Final goal-oriented data model

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 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

D5.3 Final goal-oriented data model

Revision 1.0

Due date of deliverable: 2009-07-31
Actual submission date: 2009-07-31

Organisation name of lead partner for this deliverable:
UNITN Universita degli Studi di Trento, Italy

Contributing partner(s):
TARC-PL Telcordia Poland, Poland
TUV Vienna University of Technology, Austria
USTUTT Universitaet Stuttgart, Germany
UCBL Université Claude Bernard Lyon 1, France

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th>Project funded by the European Commission within the Seventh Framework Programme</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU</td>
<td>Public</td>
</tr>
<tr>
<td>PP</td>
<td>Restricted to other programme participants (including the Commission Services)</td>
</tr>
<tr>
<td>RE</td>
<td>Restricted to a group specified by the consortium (including the Commission Services)</td>
</tr>
<tr>
<td>CO</td>
<td>Confidential, only for members of the consortium (including the Commission Services)</td>
</tr>
</tbody>
</table>
## History chart

<table>
<thead>
<tr>
<th>Issue</th>
<th>Date</th>
<th>Changed page(s)</th>
<th>Cause of change</th>
<th>Implemented by</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>2009-06-22</td>
<td>All sections</td>
<td>New document</td>
<td>UNITN</td>
</tr>
<tr>
<td>0.2</td>
<td>2009-07-03</td>
<td>Section 5 and Appendix A</td>
<td>New text</td>
<td>UNITN</td>
</tr>
<tr>
<td>0.3</td>
<td>2009-07-09</td>
<td>Section 4</td>
<td>Event producers</td>
<td>TUV</td>
</tr>
<tr>
<td>0.4</td>
<td>2009-07-10</td>
<td>All sections</td>
<td>New conceptual model and descriptions about the reference scenario</td>
<td>UNITN</td>
</tr>
<tr>
<td>0.5</td>
<td>2009-07-14</td>
<td>All sections</td>
<td>New text</td>
<td>UNITN</td>
</tr>
<tr>
<td>0.6</td>
<td>2009-07-14</td>
<td>Section 4</td>
<td>Revised event producers + CEP</td>
<td>TUV</td>
</tr>
<tr>
<td>0.7</td>
<td>2009-07-15</td>
<td>All sections</td>
<td>Updated and revised section 4. Updated and corrected literature references.</td>
<td>USTUTT</td>
</tr>
<tr>
<td>0.8</td>
<td>2009-07-14</td>
<td>All sections</td>
<td>New text and updates</td>
<td>UNITN</td>
</tr>
<tr>
<td>0.85</td>
<td>2009-07-19</td>
<td>Section 3</td>
<td>Added description of data model requirements and proposed solution</td>
<td>UNITN</td>
</tr>
<tr>
<td>0.87</td>
<td>2009-07-24</td>
<td>Added section 3.3, modified</td>
<td>Added an example of using warehouse, improved the text in the sections</td>
<td>UNITN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sections 1 and 2 and 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.88</td>
<td>2009-07-24</td>
<td>Sections 1-3 and 7</td>
<td>Addressed some comments by TILBURG</td>
<td>UNITN</td>
</tr>
<tr>
<td>0.89</td>
<td>2009-07-27</td>
<td></td>
<td>Updates from UNITN and TARC-PL</td>
<td>UNITN, TARC-PL</td>
</tr>
<tr>
<td>0.90</td>
<td>2009-07-28</td>
<td>Section 4.4</td>
<td>Added information concerning JBI for integration with Apache ServiceMix</td>
<td>USTUTT</td>
</tr>
<tr>
<td>0.91</td>
<td>2009-07-29</td>
<td>All sections</td>
<td>Addressing reviewers’ comments</td>
<td>UNITN</td>
</tr>
<tr>
<td>0.92</td>
<td>2009-07-29</td>
<td>All sections</td>
<td>Addressing additional reviewers’ comments</td>
<td>UNITN</td>
</tr>
<tr>
<td>1.0</td>
<td>2009-07-31</td>
<td>All sections</td>
<td>Removing comments and preparing for the release</td>
<td>UNITN</td>
</tr>
</tbody>
</table>

## Authorisation

<table>
<thead>
<tr>
<th>No.</th>
<th>Action</th>
<th>Company/Name</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prepared</td>
<td>UNITN</td>
<td>2009-07-31</td>
</tr>
<tr>
<td>2</td>
<td>Approved</td>
<td>TUV</td>
<td>2009-07-31</td>
</tr>
<tr>
<td>3</td>
<td>Released</td>
<td>TUV</td>
<td>2009-07-31</td>
</tr>
</tbody>
</table>
## Contents

1. Introduction ................................................................................................................................. 9
   1.1. Purpose and scope................................................................................................................ 9
   1.2. Document overview .......................................................................................................... 10
   1.3. Improvements with respect to “Initial goal-oriented data model” [D5.2] ................. 11
   1.4. Reference scenario ........................................................................................................... 11
      1.4.1. Compliance requirements .................................................................................. 13
   1.5. Definitions and glossary ................................................................................................. 15
   1.6. Abbreviations and acronyms ......................................................................................... 15

2. Conceptual model of compliance ............................................................................................. 16
   2.1. Compliance management concepts ............................................................................. 16
   2.2. Example instantiation of the conceptual model ........................................................... 21

3. Warehouse data model .............................................................................................................. 22
   3.1. Requirements ................................................................................................................... 22
   3.2. Data model for compliance ........................................................................................... 23
      3.2.1. Facts and dimensions ....................................................................................... 23
      3.2.2. Data model .......................................................................................................... 26
   3.3. Reference scenario example ......................................................................................... 29

4. Events in COMPAS .................................................................................................................. 33
   4.1. Event model ..................................................................................................................... 33
   4.2. Event generation ............................................................................................................. 34
   4.3. Event processing ............................................................................................................. 36
   4.4. Event logging .................................................................................................................. 37

5. The ETL process ....................................................................................................................... 39
   5.1. Application integration component .............................................................................. 40
      5.1.1. Event metadata .................................................................................................. 41
      5.1.2. Wrappers ........................................................................................................... 41
      5.1.3. Extraction Routine ............................................................................................ 42
   5.2. Data integration component .......................................................................................... 42
   5.3. Event loading and temporal representation of the facts ............................................... 43
   5.4. Dynamic aspects of the ETL process ........................................................................... 44

6. Real-time detection and analysis of the compliance deviations ............................................ 45
   6.1. Overview of the Event-to-Compliance (E2C) architecture ........................................ 45
   6.2. Event contextualization ............................................................................................... 45
   6.3. Compliance evaluation ................................................................................................. 47
6.4. Compliance notification .................................................................................................... 48
6.5. Example of real-time detection of compliance violation ................................................ 48
7. A model and DSL for compliance to licenses ......................................................................... 49
  7.1. Introduction ........................................................................................................................ 49
  7.2. License profile ................................................................................................................... 50
  7.3. A model and a DSL for service licenses .......................................................................... 50
    7.3.1. Subject model and syntax .......................................................................................... 54
    7.3.2. Scope of rights model and syntax ............................................................................. 54
    7.3.3. Financial terms model and syntax ............................................................................. 58
    7.3.4. Warranties, Indemnities, and Limitation of liabilities. Models and syntax ............ 58
    7.3.5. Evolution model and syntax ...................................................................................... 61
  7.4. DSL for service licensing in the WatchMe Scenario ...................................................... 63
8. Conclusions ............................................................................................................................... 65
9. Reference documents ................................................................................................................ 66
  9.1. Internal documents ............................................................................................................ 66
  9.2. External documents ........................................................................................................... 66

List of figures

Figure 1 Compliance governance runtime architecture......................................Error! Bookmark not defined.
Figure 2 WatchMe scenario................................................................................................. 12
Figure 3 WatchMe BPMN model ...................................................................................... Error! Bookmark not defined.
Figure 4 Conceptual model of the compliance management problem. The figure introduces the basic concepts, terminology and relationships. ................................. 17
Figure 5 Instantiation of the conceptual model showing a specific instant in time of the compliance state of WatchMe ...................................................................................................... 21
Figure 6 Dimensional fact model of a business event ............................................................ 23
Figure 7 Dimensional fact model of a business process instance ......................................... 25
Figure 8 Dimensional fact model of a business process activity instance ............................. 25
Figure 9 Dimensional fact model of a service instance .......................................................... 26
Figure 10 The warehouse data model .................................................................................... 28
Figure 11 Architecture and implementation details of the Event log ..................................... 38
Figure 12 Structure of the Log_table .................................................................................... 39
Figure 13 Example of a Log_table query .............................................................................. 39
Figure 14 COMPAS ETL process ......................................................................................... 40
Figure 15 The dynamic aspects of the COMPAS ETL process ............................................ 44
Figure 16 Event-to-Compliance (E2C) architecture of the Runtime rule evaluator component ................................................................. 45
Figure 17 Compliance evaluation specification model for WatchMe service delivery rate QoS compliance ...................................................... 49
Figure 18 License profile ......................................................................................................................................................... 50
Figure 19 Model of a service license ........................................................................................................................................ 51
Figure 20 Conceptual mapping of service licensing clauses in ODRL-S ................................................................................... 53
Figure 21 Graphical syntax of license clause models ................................................................................................................ 53
Figure 22 ODRL-S Subject license clause ............................................................................................................................... 54
Figure 23 ODRL-S Scope of rights license clause ...................................................................................................................... 56
Figure 24 ODRL-S Financial terms license clause ................................................................................................................... 58
Figure 25 ODRL-S Warranty license clause .......................................................................................................................... 59
Figure 26 ODRL-S Indemnity license clause ........................................................................................................................... 59
Figure 27 ODRL-S Limitations of liability license clause ........................................................................................................ 60
Figure 28 ODRL-S Evolution license clause ......................................................................................................................... 62
Figure 29 Licensing clause of Provider1 - Pay-view plan ........................................................................................................... 64

List of tables
Table 1 Compliance requirements for Licensing ......................................................................................................................... 14
Table 2 F_BusinessEvent table ..................................................................................................................................................... 29
Table 3 F_BusinessEventBridge table ......................................................................................................................................... 30
Table 4 F_BPInstance table .......................................................................................................................................................... 30
Table 5 F_BPInstanceData table .................................................................................................................................................... 30
Table 6 F_VideoStreams_BusinessData table ............................................................................................................................ 30
Table 7 F_BPActivityInstance table ........................................................................................................................................... 31
Table 8 F_ServiceInstance table .................................................................................................................................................. 31
Table 9 F_ComplianceEvaluation table ..................................................................................................................................... 32
Table 10 D_ComplianceConcern table ..................................................................................................................................... 32
Table 11 D_Role table .................................................................................................................................................................. 32
Table 12 D_Actor table .................................................................................................................................................................. 32
Table 13 D_Time table .................................................................................................................................................................. 32
Table 14 D_CalendarDate table .................................................................................................................................................... 32
Table 15 D_BP table ...................................................................................................................................................................... 32
Table 16 D_BPActivity table ........................................................................................................................................................ 33
Table 17 D_Service table ............................................................................................................................................................ 33
Abstract

Warehousing business process data is the first step toward enabling the analysis of business data and the detection of violations of compliance rules in business process executions. Although many solutions for warehousing exist, compliance is usually not taken into account as a first-class concept.

In this deliverable, we first describe a reference scenario taken from one of our project case studies; it will allow us to exemplify the concepts, ideas, and solutions introduced in this deliverable. Specifically, we start with describing the problem of compliance governance from a conceptual point of view. Then, we describe the final version of the goal-oriented data model for warehousing business process execution data and compliance data. The model allows for the storage of business process data in relation to compliance data, thus supporting the analysis of compliance deviations, as well as for traditional OLAP-like analysis and the implementation of generic Business Intelligence (BI) applications. After describing the data model, we describe the ETL process that we use to load the data into the warehouse.

Finally, in this deliverable, we further describe the architecture for real-time detection and analysis of the compliance violations and present a model and a DSL for the specification of compliance requirements regarding licenses.

This deliverable refines and replaces Initial goal-oriented data model [D5.2] and is the result of the first implementation experiences.
1. Introduction

A data warehouse (DW) is an integrated, non-volatile, historical, subject-oriented data collection, aimed at supporting decision-making processes [Inm05]. In addition, there is the data warehousing environment, which is a more comprehensive concept defined as a process for assembling and managing the DW. For that reason, a data warehousing solution has to encompass: i) business modelling, ii) Extract, Transform and Load (ETL) tools, iii) analytical tools targeting at end-users, and iv) repository management and maintenance functionality. The COMPAS framework aims at addressing almost all of these aforementioned items. The first, the second, and the fourth items are discussed in this document, while the third will be defined according to the DoW specification in one of the forthcoming deliverables.

This deliverable describes the data model for the warehouse that will be developed in the COMPAS project. The distinguishing feature of this data model with respect to traditional approaches for warehousing business process execution data is its focus on compliance data and their relation with business process execution data. This is required in order to check for compliance deviations occurred during business process executions. Also, the DW supports generic OLAP-style analyses and the execution of the BI algorithms developed in WP5.

