levels of abstractions? This can be assessed using questionnaire over a number of users.

3.5. Research Metrics for WP3

The primary concern of WP3 is to design formal behavioural models for specifying business processes and Web Service compositions as well as to provide tools for their automated analysis and verification against formally expressed compliance regulations. With respect to these goals we can evaluate the contribution of WP3 using the following metrics:

- **Level of automation** can be measured as the amount of automated vs. manual model transformations required to enable formally verified business process development.
- **Level of coverage** can be defined as a fraction of compliance requirements against all compliance requirements relevant to each case study verified automatically using behavioural models and model checking tools developed in WP3.
- **Model expressiveness.** Fraction of business process aspects (control flow, data flow, time properties, performance, resources) and compliance requirements that can be expressed using behavioural models developed in WP3.
- **Extensibility.** A binary metric that reflects the ability to extend models and tools developed in WP3 with new functionalities.
- **Interoperability.** A binary metric that reflects the ability to integrate the models and tools developed in WP3 into various software (re-)engineering scenarios and work with other business process and service composition development tools.
- **Usability.** Usability can be seen as a compound metric that reflects suitability of the tools developed in WP3 to the needs of industrial partners. We assume that it can be measured using questionnaires aiming to understand whether the developed models are intuitively understandable while their supporting tools provide sufficient functionalities and have been helpful in the implementation of COMPAS case studies.

3.6. Research Metrics for WP4

The planned research work in WP4 addresses the reduction of development complexity by providing the classification and specification of reusable business process fragments augmented with compliance concerns, which can be composed using a model-driven software framework.

The assessment criteria that we want to propose as research metrics thus mainly concern the life-cycle of business process fragments - from modelling to execution and finally to monitoring. The application of the concepts in the industry case studies could be seen as a cross-cutting factor for evaluation, that affects all of the criteria listed in the following, not only those where the case studies are explicitly mentioned. The criteria for the supporting infrastructure can in addition be seen as a proof of the concept of the performed research work.

- Classification and specification of process fragments
  - Modelling:
    - In how far does the concept of process fragments support the reuse in the field of processes and service compositions? Is there, beyond the scope of COMPAS, additional tooling or research work required for leveraging the concept?

        - What are the limitations of this approach, and can they be measured? Can problem-classes be classified, for which this approach is not applicable?
        - Are process fragments feasible for modelling the compliance concerns specified in the case studies? In case not, or just partially, what are the other possibilities and alternatives?
    - Execution:
        - Can process fragments be executed on a process engine? Are there extensions or modifications of the engine necessary? Which are those extensions?
        - What is the event model that is required to cover compliance? Is a process engine already capable to generate all those events during execution? Is such event model limited to the field of compliance, or is it also applicable in other domains?
    - Monitoring:
        - Is there a way to transform a process fragment into a format that can be processed during monitoring? If not, why? If yes, how?
        - Which events are necessary to assess compliance of the execution of a business process? What is the event model? Can any compliance concern be captured by events?
- Design and implementation of supporting infrastructure
    - Process Engine:
        - Can the process fragments and the process models be executed in the (extended) process engine?
        - Does the process engine support the monitoring of the process executions by generating all required events?
        - Is this implementation applicable to all compliance requirements that are defined in the case study and that are said to lie within the scope of the project? If not, why?
    - Process (-fragment) Repository:
        - Can the models represented by BPEL artefacts or textual annotations be retrieved and stored in this repository?
        - Are there any differences between the developed repository and conventional repositories, e.g., a repository for source code management?
    - Annotation Tool:
        - Does this tool support the annotation of processes and process fragments in a sufficient manner?

3.7. Research Metrics for WP5

The main goal of WP5 is to provide support to monitoring and management of compliance concerns. To achieve these goals, WP5 will use process monitoring and managing of business events.

This way, WP5 will provide a data warehousing solution to store business process execution and compliance events, in order to offer visualizations components like OLAP features and web-based dashboards. Based on these components, this work package measures business performance with special focus on compliance in order to support quick and more reliable decision making, before problems arise or bad performance reaches a critical level.

With respect to the these objectives WP5 can evaluate its contribution using the following metrics

- Can we store information about all processes and violations? (%)
- Can we compute all the indicators?
- Can we positively evaluate the compliance indicators?
- Can we find correlations among the indicators?
  - We can get quantitative metrics from the case studies and then a qualitative discussion of all of them.
- Can we properly model the licensing concerns with the DSL metal-model proposed?
- Can we present the main information to start the monitoring and management compliance analysis based on the dashboards visualization?
- Can we identify at a glance the main indicators presented by the dashboard interface?
- Can we provide a user friendly solution to the users navigate thought the data stored in the data warehouse?

4. Reference documents

4.1. Internal documents

[DoW]	"Description of Work" for COMPAS, final version of 2008-02-01
[D1.1]	"Model Driven Integration Architecture for Compliance", 2009-01-31.
[D5.2]	"Initial Goal-oriented data model", 2009-01-31.
[D6.1]	"Use Case, Metrics and Case Study Definition", 2008-07-31

4.2. External documents

[EcBPEL]	Eclipse BPEL Project; http://www.eclipse.org/bpel/.