1.1. Purpose and scope

First of all, it is important to note that this deliverable is the evolution of Initial goal-oriented data model [D5.2], which already introduced an initial version of the DW model, along with some correlated concerns, such as the ETL procedures, the logic of how to perform complex event processing to check compliance rules, and a DSL for licensing concerns.

This being said, to better understand the purpose and scope of this deliverable let us consider Figure 1, which shows a high-level view of the compliance governance runtime architecture (part of the overall architecture of COMPAS - [DA.1]) and helps us to map the contributions of this deliverable to the respective components of the architecture.

![Figure 1 Compliance governance runtime architecture](image-url)
The left-hand side of the figure represents the runtime aspect of the architecture; the right-hand side represents the compliance evaluation aspect of the architecture. The execution of services and business processes generates events, which provide information about the concrete execution of service and process instances and enable the assessment of their compliance. Events in this architecture are therefore generated by the Business process engine and by the (instrumented) services and published via a central Enterprise Service Bus (ESB) that is a unified communication channel between the components. The ESB provides publish/subscribe support for events to the components on the right part of the figure. The Event log, Runtime rule evaluator, and Business protocol monitoring components “consume” events by subscribing to them. The latter two components also produce high-level events in reaction to events coming from the Instrumented services and the Business process engine. These events are again published on the ESB to any “interested” component that has subscribed for those types of events. In the proposed architecture the Event log component is subscribed to any event that is relevant for monitoring and further processing such as Log mining. Additional process execution data (e.g., data that is not carried with events, such as engine-internal events) are stored in a dedicated Audit trail. Process and process fragment models are stored in an own repository and processes are deployed onto the Business process engine for execution.

Starting from the Event log and the Audit trail, the right-hand side of the figure details the COMPAS Analysis/BI solution, including components such as ETL procedures, the DW, the Log mining component, and the Compliance governance web user interface that includes Reporting dashboards. The BI component is in charge of analyzing the data in the DW and of generating compliance reports.

With respect to Figure 1, this deliverable discusses the data model for the Data warehouse, the Event log, the ETL procedures, and the Runtime rule evaluator. The other components of the architecture are the subjects of other deliverables. The deliverable also contains a DSL for licensing. We would like to stress that the DSL presented here is just one of several DSLs produced by COMPAS. The role of DSLs in COMPAS, their relation with business processes, the description of the main actors involved in using DSLs are out of the scope of this deliverable and are explained in “Model-driven integration architecture for compliance” [D1.1].

1.2. Document overview

This deliverable consists of two parts:

The first part focuses on the data warehousing aspect of the architecture, also introducing the necessary concepts and terminology. It is divided into several sections: in Section 1, we introduce the reference scenario that will drive our discussion throughout the deliverable. In Section 2, we describe and explain the main business and compliance concepts reflected in the data model and relations between them, while in Section 3 we present the data model of the DW for compliance analysis. Section 4 focuses on the events in the COMPAS infrastructure and introduces the Event log, in order to discuss in Section 5 the ETL process used to fill the DW. In Section 6, we approach the problem of real-time detection and analysis of compliance deviations, as required for Runtime rule evaluator.

The second part (Section 7) presents a meta-model and a DSL for compliance to licenses and provides a usage example. Finally, Section 8 concludes the deliverable.
1.3. Improvements with respect to “Initial goal-oriented data model” [D5.2]

In order for the reader to better understand which improvements have been made to the “Initial goal-oriented data model” [D5.2], we summarize the main aspects that differentiate this deliverable from the initial version:

1. We now use one case study example as a reference scenario throughout the whole deliverable: the WatchMe Mobile Virtual Network Operator (MVNO) scenario introduced in the next subsection.

2. We provide an updated version of the conceptual model underlying the overall data warehousing approach for compliance assessment, along with the necessary terminology. Both the conceptual model and the terminology are the output of the dedicated COMPAS Terminology meeting held in Trento on July 6-7, 2009. The aim of the meeting was exactly the definition of a project-wide, shared conceptual model and glossary.

3. We show an example of how to instantiate the conceptual model, taking into account the WatchMe scenario and the new conceptual model.

4. We refine the data warehouse model according to the new conceptual model and to the first implementation experiences.

5. We introduce a new section discussing all event-related issues in WP5, as they are the main input to the compliance assessment and, hence, of high importance to the development of all the other solutions in WP5.

6. We describe now more concretely how the ETL process will look like, taking into account the new DW model and the common event format.

7. We provide an example of real-time detection of compliance violation, considering the WatchMe scenario.

8. We provide an example of how to apply the licensing DSL to the new reference scenario.

The aspects not explicitly mentioned in the above list might have changed slightly in this deliverable, but without impacting their idea and understanding in comparison to [D5.2].

1.4. Reference scenario

This subsection presents the WatchMe reference scenario that is going to be used along the whole document with the purpose of exemplifying how our solution addresses compliance requirements. As in this deliverable we also introduce a DSL for licensing, we will specifically focus on the case of compliance to licenses. Next, we describe the scenario and then actor and business perspectives. We conclude the scenario description by highlighting the compliance requirements that must be monitored.

WatchMe focuses on the advanced telecom services offered by an MVNO. Such services combine value-added application capabilities with Internet and the next generation of mobile telecommunication network features. All these capabilities are integrated by MVNO services to provide combinations of call/session control, messaging features, presence and location features, multimedia content streaming, parental monitoring, etc. The MVNO environment is particularly challenging due to the heterogeneous characteristics of the network infrastructure.
Many of the applications that provide the MVNO service components are owned and managed by different enterprises (i.e., the MVNO, the network providers, and third-party application providers). Moreover, the current trends of such systems of services include organizing them in a SOA-oriented fashion, thus fostering loosely coupled networks of interoperating web services.

Figure 2 illustrates part of the WatchMe scenario: we are going to deal mainly with one type of compliance sources, i.e., licenses. The left-hand side of the figure shows the video (Video1 and Video2) and audio (Audio1 and Audio2) providers available in the WatchMe system, which can be dynamically composed and consumed by customers (e.g., Bob) via multimedia streaming. The video providers impose their licenses (L1 and L2), which they wrote in conjunction with the audio providers they have agreements with. These licenses, therefore, state which audio providers can be used to merge an audio stream with the video provider’s video stream (e.g., video provided by Video2 just can be downloaded with audio offered by Audio2). The right-hand side of the figure concentrates on the QoS provided to consumer (e.g., throughput, response time, download fare).

Given that scenario, COMPAS should now monitor the compliance of the WatchMe scenario with the licenses and QoS agreements by verifying if the plans and services sold to its customers are compliant to the set of requirements explained in the following.

The business process model (modelled in BPMN) depicted in Figure 3 represents WatchMe internal procedures used to support the interactions between the customers’ services (e.g. Consumer’s cell phone service) and the third party services that constitute the system. Basically, the execution flow starts with consumer’s subscription to the WatchMe service offered by the MVNO. Then, if the login is correct, consumer is allowed to specify search criteria for the stream he’s looking for (e.g. newest episode of Dr. House with English audio track). Then, MVNO service – WatchMe business process – duplicates and resends the search messages to each video and audio third party providers, who signed appropriate license with the MVNO. Appropriate replies are gathered by WatchMe and merged in a single message with summary of the results, which is further sent to the client. Consumer chooses video and audio stream which he’d like to get access to. Finally, WatchMe business process receives endpoints for the chosen streams from the third party providers, merges them and streams the media directly to consumer’s cell phone.
1.4.1. Compliance requirements

Currently, WatchMe supports two types of plans dedicated to clients of streaming download: *Pay-per-view* and *Time-based*. Both state how many stream acquisitions a customer can do based on the plan he chose. In the implementation view, it means how many successful service invocations the client is allowed to do, based on a parameter $n$ defined according to the price ($p$) paid by a customer to be signed in a specific plan. Additionally, during the stream acquisition it is necessary to obey to the licensing terms established with the providers. For instance, the video for the last season of Dr. House can only be downloaded with English
audio. In order to monitor the compliance during stream acquisition, the list of compliance requirements, shown in Table 1, must be checked and controlled at runtime. The table includes the name of the compliance requirements, the controls that the process must satisfy to be compliant, and some additional descriptions. The definition of this reduced set of requirements has the aim of exemplify the compliance issues that must be addressed to achieve compliance. In this deliverable, we focus on compliance for licensing. Security and QoS issues will be addressed by UCBL and TUV D5.4 and D1.2, respectively.

<table>
<thead>
<tr>
<th>Name of the Compliance Requirements</th>
<th>Control</th>
<th>Additional Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pay-per-view plan</td>
<td>The user ID subscribed for the plan can acquire only ( n ) possible streams at price ( p ).</td>
<td>We check whether we comply with the max number of video playbacks allowed, which means max number of service invocations. We do not check (for now) whether the payment has been made.</td>
</tr>
<tr>
<td>Time-based plan</td>
<td>The user ID subscribed for the plan can acquire any number of times any possible streams from StartDate till EndDate of the plan.</td>
<td>We do check whether we comply with the subscription period. We do not check (for now) whether the payment has been made.</td>
</tr>
<tr>
<td>Composition permission</td>
<td>Only pre-defined combinations of video and audio providers are allowed due to the licenses specified by the video provider (V1 with A1 or A2, V2 with A2).</td>
<td>Composition in the sense of Table 20. If V1 and V2 are provided by two different &quot;content providers&quot;, then each should only mention its own content streams not the other ones.</td>
</tr>
</tbody>
</table>

Table 1 Compliance requirements for Licensing

Note that the full compliance with the described license restrictions, of course, requires that all business practices in the company are compliant; the case of the merge of video and audio streams is only one of the services provided by WatchMe and, therefore, represents only one out of multiple business practices in the company that are subject to the licenses by the video providers. Hence, the compliance of license does not imply the compliance of the overall business. Also, regarding the process depicted in Figure 3, we assume that the execution of the described process is automatically assisted by means of web services/applications and a business process engine orchestrating them. Nonetheless, the sole implementation of the compliant business process does not yet guarantee compliance: failures during process execution may happen (e.g. due to human errors, system failures or the like), and disallowed combinations of video and audio streams could be merged, or the terms of the subscription plans could be violated.
Awareness of such problems is of utmost importance to the company, in order to be able to react timely and, hence, to assure business continuity. In this regard, periodical and up-to-date reports about the compliance state of the company and, possibly, the ability to perform root-cause analysis to understand the reason for specific non-compliance problems starting from the monitoring of the company’s IT infrastructure and business processes are required, so as to allow the company to improve its responsiveness and the level of compliance.

1.5. 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.

1.6. Abbreviations and acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BI</td>
<td>Business Intelligence</td>
</tr>
<tr>
<td>BPEL</td>
<td>Business Process Execution Language</td>
</tr>
<tr>
<td>BPMN</td>
<td>Business Process Modelling Notation</td>
</tr>
<tr>
<td>CEP</td>
<td>Complex Event Processing</td>
</tr>
<tr>
<td>COMPAS</td>
<td>Compliance-driven Models, Languages, and Architectures for Services</td>
</tr>
<tr>
<td>CTR</td>
<td>Cleaning and Transformation Routines</td>
</tr>
<tr>
<td>DD</td>
<td>Data Dictionary</td>
</tr>
<tr>
<td>DoW</td>
<td>Description of Work</td>
</tr>
<tr>
<td>DSA</td>
<td>Data Staging Area</td>
</tr>
<tr>
<td>DSL</td>
<td>Domain-specific Language</td>
</tr>
<tr>
<td>DW</td>
<td>Data Warehouse</td>
</tr>
<tr>
<td>E2C</td>
<td>Event-to-Compliance</td>
</tr>
<tr>
<td>ESB</td>
<td>Enterprise Service Bus</td>
</tr>
<tr>
<td>ETL</td>
<td>Extract, Transform, and Load</td>
</tr>
<tr>
<td>HIPAA</td>
<td>Health Insurance Portability and Accountability Act</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>MVNO</td>
<td>Mobile Virtual Network Operators</td>
</tr>
<tr>
<td>ODRL</td>
<td>Open Digital Rights Language</td>
</tr>
<tr>
<td>ODE</td>
<td>Orchestration Director Engine</td>
</tr>
<tr>
<td>OLAP</td>
<td>OnLine Analytical Processing</td>
</tr>
<tr>
<td>OWB</td>
<td>Oracle Warehouse Builder</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>SOA</td>
<td>Service Oriented Architecture</td>
</tr>
</tbody>
</table>
2. Conceptual model of compliance

The focus of this deliverable is the development of the data warehousing solution for compliance governance. Before starting to think about concrete data models or ETL solutions to populate the warehouse with actual data, that is, before thinking about the solution, it is important to thoroughly understand the problem, i.e. IT-assisted compliance governance. Our reference scenario in Section 1.4 intuitively conveys what are the issues that characterize the typical compliance governance problem and what can we aim to do is to help companies in being compliant.

Here, we abstract from the peculiarities of the specific example and describe a conceptual domain model that introduces the reader with the main terminology and concepts used throughout the rest of the deliverable (and the project). As an example for the application of the conceptual model to real world business, we then again look at the compliance governance scenario described in the reference scenario, taking into account the terms and concepts of the conceptual model.

Please note that the following conceptual model and terminology are the outcome of the dedicated project meeting, which was held on July 6-7, 2009, in Trento. The aim of the meeting was exactly the definition of a COMPAS wide conceptual model and an improved COMPAS glossary that has been accepted by all the partners involved in the project.

2.1. Compliance management concepts

Intentionally, our reference scenario is written in an intuitive and easy-to-understand fashion. However, tackling compliance in a BI solution for compliance governance requires a more rigorous definition of concepts, relationships, and terminology. For this purpose, Figure 4 depicts the conceptual model that characterizes the domain of compliance governance from a business perspective and that underlies the DW schema discussed in the next section.

Notice that the model provides a business perspective on the compliance governance problem, since this is the perspective of the final users of the compliance dashboard, delivering compliance reports computed over the DW. Here we do not focus on how specific solutions are implemented, but on what are the concepts, relationships and features that characterize the problem space.
Figure 4 Conceptual model of the compliance management problem. The figure introduces the basic concepts, terminology and relationships.

According to the UML syntax for class diagrams, concepts are represented as objects (boxes), and we distinguish four kinds of relationships: associations (normal line connectors), aggregations (connectors with a white diamond), compositions (connectors with a black diamond), and specializations (connectors with a triangle). Relationships may have cardinality constraints.

The meaning of the concepts and relationships in the models is as follows (see also the Terminology section in the project web site: http://www.compas-ict.eu/terminology.php):

- **Compliance Source**: A document that is the origin of compliance requirements. For example, SOX, HIPAA, License.

  Compliance Sources are the actual driver of the compliance problem, in that they pose requirements (in form of laws, regulations, or similar) to how the business of a company should be executed.

- **Compliance Requirement**: A constraint or assertion that results from the interpretation of the compliance sources. It may be defined in various levels of abstraction. For example: whenever you enter a room you have to say “hello”; or: the person who processes a loan request cannot be the one who approves it.

  Compliance Requirements are derived from the Compliance Sources and represent those requirements expressed in the source the company wants to comply with. In a given source, there are typically a large number of requirements; only some of them really apply to the company, typically based on its particular business area.
• **Compliance Objective**: The compliance requirements to which an actor wants to comply.

• **Compliance Risk**: The risk of impairment to the organization’s business model, reputation and financial condition (resulting) from failure to meet compliance requirements.

• **Compliance Rule**: An operative definition of a compliance requirement. For example: \(G(\text{PostCreditWorthinessCheck} \rightarrow G(\text{PostCreditWorthinessCheck}.\text{Role}(\text{PostProcessingClerk})))\) (Description: Post CreditWorthinessCheck should be performed by Post Processing Clerk).

A Compliance Rule is an operative representation of a Compliance Requirement, which is typically specified in natural language and, therefore, needs interpreted and adapted to a company’s internal practices, in order to be executed and controlled.

• **Compliance Policy**: A set of compliance rules. For example: Monitoring, Privacy, Security, QoS policy, Service composition policy.

For internal order, Compliance Rules are typically aggregated into Policies collecting rules with similar semantics. For instance, all the Compliance Rules deriving from a set of licenses (multiple Compliance Sources) can be grouped into a company’s internal licensing policy, i.e., a document that describes how the company intends to deal with compliance with licenses.

• **Compliance Annotation**: It is a representation of compliance rules in textual form or specified as process fragment, which is connected to one or more controls e.g., a Business Process Logic Fragment, or one or more compliance targets. For example: (i) the Textual Annotation defining the different roles for two different actors of two activities of a Business Process Logic Fragment therefore specifying segregation of duty, or (ii) annotation of a loan approval Business Process with an Annotation Business Process Fragment specifying the order of execution of the activities: risk assessment before loan approval.

It is important to note that in COMPAS operative compliance requirements can be expressed in two different ways, (i) Compliance Rules and (ii) Compliance Annotations. The former are simple statements, the latter also describe some process logic plus constraints.

• **Textual Annotation**: It is the representation of one or more Compliance Rules in textual form. For example: store the execution data of every plane manufacturing Business Process instance for 20 years after delivery to the airline.

• **Annotation Business Process Fragment**: It is the representation of one or more Compliance Rules in a form of a Business Process Fragment. For example, Annotation of a loan approval Business Process with an Annotation Business Process Fragment specifying the order of execution of the activities: risk assessment before loan approval.

Note that the Annotation Business Process Fragment is both a specialization of Compliance Annotation and BP Fragment (defined next).
• **Business Process Fragment**: A process fragment is a (parameterized and/or constrained) set of process parts that represent a reusable solution (pattern) to achieve compliance requirements. It is implied that the process parts within the fragment are somehow related.

That is, the Business Process Fragment can be re-used in the design of a business process, thereby providing the process with a piece of process logic that (i) performs some business action and (ii) contains a control mechanisms to check compliance with a given set of rules or annotations.

• **Business Process Logic Fragment**: A Business Process Logic Fragment implements one or more Controls. For example, a Business Process Logic Fragment containing the activities customer solvency check before charge of the customer’s credit card and delivery of ordered goods afterwards.

• **Control**: A logical part of the business process that exercises the restraining or directing influence to check, verify, or enforce rules to satisfy a compliance requirement. For example, customer's initial credit worthiness checks (by credit broker) are segregated from post credit worthiness check (by Post-processing clerk). If a credit exceeds 1 M Euro, the post-processing supervisor checks whether the operation is profitable after 2nd check.

In order to check the Compliance Rules and Compliance Annotations it is necessary that there are some Controls in place, which apply the logic specified by them to the Compliance Targets deployed in the company.

• **Compliance Target**: The generic target of compliance requirement. For example, Business processes, Business process activities, Web services.

Compliance Rules and Compliance Annotation are not just generic rules for business. They can be attached to specific Business Processes, Business Process Activities, or Web Services, i.e., to so-called Compliance Targets, which, therefore, must comply with them.

• **Business Process**: A composition of activities into a structured order that implements the procedure to be followed in order to achieve a business goal.

• **Business Process Activity**: A unit of work performed automatically or manually by actors in a business process. For example, approve a business trip.

• **Web Service**: A Web service is a software system designed to support interoperable machine-to-machine interaction over a network [ACK+03]. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. In addition, it is important to highlight that Web services represent an important approach to realize SOA.

• **Compliance Target Instance**: A single execution of a compliance target. For example, BP and Activity: one concrete instance or Web Service: one concrete conversation.
When assessing compliance, auditors typically do not only look at how a company’s business practices (the Compliance Targets) are designed and implemented, but also—and more importantly—at how they are actually executed. That is, compliance is basically assessed by looking at how business has been executed, not at how business should be executed.

- **Actor**: A human or non-human entity that performs a piece of work. The actor is someone/something that can have a role and execute a task. For example: Bank, Clerk, Department of an organization, Web service.

  The execution of a Compliance Target Instance typically involved one or more persons (Actors). For instance, it is important to track which person performed which activity in order to check whether separation of duties rules have been followed or not.

- **Role**: A description of a set of responsibilities in a business environment played by an actor. For example: Compliance Expert, Process Manager, Compliance Officer, Technical Specialist, Process Analyst, Internal and External Auditors.

- **Business Event**: The event that occurs during the execution of a business process and that has relevance from a business standpoint. For example, the receipt of an invoice.

  Business Events (together with Business Data) are the basis for offline compliance assessment. Events in COMPAS refer to standard business process execution events (e.g., process started), events that signal compliance assessment related information (e.g., composite stream requested), and compliance evaluation events (e.g., license clause X violated).

- **Business Data**: The data that is needed and understood by actors to perform their job. This is the data processed and transformed by the business process. For example, an invoice amount is business data, while the start time of a task in a BPEL process or the header of a SOAP message is technical data.

- **Compliance Rule Violation**: A dissatisfaction of a compliance rule with respect to a compliance target or compliance target instance. For example, a loan processed and approved by the same person. This is a violation referring to the following compliance rule: 
  \[ G(\text{PostCreditWorthinessCheck} \rightarrow G(\text{PostCreditWorthinessCheck.Role(\text{PostProcessingClerk})})) \] (Description: PostCreditWorthinessCheck should be performed by PostProcessingClerk).

  In case the evaluation of Compliance Rules or Compliance Annotations detects a violation, the Compliance Rule Violation entity tracks the registered violation for reporting and analysis purposes. This is one of the most important concepts in the whole compliance management life cycle, as it allows the company to understand when and where a Compliance Target violated a given rule. This allows the company to improve its business practices and the level of compliance.

- **Compliance Request**: A request to: (i) check whether a set of compliance targets conforms to a set of applicable compliance requirements, and (ii) identify how a process can/should be changed to make it (more) compliant.
• **Business Protocol**: It is a specification of all possible messages sequences accepted by a web service. For example: All the sequences of operations to be carried for ordering an enterprise product or solution via a web service.

• **Behavior Violation**: The difference detected during the comparison between the compliance target and the discovered behavioral model. For example, i) if the sequence Login-Pay-Deliver-Logout in the compliance target is found to be executed as Login-Deliver-Logout, or ii) if the designed finite-state machine of the business protocol and the inferred one are not equivalent.

• **Discovered Behavior Model**: A visual and schematic representation of the typical behavior of a Compliance Target in the form of a structured model inferred from a set of Compliance Target Instances. For example, any structure (Finite-State Machine, Petri net...) that visualizes the inferred behaviour model/logic.

### 2.2. Example instantiation of the conceptual model

In Figure 5 we show an instance of the conceptual model describing the compliance state of the WatchMe streaming process in a given instant of time (referring to the detection of a compliance violation).

![Diagram of the conceptual model](image)

**Figure 5 Instantiation of the conceptual model showing a specific instant in time of the compliance state of WatchMe**

The Compliance Source we are interested in Figure 5 is the license by the video provider 2, as introduced in the reference scenario. From that license, we can derive the Compliance Requirement “Video from Video provider 2 can only be merged with audio streams from Audio provider 2,” along with the Compliance Objective “Comply with composition restrictions.”
restrictions”. In order to really check whether the requirement is met or not, we assume that we specified a Compliance Rule for the ESPER rule engine (represented by the rule R1 in the figure). The rule is grouped in the Compliance Policy called “Internal licensing policy”. For simplicity, in the example we highlight only one rule; actually, from the Compliance Source we could derive multiple Compliance Requirements and multiple Compliance Rules. Here it suffices to focus on only one rule. This concludes the design time aspect of the model.

Rule R1 is associated to the WatchMe streaming Business Process (the specific Compliance Target) by means of the ESPER rule engine that is able to assess whether R1 holds for each instance of the process. That is, the rule engine together with the Compliance Rule represents the Control that the company put into place. In order to assess compliance, in Figure 5, for instance, we look at an instance of the WatchMe process, which has associated two Business Events (“Process started” at time \( t_1 \), and “V2 merged with A1” at time \( t_2 \), plus one piece of Business Data (“Price = 5,90€”). Also, the Business Process Instance has been started by Jenny (Actor) who is an Employee of the company (Role). These data represent the runtime data that we have about the execution of the business.

Finally, given this characterization of the state of the company’s business, we can detect a runtime violation of the rule R1 (represented by the Compliance Violation “Violation of rule R1” detected at time \( t_3 \)), as the event “V2 merged with A1” violated R1.

3. Warehouse data model

In this Section we present a warehouse data model that supports compliance. We first identify requirements on the data model and then identify the facts and dimensions of the DW model in accordance with the conceptual model introduced in Section 2. Finally, we present a detailed data model that contains also implementation-specific constructs.

3.1. Requirements

The following requirements on the data warehousing solution for the COMPAS project have been identified:

- **R1)** Store information about business processes, BP execution data, compliance requirements
- **R2)** Enable OLAP-analysis as well as mining and support application of BI techniques
  - R2.1) In particular, support semi-automatic derivation of the knowledge related to violated compliance requirements as selected by compliance objectives
  - R2.2) Support performing compliance analysis, including root-cause analysis for cases when compliance requirements were not met
- **R3)** Support generally applicable process models that accommodate business data from most processes while allowing room for custom, process-specific extensions
- **R4)** Provide (to reporting tools) information required to enable and facilitate the visual navigation over process execution and compliance-related indicators
  - By compliance-related indicators we mean indicators that are related to compliance, e.g. the degree of compliance of compliance targets to compliance requirements.
- **R5)** Support of the drill-down functionality
Requirement R5 is related to requirement R4 in the sense that the drill-down is a part of visual navigation over process and compliance data.

R6) Support integration with TUV’s views from the view-based modelling framework.

The integration with TUV’s view-based modelling framework will be carried within T1.5, which starts at M17.

3.2. Data model for compliance

3.2.1. Facts and dimensions

The first step in the design of the data model was to identify what concepts from the conceptual model in Section 2.1 are relevant to the data model, and which of them are facts and which are dimensions. We show the facts and dimensions in Figure 6 through Figure 9 using a Dimensional Fact Model (DFM) [GMR98] – a notation for designing a conceptual model of a DW. Each box represents a fact table that contains the name of the fact in the top part and measures in the bottom part. Measures, according to [Kim02], are “business performance metric captured by an operational system and represented as a fact in a dimensional model”. Each circle is a dimension that one can use for aggregating facts. Other related, but useless for aggregation, attributes are represented as a label on a line. A dash on the arc linking dimension with a fact or another dimension means that it is an optional attribute. To model multiple links to attributes we use a “*group” entity that links to one attribute of the group.

![Figure 6 Dimensional fact model of a business event](image)

We identified four main facts: business event, business process instance, activity instance, and service instance. Figure 6 shows the DFM of an event. Since the details of the structure of an

---

1 We would like to thank Prof. Golfarelli for providing us with a full version and documentation of the Wand tool for designing dimensional fact models.
event in COMPAS is not yet available, we show possible measures as abstract EventData attribute. Business data is an example of the data pertaining to an event. An event has the following dimensions: EventType; EventSource, which specifies where the event is originated; EventDestination, which shows where the event should be delivered; BPActivityInstance, ServiceInstance, BPInstance dimensions, which show executions related to the event, EventTime, and EventDate. For the EventDate dimension we show the complete hierarchy, i.e. it is possible to group events by weeks, day of the week, month, year, etc. In the following DFMs, we will not show such level of details to avoid cluttered diagrams. For more information about the structure of an event we refer the reader to Section 4.1.

Figure 7 through Figure 9 show the DFMs of business process instance, business process activity instance, and service instance. We identified the following main dimensions common for business executions:

- **Time:** this dimension represents general temporal dependence. Each fact has the temporal dimension. In some cases, more than one association might be present – a business process instance might have start time and end time. The following attributes in the figures represent the time dimension: StartTimestamp, EndTimestamp, BPActInstTimestamp, and ServInstTimestamp.

- **Definition:** this dimension associates a particular instance with a class of such instance, as defined by BPDefinition, BPActivityDefinition, or ServiceDefinition.

- **BusinessEvent(s):** Each instance is linked to a group of business events that contain the data used during the ETL to create or update instances.

- **ComplianceConcern(s):** this dimension links an instance to a group of relevant compliance concerns. Each compliance concern includes the following hierarchy of the dimensions:
  - **EvalTimestamp:** contains information about the time compliance concern was evaluated
  - **IsViolation:** indicates if this evaluation is a violation
  - **ComplianceRule:** contains information about compliance logic
  - **CompliancePolicy:** contains information about rules associated with this policy
  - **ComplianceRequirement:** constraint or assertion that results from the interpretation of compliance sources
  - **ComplianceObjective:** contains information about requirements associated with this objective
  - **ComplianceSource:** refers to the source of the compliance requirement
  - **ComplianceRisk:** risk associated with the compliance requirement
Note that at this level we drop the concept of compliance target instance, having three facts representing instances instead. Similarly, the Compliance Target entity is replaced with BP, BP activity, and service as definition dimensions.

Apart from the dimensions common for all instances, for the activity instance we have the following additional dimensions:

- **Actor**: this dimension connects an instance to a human actor that performed the activity
- **Role**: this dimension specifies the role played by the actor while performing the activity
- **BPInstance**: this dimension connects an activity with a process instance that includes this activity

![Dimensional fact model of a service instance](image)

Figure 9 Dimensional fact model of a service instance

The proposed facts and dimensions take into account requirement R1, R2, R4, and R5, allowing for the drill-down functionality (e.g. rule->requirement->source->...), quick navigation from one entity to another, and facilitate OLAP analysis and BI application.

### 3.2.2. Data model

After identifying the facts and dimensions we produced the data model shown in Figure 10. This data model implements the DFM described in Section 3.2.1. All names of fact tables start from “F_”, all names of dimension tables start from “D_”, and all names of bridge tables start from “B_”. Some foreign keys are hidden in order not to clutter the diagram.

In the following we explain some implementation choices and describe the tables in the data model in detail.

**D_ComplianceTarget** is a general table for definitions of compliance targets such as processes, services, activities (depending on the value of the **ComplianceTarget_Type** field). It contains attributes that are common for all types of compliance targets, for instance, **ComplianceTarget_Version**, **ComplianceTarget_ValidFrom**, **ComplianceTarget_ValidTill** fields for specifying the version of the compliance target definition and the time range when this version is valid. This table also contains the **Entity_Table_Name** attribute field that contains the name of the table with target-specific instance data, as further illustrated for business process instances.

**D_BP, D_Service, D_BPActivity** tables allow for distinguishing all targets by types and can be used for storing target-specific information about its definition. For instance, the field **BPActivity_Max_Duration** of the **D_BPActivity** table specifies maximal duration of an activity, while this attribute is not necessarily applicable to processes and services. Even
though right now $D_{BP}$ and $D_{Service}$ tables contain only one column, we expect they will be extended with target-specific data later.

Process instance data is stored in the table $F_{BPInstance}$. Apart from start/end date/time, there is also reference to the process definition ($ComplianceTarget_Key$) in the $D_{ComplianceTarget}$ table (via the $D_{BP}$ table). This table also refers to the status of the instance stored in the table $D_{BPInstance_Status_Info}$, tells if the instance caused any violation ($CausedViolation$), and contains measures related to the instance, such as $BPInstance_Duration$. For this and other tables related to instances, the measures stored are only examples of those which might be required for the analysis and for reporting purposes. We expect that more measures will be determined during the implementation of the case studies.

To address requirement R3, we introduced two possibilities to store business data related to process executions: in the table $F_{BPInstanceData}$, using attribute-value format (columns $BPIData_AttrName$, $BPIData_AttrValue$) and referencing the instance from the data table (the field $BPInstance_Key$), or in process-specific tables, such as $F_{InvoiceBusinessData}$, which we put in the data model as an example. Process types are linked to such process-specific tables using the $EntityTable_Name$ column of the table $D_{ComplianceTarget}$ referenced from the table $D_{BP}$. The field $BPInstance_Data$ of the $F_{BPInstance}$ table refers to the corresponding data entry in the process-specific table.

For the activities, we applied the approach described in [CCD+07], and decided to store the data about activity instances in a single table $F_{BPActivityInstance}$. Each activity instance is linked to the parent process via the field $BPInstance_Key$ and to the previously executed activity via the $Previous_BPActivityInstance_Key$ field. The fields $Initial_Proc_Key$, $Final_Proc_Key$, $Initial_Proc_Role_Key$, and $Final_Proc_Role_Key$ allow one to identify the employee and the role, to which the activity was assigned initially, and the employee and the role that completed the activity, respectively. The table also contains information about the creation ($State_New_Date_Key$, $State_New_Time_Key$), activation (somebody begins to work on the step), and completions step states, which are often required for reports. Values of measures such as $New_To_Ended_Duration_Secs$, $Active_To_Ended_Duration_Secs$ are pre-calculated in the ETL time. Finally, this table also refers to the additional information about activity instance: the field $BPActivityInstanceFlag_Key$ links to the $D_{BPActivity_Flag}$ table. Examples of such additional information are: if the activity is the first/last in the process, what is its status, is it on time.

The fact table $F_{ServiceInstances}$ contains information about instances of business services. The measures and the additional dimensions to include in this table are subject to further investigation.

All three instance-related fact tables ($F_{BPActivityInstance}$, $F_{BPInstance}$, $F_{ServiceInstance}$) and the three subject definition tables ($D_{BPActivity}$, $D_{BP}$, $D_{Service}$) are used as dimensions in the table $F_{ComplianceEvaluation}$. This table links the instance information to the compliance concerns applicable to each instance. For each compliance concern the following information is reported: the date and the time when the compliance concern is checked, and whether the compliance concern is violated (field $IsViolation$). Each compliance concern is represented as a row in the $D_{ComplianceConcern}$ table and contains a rule, a policy, a requirement, a source, a risk, and an objective that are checked. Even though rules, policies, requirements, sources, risks, and objectives are represented as dimensions in DFM s, in the data model they are combined in a single dimension table. Such de-normalization is required in order to provide quick access from compliance evaluations to the evaluated objects. If there are several rules corresponding to a policy, etc., the information
Figure 10 The warehouse data model
about policy/requirement/etc. will be duplicated.

The table \textit{F\_BusinessEvents} contains information about events loaded and linked to the instances in many-to-many relation by means of the \textit{B\_BusinessEventBridge} table. This latter table allows for specifying the type of link. The type of link explains in what way an event is related to an instance, e.g. we could have link types corresponding to creation, activation, or finishing of a BP activity. It is still under discussion what measures should be stored in the \textit{F\_BusinessEvents} table.

The tables \textit{D\_Calendar\_Date} and \textit{D\_Time} have been added for reporting purposes (requirement R4) since in many cases information in fields such as \textit{Day\_Number\_Of\_Month}, \textit{Day\_Number\_Of\_Week} would be required.

The proposed warehouse data model has a star-like schema [Kim04], consisting of several stars that correspond to instances and compliance evaluations. There are also some additional connections between fact tables, e.g., between \textit{F\_BPActivityInstance} and \textit{F\_BPInstance}.

### 3.3. Reference scenario example

In this section we will use the example scenario from Section 1.4 to illustrate how the data will be stored in the warehouse.

Let us start the description from the events that are the source of all information. In the event table \textit{F\_BusinessEvent} we have four events corresponding to getting the streams from providers and one event corresponding to viewing the delivered media by a customer (see Table 2). For the sake of simplicity let us assume that all the events are generated by ProcessEngine1 and have an empty body. For each GetStream event there are parameters that tell which types of stream are acquired from which providers, while for ViewDeliveredMedia event the parameter tells who is viewing the media.

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Source</th>
<th>Destination</th>
<th>Timestamp</th>
<th>Parameters</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GetStream</td>
<td>ProcessEngine1</td>
<td>NULL</td>
<td>2008-11-12-03:55</td>
<td>Type=video</td>
<td>NULL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Provider=V</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>GetStream</td>
<td>ProcessEngine1</td>
<td>NULL</td>
<td>2008-11-12-09:01</td>
<td>Type=audio</td>
<td>NULL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Provider=A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Provider=V</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>GetStream</td>
<td>ProcessEngine1</td>
<td>NULL</td>
<td>2008-11-16-10:05</td>
<td>Type=audio</td>
<td>NULL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Provider=A</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>ViewDeliveredMedia</td>
<td>ProcessEngine1</td>
<td>NULL</td>
<td>2008-11-12-10:05</td>
<td>Customer=Bob</td>
<td>NULL</td>
</tr>
</tbody>
</table>

Table 2 \ F\_BusinessEvent table

Let us now have a closer look how these events are transformed into instances of business processes, activities, and services. In the \textit{F\_BusinessEventBridge} table we have several rows for the event 11 that produced BP instance 122, BP activity instance 701, activated BP
activity instance 701 and also finished that activity². Event 12 created activity instance 707, while event 15 finished business process instance 122 (see Table 3).

<table>
<thead>
<tr>
<th>EventK</th>
<th>BPInstanceKey</th>
<th>BPActivityInstanceKey</th>
<th>ServiceInstanceKey</th>
<th>LinkType</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>122</td>
<td>NULL</td>
<td>NULL</td>
<td>CREATE</td>
</tr>
<tr>
<td>11</td>
<td>NULL</td>
<td>701</td>
<td>NULL</td>
<td>CREATE</td>
</tr>
<tr>
<td>11</td>
<td>NULL</td>
<td>701</td>
<td>NULL</td>
<td>ACTIVATE</td>
</tr>
<tr>
<td>11</td>
<td>NULL</td>
<td>701</td>
<td>NULL</td>
<td>FINISH</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>12</td>
<td>NULL</td>
<td>707</td>
<td>NULL</td>
<td>CREATE</td>
</tr>
<tr>
<td>15</td>
<td>122</td>
<td>NULL</td>
<td>NULL</td>
<td>FINISH</td>
</tr>
</tbody>
</table>

Table 3  F_BusinessEventBridge table

In the table of business process instances (Table 4) we have the above-mentioned instance 122 with start/end time and start/end date referring to the corresponding entries in the D_Time and D_Calendar_Date dimension tables. We also have references to the process definition (BPK), status (StatusK), BP data (DataK) and measured duration, e.g. 4 days, 1 hour, 0 seconds, and the violation flag.

<table>
<thead>
<tr>
<th>Key</th>
<th>StartTimeK</th>
<th>EndTimeK</th>
<th>StartDateK</th>
<th>EndDateK</th>
<th>StatusK</th>
<th>DataK</th>
<th>BPK</th>
<th>Duration</th>
<th>CausedViol</th>
</tr>
</thead>
<tbody>
<tr>
<td>122</td>
<td>32</td>
<td>34</td>
<td>41</td>
<td>43</td>
<td>201</td>
<td>8</td>
<td>7</td>
<td>4d1h4m0s</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4  F_BPIInstance table

Let us have a closer look at how the business data is stored. As mentioned in Section 3.2.2, there are two possibilities: the generic table F_BPIInstanceData, in which, in our example, the information about providers is stored (Table 5), and a process-specific table F_VideoStreams_BusinessData that contains the name of the stream (Table 6).

<table>
<thead>
<tr>
<th>Key</th>
<th>BPK</th>
<th>AttrName</th>
<th>AttrValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>122</td>
<td>ProviderName</td>
<td>V</td>
</tr>
</tbody>
</table>

Table 5  F_BPIInstanceData table

<table>
<thead>
<tr>
<th>Key</th>
<th>StreamName</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Doctor House episode 2</td>
</tr>
</tbody>
</table>

Table 6  F_VideoStreams_BusinessData table

The following table, F_BPActivityInstance (Table 7), shows the activity instances mentioned in the F_BusinessEventBridge table. Please note that many activities corresponding to the process are omitted in this table, e.g. activity 677 and 703 to which activity instances 701 and 707 refer to as to previous activity instances. For both instances the processor is Julie and her

2 For the sake of simplicity we assume that this activity has duration 0 but show that for each activity there can be at least three important state changes, as shown in the table of activity instances. For the sake of readability, we put LinkType as text, even though earlier we said that it is integer.
role is Security expert, duration is 0 for the sake of simplicity. Activity instance 701 does not correspond to any violation, while activity instance 707 produces a violation.

<table>
<thead>
<tr>
<th>Key</th>
<th>701</th>
<th>707</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPInstK</td>
<td>122</td>
<td>122</td>
</tr>
<tr>
<td>ActivityK</td>
<td>1221</td>
<td>1223</td>
</tr>
<tr>
<td>PrevActivInstKey</td>
<td>677</td>
<td>703</td>
</tr>
<tr>
<td>InitProcK</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>FinProcK</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>InitProcRoleK</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>FinProcRoleK</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>State_NewTimeK</td>
<td>32</td>
<td>33</td>
</tr>
<tr>
<td>State_NewDateK</td>
<td>41</td>
<td>42</td>
</tr>
<tr>
<td>State_ActiveTimeK</td>
<td>32</td>
<td>33</td>
</tr>
<tr>
<td>State_ActiveDateK</td>
<td>41</td>
<td>42</td>
</tr>
<tr>
<td>State_EndedTimeK</td>
<td>32</td>
<td>33</td>
</tr>
<tr>
<td>State_EndedDateK</td>
<td>41</td>
<td>42</td>
</tr>
<tr>
<td>FlagK</td>
<td>801</td>
<td>801</td>
</tr>
<tr>
<td>NewToEndedDuration</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NewToActiveDuration</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ActiveToEndedDuration</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CausedViolation</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 7     F_BPActivityInstance table

In this example we do not show any service instances, so the table F_ServiceInstance remains empty (Table 8).

<table>
<thead>
<tr>
<th>Key</th>
<th>ServK</th>
<th>IsViolated</th>
</tr>
</thead>
</table>

Table 8     F_ServiceInstance table

To not overload the example with details, we do not show additional information about instances, stored in the tables D_BPInstance_Status_Info and D_BPActivityFlag.

In the table F_ComplianceEvaluation (Table 9) we show the two violations and one non-violated compliance concern that were checked. Violation 1001 corresponds to the violation of the business process instance 122, BP definition 7, occurred at date 42 and time 33, the violated compliance concern is represented in the table D_ComplianceConcern with ID 1101 (Table 10). It shows the violated rule about getting only one video from provider V, the corresponding policy, requirement, source, risk, and objective.

Violation 1002 corresponds to the violation of the activity instance 707, and so on. The evaluation of the compliance concern 1003 corresponds to the rule that tells that no more than two audio streams can be acquired from provider A. In our example, the constraints

---

3 This is just to illustrate how we store information about people and roles in the warehouse, while in our example processor would be the process engine, not human.
corresponding to this rule are met so the compliance concern is positively evaluated and does not cause any violation.

<table>
<thead>
<tr>
<th>Key</th>
<th>Compl Conc</th>
<th>DateK</th>
<th>TimeK</th>
<th>BPAct K</th>
<th>ServK</th>
<th>BPInst K</th>
<th>BPAct InstK</th>
<th>ServInstK</th>
<th>IsViolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>1101</td>
<td>42</td>
<td>33</td>
<td>7</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>1</td>
</tr>
<tr>
<td>1002</td>
<td>1101</td>
<td>42</td>
<td>33</td>
<td>1223</td>
<td>NULL</td>
<td>NULL</td>
<td>707</td>
<td>NULL</td>
<td>1</td>
</tr>
<tr>
<td>1003</td>
<td>1101</td>
<td>42</td>
<td>33</td>
<td>NULL</td>
<td>1223</td>
<td>NULL</td>
<td>707</td>
<td>NULL</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 9  F_ComplianceEvaluation table

<table>
<thead>
<tr>
<th>Key</th>
<th>Rule</th>
<th>Policy</th>
<th>Requirement</th>
<th>Source</th>
<th>Risk</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>1101</td>
<td>ROOnly1Video</td>
<td>VProviderPolicy</td>
<td>Get no more than 1 video</td>
<td>License of V</td>
<td>NULL</td>
<td>Meet V requirements</td>
</tr>
<tr>
<td>1103</td>
<td>NoMoreThan2Audio</td>
<td>AProviderPolicy</td>
<td>Get no more than 2 audio streams</td>
<td>License of A</td>
<td>NULL</td>
<td>Meet A requirements</td>
</tr>
</tbody>
</table>

Table 10 D_ComplianceConcern table

In the following we show the other relevant tables, i.e. the ones for roles (Table 11) and actors (Table 12), time (Table 13), date (Table 14), and compliance targets (Table 15-Table 18).

<table>
<thead>
<tr>
<th>Key</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Security expert</td>
</tr>
</tbody>
</table>

Table 11 D_Role table

<table>
<thead>
<tr>
<th>Key</th>
<th>UAID</th>
<th>Name</th>
<th>Lastname</th>
<th>DisplayName</th>
<th>E-Mail</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>UAID1</td>
<td>Julie</td>
<td>Smith</td>
<td>Julie Smith</td>
<td><a href="mailto:Julie@somecompany.com">Julie@somecompany.com</a></td>
<td>882052</td>
</tr>
</tbody>
</table>

Table 12 D_Actor table

<table>
<thead>
<tr>
<th>Key</th>
<th>TimestampInSecs</th>
<th>Hour</th>
<th>Minutes</th>
<th>Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>999900000</td>
<td>3</td>
<td>55</td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>999900000</td>
<td>9</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>33</td>
<td>999900000</td>
<td>23</td>
<td>43</td>
<td>0</td>
</tr>
<tr>
<td>34</td>
<td>999900000</td>
<td>10</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 13 D_Time table

<table>
<thead>
<tr>
<th>Key</th>
<th>SysDate</th>
<th>DayOfWeek</th>
<th>DayOfWeek</th>
<th>DayNum OfWeek</th>
<th>DayNum OfMnth</th>
<th>DayNum InYear</th>
<th>WeekEndDate</th>
<th>WeekNum InMnth</th>
<th>IsWorkDay</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>2008-11-12</td>
<td>WEDNESDAY</td>
<td>3</td>
<td>12</td>
<td>316</td>
<td>2008-11-16</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>2008-11-15</td>
<td>SATURDAY</td>
<td>3</td>
<td>15</td>
<td>319</td>
<td>2008-11-16</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>2008-11-16</td>
<td>SUNDAY</td>
<td>3</td>
<td>16</td>
<td>320</td>
<td>2008-11-16</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Table 14 D_CalendarDate table

| Key | |
|-----| |

Table 15 D_BP table
### 4. Events in COMPAS

A business event occurs during the execution of a business process and has relevance from a business standpoint [D7.1]. For example, the receipt of an invoice. In the COMPAS context, the events present an important part, since the compliance monitoring (runtime and offline) are done based on their behaviour during execution time. To facilitate the understanding of the events in that context, this section encompasses the events in COMPAS, since their creation until their compliance monitoring. For that, we first introduce the Event model that states how the events look like. Then, we explain how the events are generated in COMPAS and describe their main generation sources (e.g., Process Engine and Instrumented services). After that, we describe how the events are processed in COMPAS in order to detect non compliant behaviours, and finally, we present the Event log responsible for storing all events delivered to the ESB.

#### 4.1. Event model

The events share some common behaviour and characteristics that can be expressed as an Event model. Each event has a set of related attributes to express its execution performance and details. Events may be expressed at different levels of granularity, but for this section we want to define a common ground. By abstracting from the details, we can say that an event is a tuple $e = <t, s, ts, d, p_1, …, p_n, B>$ with the following characteristics:

- $t \in T = \{\text{ProcessStart, ActivityExecuted, ServiceFailure, DeadlineMissed, …}\}$ where $t \neq \emptyset$ is the type of the event.
- $s \in S = \{\text{ProcessEngine_XY, Service_XY, CEP_XY, …}\}$ where $s \neq \emptyset$ is the source that generates the event.
- $ts$ is the timestamp at which the event occurred where $ts \neq \emptyset$.
- $d$ is the destination of the event. Whether such an attribute is necessary to be determined later, we may also generate events for the whole environment by using the publish/subscribe mechanism for messages of an ESB.
- $P = \{p_1, …, p_n\}$ is a set of properties (e.g. event message header properties such as correlation data, process instance identifier or similar).
• B is the optional body of the event message (e.g. containing human readable messages like “Division by zero in line 1043…” or similar).

This can be seen as a generic Event model that applies to whatever syntax we choose for the serialization of the events. Currently the Apache ODE emits events in the Common Base Event (CBE) model developed by IBM. In the IBM Developer’s Guide [IBM04] (available at http://www.ibm.com/developerworks/autonomic/books/ffpy0mst.htm) the concrete XML-Schema, as well as, examples are provided. This CBE model is a super set of the generic event model described above, it is more expressive (and detailed) than the generic COMPAS that is made up of source, target, timestamp etc. Hence, we can use it and express everything we want, as the COMPAS event model is a subset of this model. Therefore in COMPAS we plan to use those parts of the model required for compliance monitoring and mining.

4.2. Event generation

In the compliance governance architecture described in Figure 1 and explained in more detail in [DA.1] as well as in [DCD+09], we can distinguish three main producers for the generation of events. First, the Process engine describes the current status of the execution of processes by generating and emitting execution events that indicate certain steps within a running process instance (see terminology section of [D7.1]). Second, the Instrumented services that are orchestrated within the process produce events related to the action that has been invoked at this service. Third, the Runtime rule evaluator and Business protocol monitoring components correlate events in order to emit events of their own.

We make a distinction in the category of events emitted by the first two event sources and those events emitted by the third event source. The first two sources of events emit so called system level events which we consider as events representing actions that occur within the system and system components. The third source of events emits the so called business level events that represent events of interest from the business standpoint. These are explained in detail in our definition of terms.

A Process engine describes the current status of the execution of processes by generating and emitting execution events. For COMPAS we have decided to use the open source BPEL engine Apache ODE for several reasons. Compared to other open source BPEL engines, Apache ODE provides the most complete support for the BPEL 2.0 specification [OASIS07]. Furthermore the community around this tool is quite active, the code documentation is sufficient, and we have gained expertise with this tool.

According to the user documentation of the Apache ODE project (http://ode.apache.org/user-guide.html), Apache ODE supports various types of events which are relevant for the COMPAS approach:

• ActivityEnabledEvent: An activity has been enabled
• ActivityDisabledEvent: An activity has been disabled (due to dead path elimination)
• ActivityExecStartEvent: An activity has started its execution
• ActivityExecEndEvent: An activity has terminated its execution
• ActivityFailureEvent: An activity has failed
• NewProcessInstanceEvent: A new process instance has been created
• ProcessCompletionEvent: A process instance has been completed
• ProcessInstanceStartedEvent: A process instance has been started
• ProcessInstanceStateChangeEvent: The state of a process instance has changed
• ProcessMessageExchangeEvent: A process instance has received a message
• ProcessTerminationEvent: A process instance has been terminated
• ScopeCompletionEvent: A scope has completed
• ScopeFaultEvent: A fault has been produced in a scope
• ScopeStartEvent: A scope has been started
• VariableModificationEvent: The value of a variable has been modified
• VariableReadEvent: The value of a variable has been read

For the COMPAS solution however these events need to be extended to account for specific requirements related to compliance. For traceability reasons the events related to activities and process fragments that contained in the process model will be extended with Universally Unique Identifiers (UUIDs) [ITU-T04] in order to allow drill down to root cause in case of violations. As Apache ODE is an open source product, such an extension can be made with manageable effort. For supporting the implementation of extensions and modification we can make use of the developer guide which available at http://ode.apache.org/developer-guide.html. The events emitted by Apache ODE comply with the CBE model developed by IBM.

**Instrumented services** may produce events containing information regarding the (service) operations that have been invoked, as well as the parameters with which these operations have been invoked. Essentially, such events are defined as the MethodInvocationEvent type. The code to produce such events may either be embedded directly in the code of the service or attached to the service as a wrapper. For our approach, the current choice is to use a service wrapper that can possibly be generated automatically. It is hoped that this has the advantage of separation of the service implementation code from event related code while at the same time enabling legacy code to also emit events by attaching to it event creation wrappers. However, the events produced by a service are not only limited to the above mentioned type but depend on what the implementing organization is interested in monitoring.

**Runtime rule evaluator** and **Business protocol monitoring** are complementary to components developed in WP3, which try to verify the universal compliance of the system to certain types of compliance rules at design time and independently of its execution, mainly by using model checking technique. The intention of runtime verification systems is to determine in continuous manner whether the current execution preserves some specified compliance rules at runtime; and their operation does not depend only on their theoretical capacities of checking some properties, but also widely dependent on several design and technical factors such as: their ease of integration with the system environment, the supported mode of their execution with the system and so on. So, in certain contexts, some runtime verification systems are more adequate than others.

Business protocol monitoring component brings new verification capacities, for instance: verification of temporal constraints, support of more complex type of data, different mode of execution with the system (e.g., online, post-mortem monitoring, and so on).

Business protocol monitoring component is an active component, which continuously listens to and interacts with other components, by receiving and sending events through the bus. It mainly listens to exchanged messages between the monitored system and its external partners, and produces different kinds of events, which can be classified in three categories: 1) Events regarding the execution of the component it-self: start/stop of its execution, 2) Internal events
of the execution of the business protocol, such as: Change of state, message reception, these kind of events can be used by other components to control or extend the Business protocol monitoring component.

3) Events raised in a explicit way (by programming), for instance in case of compliance violation or other situations.

Business protocol monitoring can be used in complementary way with the Runtime rule evaluator component. In such a scenario, Business protocol monitoring component can act as a low level verification system with respect to Runtime rule evaluator component. For instance, the business protocol monitoring can be used to check that a user does not ask more than one video at a time, and a video sent to the user corresponds to a certain resolution, and accordingly generate some events: e1, e2... which will be combined with other events by the Runtime rule evaluator component to determine the compliance to some rules.

4.3. Event processing

Of the components mentioned, the Runtime rule evaluator and Business protocol monitoring components are unique in the sense that they can act both as event producers and event consumers. The first contains a Complex event processing (CEP) engine. This receives a number of events and performs processing in near real-time to determine whether events or specific sequences/patterns of events have occurred. CEP should detect particular events or patterns of events, and also be the responsible for starting actions. In our case that action could be, for example, a notification that a pattern of events corresponding to a violation of a compliance concern has occurred. In order to perform this functionality, the CEP engine needs to be configured through a query language. The occurrence of this pattern of events is considered to be a business level event (i.e. a compliance violation is an event of interest at the business level). ESPER [ESP09] was the selected CEP tool to be adopted by the COMPAS solution.

With this language, one is able to specify the patterns of system level events that translate into particular business events of interest. In Table 19, for example, we illustrate some of queries that are used to identify patterns corresponding to the licensing requirements for our WatchMe reference scenario.

In addition to processing events and identifying patterns, the engine can be configured with listeners for particular patterns. The listeners are simply executable code that initiates an action when a pattern of interest is identified. These actions can, therefore, be programmed in any number of ways, for example, to send out a notification about the occurrence of a business level event to the dashboard for display.

<table>
<thead>
<tr>
<th>Name of the Compliance Requirements</th>
<th>Control</th>
<th>Info Required</th>
<th>Events</th>
<th>ESPER Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pay-per-view plan</td>
<td>The user ID subscribed for the plan can acquire only n possible streams at price p.</td>
<td>- The timestamp of each video playback (service invocation). - Provider id for each video stream.</td>
<td>- Video stream acquisition event generated by WatchMe business process.</td>
<td>SELECT count(*) AS numberOfPlaybacks, providerID, timestamp FROM VideoStreamAcquisitionEvent WHERE numberOfPlaybacks &gt; var_numberOfPlaybacks[providerID]</td>
</tr>
</tbody>
</table>
Time-based plan

The user ID subscribed for the plan can acquire any number of times any possible streams from StartDate till EndDate of the plan.

- The timestamp of each playback (service invocation).
- Provider id for each video stream.
- Video stream acquisition event generated by WatchMe business process.

SELECT providerID, timestamp FROM VideoStreamAcquisitionEvent (NOT timestamp in (StartDate:EndDate))

Composition permission

Only pre-defined combination s of video and audio providers are allowed (V1 with A1 or A2, V2 with A2).

- Timestamp.
- Video provider id.
- Audio provider id.
- Patterns of events generated by WatchMe based on user’s stream selection (e.g. V1 – event notifying that video stream 1 has been selected).

SELECT videoProviderID, audioProviderID, timestamp FROM pattern [not ((V1 and (A1 or A2)) or (V2 and A2))]

| Table 19 ESPER rules to monitor the compliance requirements of the WatchMe scenario |

4.4. Event logging

The Event log takes via the ESB all events emitted by the COMPAS runtime environment and stores them. 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. The communication between the ESB and the Event log is done by Logging Services. Figure 11 illustrates the interactions of these components and some details about the table log where the events are stored according to the Event model. More details about the specification and implementation of this component are present in the next paragraphs.
Figure 11 Architecture and implementation details of the Event log

The **Logging Services** are coded as web services in Java and published using Apache Tomcat 6 [Apa09] and Apache Axis 1.4 [Axi09], all widely used, open source, and easily available in Internet. So far, just three services are defined:

- **Subscribe({Et})**: In COMPAS the open source Enterprise Service Bus (ESB) Apache ServiceMix ([http://servicemix.apache.org](http://servicemix.apache.org)) is used, which is built from the ground up on the Java Business Integration (JBI) [TW05] specification JSR 208 ([http://jcp.org/aboutJava/communityprocess/final/jsr208/](http://jcp.org/aboutJava/communityprocess/final/jsr208/)). JBI is a Java-based standard that defines runtime architecture for plugins to interoperate via a mediated message exchange model, which was simply adopted from the WSDL 2.0 spec's message exchange patterns (MEPs). There are two types of plugins binding components and service engines, generically referred to as JBI components. For details concerning the usage of JBI components for integrating different components see the documentation of Apache ServiceMix available at [http://servicemix.apache.org/documentation.html](http://servicemix.apache.org/documentation.html).

- **Put(event)**: this service receives as an input the event that must be inserted into the Event log, more precisely in the Log Table. The service is developed to deal with more than one event per invocation. As an output the services sends after its execution a Boolean response (Yes, No) according to the status of the insert command. The Put service can be accessed by any user, without restrictions. Additional configurations in this link [http://compas.disi.unitn.it:8080/axis/Put.jws](http://compas.disi.unitn.it:8080/axis/Put.jws).

- **Get(sql)**: this service queries the Log Table based on PL-SQL commands [PL09], and returns all the record sets found. The output format of this service is simple, it consists on record fields values separated by commas (,). Different records are identified by semicolons (;). This service is currently available at: [http://compas.disi.unitn.it:8080/axis/Get.jws](http://compas.disi.unitn.it:8080/axis/Get.jws).

The **Event log** is developed with an open source version of Oracle 11g [Ora09]. Its implementation encompasses the log_table creation based on the Event model (Section 5.1) and the XML Event format. The pictures bellow (Figure 12 and 13) illustrate the structure and examples of the mentioned table.
5. The ETL process

The ETL process is considered the most important and cost intensive task in setting up and operating a data warehouse ([DA06], [SIL05]). Typically, the ETL process is composed of pieces of software responsible for extracting raw data from several heterogeneous sources, then transforming (e.g., cleaning, customizing), and loading them into a DW [VSG+03]. That is, the ETL process is responsible for transforming the data sourced from operative systems, whose data structures are typically designed to cater for high operative throughput and interactive querying needs, into data that can be loaded into a DW, whose data structure is designed to provide for fast reporting and analysis. In doing so, the ETL process must ensure well structured, consistent, and high-quality data, in order to assist decision making and, in our case, to report on compliance.

Figure 14 illustrates the ETL process that comprises: i) Application Integration Component, ii) Data Integration Component, iii) Warehouse, and iv) Presentation Components. The Application Integration Component automatically pulls out data (events) from heterogeneous sources (i.e., the Log event and Audit Trail). The extraction is made by wrappers, based on Event metadata and coordinated by the Extraction Routines. Then the Data Integration Component cleans and performs transformation routines to standardize the event data in the Data Staging Area (DSA), and afterwards it incrementally loads data into the DW to finally render them in the Presentation Component. The depicted black semi-circles are the web services used during the execution of the ETL routines. Such architecture is based in the Software Development Process Performance Data Warehousing (SPDW) solution; respective details can be found in [BRC+06]. Such solution is applied to capture software quality metrics in order to support software organizations to achieve capability and maturity certification, as CMMI (Capability Maturity Model Integration).

In Section 5.1 and 5.2, we present the Application and the Data Integration Components, as well as, the Events loading and the temporal aspects involved on that. Finally, in Section 5.3 we describe the dynamic aspects regarding the execution of the ETL process.
5.1. Application integration component

The ETL process starts by the execution of the Application Integration Component, which is composed of: Event metadata, Wrappers, and Extraction routine. The latter, based on the metadata information, extracts the events from two main data sources: Event log (Section 4.4) and Audit trail.

The Audit trail database is the storage of process execution information generated by the Process engine. It contains an entry for all relevant actions such as the start and the completion of an activity [LeRo00]. Such entries contain additional important information about the (engine-internal) events (e.g., input passed to the activity, the output produced by the activity and the time the events occurred). Such information is going to be used during the transformation step as a complement to the raw events extracted from the Event log. Note that what kind of information the Audit trail stores, depends on the specific implementation. In the case of Apache ODE (adopted in COMPAS) this database is embedded into the engine and only can be accessed from the outside via the web service interfaces it provides, which realize access in a pull-mode. For instance, the Apache ODE BPEL engine provides web service interfaces for management functions that can be used to access the Audit trail. Details for this interface are described at [http://ode.apache.org/bpel-management-api-specification.html](http://ode.apache.org/bpel-management-api-specification.html) and in the D5.2.

![COMPAS ETL process diagram](image-url)
5.1.1. Event metadata

The Event metadata contains information that determines which data must be extracted from the different sources: Event log and Audit trail. That way, it is possible to choose different sources of data, according to the environment where the solution is adopted without changing the whole Application Integration Component, since the ETL process can choose its own set of events, and define how information is structured in the desired sources. Each data source contains set of events and business data that should be captured, and for each of these sources, the metadata stores their access information (tool’s name, URL, Login and Password). Such information is used in order to connect the event sources and extracted the desired content. Listing 1 is an example of the first lines of the Event metadata defined for the Event log source.

The adoption of Event metadata facilitates event extraction and the modification of their characteristics (e.g. attribute names, origin table, and domain). Minor metadata modifications allow software development management variation without the need of modifying the other Application Integration Component, thus reducing the coding effort.

```xml
<?xml version="1.0" standalone="yes"?>
- <EventSources>
  <Source>
    <Name>LogEvent</Name>
    <URL>jdbc:odbc:ProjectServer</URL>
    <Login>COMPAS</Login>
    <Password>COMPAS</Password>
    - <Field>
      <Description>Log_ID</Description>
      <Table>Log_Table</Table>
      <Field_name>log_id</Field_name>
      <Type>decimal</Type>
      <Size>2</Size>
    </Field>
    - <Field>
      <Description>Event_Type</Description>
      <Table>Log_Table</Table>
      <Name>event_type</Name>
      <Type>VARCHAR</Type>
      <Size>20</Size>
    </Field>
  </Source>
...
```

Listing 1 Example of an Event metadata

5.1.2. Wrappers

Wrappers are packages that encapsulate one or more applications through a unique interface [ACK+04]. This solution determines that each event source has a specific wrapper, responsible for capturing events from their underlying databases or files (if necessary). Wrappers are defined here as web services and described according to the WSDL standard. Together with Event metadata, the proposed wrapper approach has the following advantages: (i) it allows the easy addition of new event sources, requiring only to define and publish new services and to update/create the metadata; (ii) it allows data extraction from different sources
executed in different operational systems; and (iii) it offers freedom regarding the programming language used for encoding the tool data extraction service.

5.1.3. Extraction Routine

Extraction Routine is the core of the Application Integration Component, which is responsible for coordinating wrappers invocation and handling: (i) access to different extraction sources; (ii) events extraction synchronism; (iii) events consolidation in a unique package; and (iv) extraction homogeneity and consistency. Thus, this routine finds and invokes different extraction wrappers, considering a defined order and synchrony, which enables consistently capturing data in a specific timestamp. Extraction Routine presents two distinct services layers, where the first one is used to coordinate the communication among wrappers and the second one interacts directly with the DSA for loading data into this temporary repository.

A bidirectional data flow and services layers among wrappers and Extraction Routine permit two types of ETL process execution: on demand or self-started. In the first type, the Extraction Routine is responsible for starting the extraction process, based on scheduled events or human triggering actions. The second type allows the starting of an extraction process by pro-active wrappers, which invoke these routines whenever the project environment detects updated values for its measurements are available. Therefore, service layers allow the ETL process to be always available for invocation, enabling a permanent data exchange between projects and data warehousing environment. Notice that, this approach is free of intrusion if the extraction is based on scheduled events or self-started, but its intrusion is low if triggered by a human-agent. In both types, the Extraction Routine invokes wrappers taking into account information stored in their WSDL documents and Events metadata, and has the autonomy to communicate with to the Application Integration Component, when appropriate.

5.2. Data integration component

The Data Integration Component is responsible for transforming and cleaning temporal data stored into the DSA, as a result of the Extraction Routine described in the previous section. The DSA is composed of a set of tables, where data extracted from sources are stored and pre-processed by Cleaning and Transformation Routines (CTR), considering the data model described in Section 4.1.2. The following transformation activities proposed by Kimball [Kim02] are applicable in this component:

- Solving domains through events: during events computation, conflicting domains are solved (e.g., the same event has two different sources that represent the same value, but are write in a wrong way in one of the events source), as well as missing data and value formats are handled, according to pre-defined rules and taking into account the information defined in the data model (e.g., source and destination of the events) and in the Compliance Requirements, Business Process Models, and Business Process Fragments;
- Combining data sources and verifying integrity among primary keys (e.g. mapping keys used by the tools from original data sources to dimensions keys);
- Creating surrogate keys for each dimension record, in order to avoid a dependency on legacy defined keys.
This ETL solution, based on [BRC+05], does not need human intervention to transform and store temporally the extracted data. All CTRs, including the Organizational Metadata access, can be coded as a service and can be automatically executed. However, in both cases CTR activation must obey to the two types of data flow adopted by the implementation solution. If by demand type is chosen, then these routines can be triggered by Extraction Routine invocation or another central external control (e.g., central framework). Otherwise, the self-started activation can be deployed, for instance, by a stored procedure or a pre-scheduled event.

### 5.3. Event loading and temporal representation of the facts

Event data must be available in an aggregated, consistent and up-to-date fashion in a centralized data repository that maintains historic records. That way, the available information can be used to support fast decision making. Kimball affirms that only new or updated data should be stored in the DW [Kim02]. To address these issues, we adopt validity time for facts, based on the Temporal Relational Model (TRM) [NA93] and in SPDW [BCD+06], in which data validity is defined using two temporal attributes: start validity time (ST) and end validity time (ET).

By applying a validity time to the data model, each fact record has a value for ST and ET. By the time the fact record is created, its ET is unknown (i.e., null). The ET is null until such data become invalid by the availability of more updated values for the event measurement and its ET receives a timestamp. By doing that, the repository does not store redundant data and maintains only new or updated facts, enabling an incremental loading process. Hence, this solution implements an incremental loading procedure to maintain the DW consistent and updated in the presence of frequent loads ([BDC+06], [Kim02]). In this way, it is possible to periodically assess compliance and to preserve a business/compliance performance history.

Two rules manage the incremental loading. Their main goal is to control data insertions, updates and temporal validity. A fact is an updated fact if its primary-key ST value is already present in the DW, with an older ST value, and there is at least one different measure value. Otherwise, it is a new fact.

- **Rule 1:** for a new fact, a new record should be inserted assigning the loading timestamp to ST and null to ET.

- **Rule 2:** for an updated fact, a new record should be inserted assigning the loading timestamp to ST and null to ET and the corresponding old record should be updated assigning the loading timestamp - 1 to ET.

The DW load is executed by web services, as depicted in Figure 14. The loading is executed after the data is correctly structured and consolidated inside the DSA, and the result is that the data is inserted into the DW. Among the benefits of this service-oriented approach, we can mention: (i) possibility of changing the warehouse without compromising the whole scenario; (ii) using the same loading service in similar environments; (iii) adding new event into the loading procedure by simply publishing a new service; and (iv) an intrusion-free loading WS invocation, without any human intervention. Regarding to this last benefit, the procedure execution can be on demand or self-started, depending on the solution adopted by other architecture components. That way, the loading process can be initiated by: a user invocation web interface, a pre-scheduled invocation, or a stored procedure synchronized to start after CTR completion.
5.4. Dynamic aspects of the ETL process

This section specifies the dynamic aspects involved during the execution of the COMPAS ETL process. Figure 15 illustrates these aspects by means of UML sequence diagram. The labelled arrows represent the messages changed between the objects instances (i.e., Extraction routine, Wrappers, CTR, and Event loading). Such labels contain brief text descriptions of the action done and of the parameters changed between the involved objects.

The ETL process starts with the execution of the Extraction routine, which first reads the Event metadata in order to know how to extract the row event data from the different sources. After that, this routine invokes all the necessary wrappers taking into account the information about the event sources listed in the metadata. Each source of event has a dedicated wrapper responsible of querying raw that from its respective source, according to the input parameters passed by the Extraction routine. The wrappers are depicted by many aggregated rectangles to illustrate their multiple existences. Every time a wrapper concludes its execution the Extraction routine loads the extract data into temporary tables located in the DSA. When all data were extracted, the Extraction Routines starts the CTR execution. At that time, a set of cleaning and transform routines are done until the data achieve model format supported by the DW. For that, some searches in the standard sources are necessary (i.e., Compliance requirements, Process Models, and Process Fragments) to capture complementary information needed by the CTRs. Finally, when the transformations are done the data are loaded into the DW by the incremental Event Loading and the ETL process is finished.

![Figure 15 The dynamic aspects of the COMPAS ETL process](image-url)
6. Real-time detection and analysis of the compliance deviations

This section concerns a problem of real-time detection and analysis of the compliance deviations. It is intended to deal with compliance violations in business processes, which should be detected in real-time. The purpose of such an instant detection would be to notify actors participating in the process about the violation and providing them with information on the cause of it. This would help them adjust the process so that it becomes compliant. This section deals only with the real-time detection of compliance violations and analysis of the cause of them. The automatic adjustment of the running processes is beyond the scope of the COMPAS project.

6.1. Overview of the Event-to-Compliance (E2C) architecture

In Figure 16 we present the E2C architecture for the detection and analysis of compliance violations. In Figure 1, the following architecture corresponds to the Runtime rule evaluator component. The concept originates from one of Telcordia’s projects described in [BGN+06].

The events are generated by the Business process engine and sent through ESB to the E2C software. In the first step events are contextualized, i.e. connected to proper event contexts where more data related to a specific type of event is stored (e.g. historical data, actors involved, etc.). In the next step, events are processed by a series of event operators (e.g. stream operators) and finally, if the compliance violation is detected, the proper alerts are generated and sent Compliance notification component in order to be visualized on a compliance violation dashboard. Compliance evaluation specifications are a set of predefined, customizable schemes (pipelines of event operators) for a detection of compliance violations.

![Figure 16 Event-to-Compliance (E2C) architecture of the Runtime rule evaluator component](image)

6.2. Event contextualization

Events must be cast in terms of a compliance concerns and a corresponding Compliance contexts, which are a set of resources, models, protocols and schemas that are relevant to one or more compliance concerns. Defining contexts is effectively a domain-specific modelling decision that identifies a logical structuring of information. It’s the first step in making the events understandable to the end users. Compliance contexts capture models, protocol, schemas that are relevant to one or more compliance concerns. E2C architecture will address
this problem by providing Event contextualization component with the purpose of maintaining one or more contexts that are populated with information gathered from event sources.

Many types of basic compliance contexts can be identified:

- **Process contexts**: The information stored in this type of contexts might include types of activities, control flow, data flow, resources used by the process, roles and status of the process. This type of context could support Role, Coordination, and Information exchange compliance.

- **QoS contexts**: It’s important for the evaluation of QoS type of compliance violations to store a data on both the current and past performance data. The other type of information stored in this type of context might be formulas for performance computation and predefined service quality attributes. This type of context could support Quality and Real-type compliance.

- **Resource contexts**: Resource contexts keep information on status, properties and utilization of various resources. This type of context could support Resource compliance.

- **Access control contexts**: This type of context captures information on subjects, resources and permission or denial decisions. This type of context could support Access control type of compliance.

- **Spatio-temporal contexts**: The time and space information is always a useful data. The information included in this context is: date, time, duration, situation in which event occurs and location of activities, resources and actors. This type of context could support Spatio-temporal compliance.

- **Event priority and severity contexts**: This type of contexts provides additional information about the priority of the given event and its severity.

- **Event source contexts**: These contexts help identifying the source which generates a given event. It contains a local and global identifier; where the former is an identifier provided by the source component of the event, while the latter is assigned by an event processing facility. These might be useful to distinguish between events in relation to a single business process or in relation to multiple organizational processes [D2.1].

Contexts in the Event contextualization component are the source of contextualized events to be published to the Compliance evaluation component. A Contextualized event is an event (re)cast in terms of the concepts represented in a context. Such an event may have parameters that identify the actors, their actions, and the spaces in which those actions occur during time. The Compliance evaluation component subscribes to the contextual event types of interest by reference to the event type and possibly constraints on event parameters. As these contextualized events are discovered or modified, the events of interest are published to the Compliance evaluation component.

Upon receipt of an event, the Event contextualization component must perform the following activities:

- Correlate event parameters and event ontological data with the information in relevant compliance contexts.
• Incrementally fuse the event with the information already present in the context for the specific entities related to the event. This results in an update to the context.

• Incrementally publish the resulting event(s) to subscribers.

6.3. Compliance evaluation

Compliance Evaluation component design is based on a concept of Event operators. The inputs and outputs of Event operators are streams of contextualized events, either primitive or composite. Event operators are computational units that create and emit more complex events (produced on its output stream) from a combination of simpler events (consumed on its input stream(s)). The following are the most important types of them:

• **Contextualized event operators**: can be customized to subscribe to and filter contextualized events

• **Compliance alert delivery operators**: submit alert events to users or/and compliance dashboards

• **Low-level stream processing operators**:
  - **OR**: computes a union of its input streams
  - **Difference**: computes a set of difference of input streams – outputs list of events that appear in the first stream and not in the second stream

• **Relational algebra operators**:
  - **Filtering**: culling of uninteresting events
  - **Joining**: combines related events from multiple sources into a composite event
  - **Grouping and aggregation**: grouping and aggregations of events

• **Statistical and sampling operators**:
  - **Sampling operators**: can be added to compute changes in rate of occurrence of a specific event type
  - **Statistical operators**: can be introduced to utilize learned patterns of normal behaviour to detect statistical anomalies

The E2C architecture will provide a tool – graphical editor – to enable a user to create complex Event operators (operators’ streams) from a palette of operator types, interconnect the event operators’ input and output event streams, and customize the operators’ computational behaviour via dialog boxes. These complex event operators streams form computational pipelines and transform primitive events into alerts delivered to the Compliance notification component. They constitute Compliance evaluation specifications (Figure 17).

In COMPAS project, the Compliance evaluation component will enable definition of custom rules for compliance evaluation by combining event operators into arbitrary streams.

The E2C architecture will also include an engine which will execute such operator streams. It has to be capable of performing all the low-level operations on the event streams, so an open-source event processing engine such as ESPER will be employed for this purpose.
6.4. Compliance notification

Compliance notification component delivers alerts of compliance violation to the COMPAS Enterprise Service Bus. The alerts then can be intercepted by any COMPAS component connected to the bus, e.g. COMPAS compliance dashboard (described in the dashboard will include a detailed specification of the violation. The dashboard will be a separate one developed by TARC-PL or the one prepared by UNITN will be used. It will not only provide basic alerts, but also enable for an in-depth analysis of the compliance violation – drill-down through various contexts thus providing the Compliance Officer with enough information to redesign the process, so that it becomes compliant.

6.5. Example of real-time detection of compliance violation

The example presented in this section concerns the case study of WatchMe – a media streaming service offered by MVNO. It presents run-time QoS compliance violation detection process. The streams delivered to the end users must be delivered with appropriate service delivery rate. Therefore, near real-time detection of such situations must be provided, so that MVNO can be notified of the performance drop-downs and react properly. Figure 17 presents the Compliance Evaluation model which will be specified in a graphical editing tool used by a company compliance expert. Such a Compliance Evaluation model might be also automatically generated from the Compliance Evaluation Specification based on e.g. ODRL-S licenses (ODRL-S language is described in Section 7 of this deliverable).

The model represents various kinds of system code – either in ESPER Event Processing Language (SQL-like language for event stream processing) or in Java. The diagram shows pieces of the code matched with appropriate operator blocks. Users of the tool might add their own parameterized operators to specify e.g. various filtering or grouping queries, temporal algebra based pattern-matching rules or basic mathematical algebra and logical calculations. In the example presented in the diagram (Figure 17), two event streams are flowing into the pattern matching operator, which detects patterns of the end delivery events preceded by the start assembly events. Next, appropriate successful and error deliveries are filtered in two parallel flows of operations. The events are counted and the results are used for calculation of service delivery rate by the calculations operator. Finally, the alert delivery operator is responsible to run the code sending appropriate compliance violation alert (if the service delivery rate is below required threshold) to the COMPAS ESB. This alert is then intercepted by COMPAS compliance dashboards where it’s immediately visible, thus any compliance resolution tasks can be performed.

The model can be also modified to calculate the service delivery rate only for some specified time interval. Such a change can be easily introduced, as ESPER EPL language allows for specification of time windows for the queries run on each stream. In the graphical model it will be specified by optional parameters for the operators.
7. A model and DSL for compliance to licenses

7.1. Introduction

This part of the deliverable presents a model and a DSL for expressing compliance requirements related to licenses. The licensing DSL is one of the first DSLs developed in the COMPAS project, so it would be natural to put here the overall picture about the purpose of the DSLs and their role in COMPAS. This is, however, explained in [D1.1]. Therefore, here we limit ourselves to a short description of what DSL is.

A DSL is a modelling language tailored for some domain. In our case, the domain is licenses, or, more specifically, service licenses. According to Stahl and Völter [SV06], a DSL’s purpose is to make the key aspects of a domain formally expressible and can be modelled. So, the goal of the DSL for licenses is to make service licenses expressible and possible to model by non-experts in a simple manner. In COMPAS, compliance requirements expressed in the DSL for licenses are translated into ESPER rules. The rules are checked by the Runtime rule evaluator component.

Apart from the presentation of the DSL, the goal of this part is to illustrate how a domain-specific compliance requirement can be expressed. We will use the WatchMe reference scenario from Section 1.4 to show how to use the licensing DSL to represent a license.
7.2. License profile

Before presenting a model and a DSL of service licenses, let us introduce the license profile (Figure 18). This profile defines stereotypes we will use in the model. All stereotypes extend the metaclass Classifier from the UML 2.0 metamodel [OMG07]. Figure 18 shows that each document consists of one or several parts, and each part contains several elements that could be of Expression Language (EX) or Data Dictionary (DD) type (explained in the following subsections) and can contain sub-elements and attributes. Each attribute has a value.

![Figure 18 License profile](image)

7.3. A model and a DSL for service licenses

The law of several countries\(^4\) protect the intellectual work performed as the expression of an idea in a text, image, sound, and also computer programs. In general, copyright law states that an author has all the so-called exclusive rights attached to her work. The license is the legal instrument used by copyright laws to describe the rights granted to software users by software author.

What is a service license? In this section we extend the notion of license to services, exploring the different requirement imposed by service nature being different from software. At the same time, we propose a formalization of service licenses by means of a model and a DSL.

A license \(L(S)\) for a service \(S\) consists of a finite set of license clauses\(^5\), which are defining context of the license (Subject clause), its permissions (Scope of rights and Evolution clauses), and requirements (Limitation of liability, Indemnity, Warranty and Financial terms...)

---

\(^4\) under the Berne Convention and the World Trade Organization Agreement on Trade-Related Aspects of Intellectual Property Rights

\(^5\) In the terms of ODRL, the language we use as the basis for the DSL, license clauses are also referred to as models. However, whenever possible, we try to not use the term “model” in the sense “license clause” to avoid confusions. When such use is inevitable, we use it as “ODRL model”.
Each clause (see license model in Figure 19) further consists of a set of DD and EX elements.

Elements can contain other elements that can give rise to an arbitrarily deep hierarchy of elements within elements. Elements can have attributes which are specified generally with name value pair in an element’s open tag. To not overload the figure, in Figure 19 we present details only for the Limitation of liability clause, but in the following subsections we further describe types of clauses and present a model for each.

To express service licenses we extend an existing language, Open Digital Rights Language (ODRL) [Ian02]. ODRL purpose is

to provide flexible and interoperable mechanisms to support transparent and innovative use of digital resources in publishing, distributing and consuming of electronic publications, digital images, audio and movies, learning objects, computer software and other creations in digital form. The ODRL has no license requirements and is available in the spirit of "open source" software. [Ian02]

ODRL has following three core entities.

- **Assets**: Resource being licensed (identified uniquely).
- **Rights**: Rules concerning Permissions (the actual usages or activities allowed over the assets), Constraints (limits to these permissions), Requirements (the obligations needed to exercise the permission), and Conditions (exceptions that, if become true, terminate the permissions and re-negotiation may be required).
- **Parties**: Information regarding end users and right holders, including the service provider, consumer, broker etc.,

With these three entities, ODRL expresses Offers (proposals from rights holders for specific rights over their assets) and Agreements (contracts or deals between the parties, with specific
offers). Together these core entities, ODRL allows for a wide and flexible range of expressions to be declared.

Our intentions for selecting ODRL as the basis for describing service licenses are as follows.

- ODRL is an open standard for expressing rights information.
- Defined in XML, ODRL provides syntactic and semantic interoperability which are supported by XML serialization
- Several business scenarios across various domains can be expressed in ODRL.
- Published by the World Wide Web Consortium (W3C), ODRL has wide acceptance.
- ODRL is supported by several industries and consortia like the Dublin Core Metadata Initiative (DCMI)\(^6\) and the Open Mobile Alliance (OMA)\(^7\).

We extend ODRL to express clauses for service licensing, named as ODRL-S, making a machine interpretable service license [ODRL-S, Gan08]. Diagrammatically, a license in ODRL-S (see Figure 20) is represented as a container for the clauses (shown in rectangles) and arrows represent the mapping for ODRL-S elements adopted and/or extended from the corresponding ODRL license clauses (models\(^8\)). We reuse the ODRL Asset model to represent the Subject clauses of a service license and the Payment model to represent the Financial terms of a service license. The ODRL Rights model is extended to define the Scope of rights clauses of ODRL-S. A set of new terms are defined in ODRL for representing Warranties, Indemnities, and Limitation of liability clauses of a service license. Evolution clauses are newly defined to represent the rights of a service over future versions/releases. In the following subsections we present models of ODRL-S clauses using graphical and textual (xml/xsd) syntax. We will also explain semantics of clause elements.

\(^6\) http://dublincore.org/
\(^7\) http://www.openmobilealliance.org/
\(^8\) here ODRL * model refers to ODRL * license clause. We call it “model” because it is called so in the ODRL terminology and we keep the term model when it is strongly collocated with ODRL
Figure 20 Conceptual mapping of service licensing clauses in ODRL-S

The graphical models of ODRL-S license clauses adopt the syntax shown in Figure 21 that covers both the Expression language entities\(^9\) (shown as shadowed rectangles with "EX" in the top left corner of the rectangle) and the Data dictionary elements\(^{10}\) (shown as shadowed rectangles with "DD" in the top left corner of the rectangle). The DD elements are contained in the EX elements (indicated by arrows). Arrows in graphical models correspond to the containment relations.

\[ \text{Figure 21 Graphical syntax of license clause models} \]

\(^9\) See [http://www.w3.org/TR/odrl/#82855](http://www.w3.org/TR/odrl/#82855) for the XML schema

\(^{10}\) See [http://www.w3.org/TR/odrl/#18482](http://www.w3.org/TR/odrl/#18482) for the XML schema
7.3.1. Subject model and syntax

The Subject clause of a service license could adopt the ODRL Asset model [Ian02]. The Subject license clause will contain following elements (Figure 22).

- **UID**: a unique identification code for the service.
- **Name**: a name used to describe the service.
- **Remark**: comments related to the service.
- **Version**: indicates the version of the service.
- **Date**: indicates the date the service offered or is valid for.
- **External reference**: a link (URI) to additional information about the service.
- **Service**: a link (URI) to the service.

The Subject clause of ODRL-S is based on the Data dictionary schema and elements of Asset model as defined in ODRL and we do not reproduce it here. ODRL uses the ODRL Context model to identify the asset and declare the asset uniquely.

7.3.2. Scope of rights model and syntax

In a service license, the Scope of rights describe on what could be done with the service. Given the general provision of copyright law, it is not strictly necessary to describe prohibitions associated with a specific asset, while it is necessary to describe in full details the set of allowed activities (Permissions), as well as the associated Conditions, Constraints, and Requirements.

![Figure 22 ODRL-S Subject license clause](image-url)
In ODRL-S the Scope of rights license clause is extended from ODRL Permission model, with the associated Condition, Requirement, and Constraint license clauses (see Figure 23). The Permissions considered relevant for services are associated with Composition, Derivation, and Adaptation of services. In other words, a Permission can be expressed in terms of Adaptation or Composition or Derivation, and contains a set of Conditions, of Constraints, and Requirements. Requirements could be Sharealike or Attribution.

We represent the set of operations (listed in the service interface) of a service $S$ by $O(S)$ and a single operation by $o$ where $o \in O(S)$. We denote a service license by $L(S)$. We refer to a service as independent service that will execute in a different context or that is owned and/or maintained by a different organization. With the symbol $\xrightarrow{dep}$, we denote the dependence relation of an operation (or a set of operations) in the left side on other service operation(s) in the right side.

From the perspective of rights expression, we describe in this section the rights associated with Permissions (Figure 23).

### Table 20 ODRL-S Data dictionary semantics and schema for Scope of rights

<table>
<thead>
<tr>
<th>Name</th>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptation</td>
<td>adaptation</td>
<td>The right of allowing the use of interface of a service only (independent execution).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;xsd:element name=&quot;adaptation&quot; type=&quot;o-ex:permissionType&quot; substitutionGroup=&quot;o-ex:permissionElement&quot;/&gt;</code></td>
</tr>
<tr>
<td>Composition</td>
<td>composition</td>
<td>The right of execution with the right of interface modification (dependent on the execution of services being composed).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;xsd:element name=&quot;composition&quot; type=&quot;o-ex:permissionType&quot; substitutionGroup=&quot;o-ex:permissionElement&quot;/&gt;</code></td>
</tr>
<tr>
<td>Derivation</td>
<td>derivation</td>
<td>The right of modifications to the interface and the implementation of a service (with independent execution).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;xsd:element name=&quot;derivation&quot; type=&quot;o-ex:permissionType&quot; substitutionGroup=&quot;o-ex:permissionElement&quot;/&gt;</code></td>
</tr>
</tbody>
</table>
Composition is the right of execution with the right of interface modification. A service $S$ is said to be composite if its operations depend on operations belonging to $n$ other services, represented as $O(S) \xrightarrow{\text{dep}} \{o_f : o_f \in O(S), i=[1,\ldots,n]\}$.

Derivation is the right of allowing modifications to the interface as well as the implementation of a service. Derivation of a service, inspired by Free and Open Source Software (FOSS), is a novel aspect of creating a new service from existing service, modifying the service interface and service implementation. A service $S'$ is said to be derived from another service $S$ if $O(S') \subseteq O(S)$ on satisfying the following two conditions.

1. To exist $S'$, $S$ should be a Free/Open Service and
2. $S$ and $S'$ are independent in execution.

Furthermore, Derivation requires independent execution of the service, whereas Composition is dependent on the execution of services being composed.
Adaptation refers to the right of allowing the use of interface only (executed independently).

We use the term ‘Adaptation’ to signify the making of a new independent service from an existing service interface without modifying the implementation. A service $S$ is reproduced as another independent service $S'$ by Adaptation if

1. $O(S') \neq O(S)$ and
2. $S$ and $S'$ are independent in execution.

For example, in case of Adaptation, the interface of the new service could provide only a (simplified) part of the original interface, or translate it in a different human language/notation (metric system vs. English units). Differently from the Composition case, the two service executions are completely independent.

ODRL-S reuses the concept of Sharealike and Non-commercial use of the Creative Commons profile [Ian05]. Attribution to services is facilitated by ODRL.

We present the Data dictionary semantics and schema for Scope of rights in Table 20.

<table>
<thead>
<tr>
<th>Name</th>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payment</td>
<td>payment</td>
<td>The amount of the payment containing <em>amount</em> (must be a positive decimal to two decimal places), <em>currency</em> (must use ISO4217 codes), <em>taxpercent</em> (must be a positive decimal between 0 and 100 (inclusive)), <em>code</em> (can be any standard tax identifier).</td>
</tr>
</tbody>
</table>

<xsd:element name="payment" type="o-ex:requirementType" substitutionGroup="o-ex:requirementElement"/>

Table 21 ODRL-S Data dictionary semantics and schema for Financial terms
7.3.3. Financial terms model and syntax

We represent the Financial terms license clause (see Figure 24 adopted from [Ian02]) in ODRL-S using ODRL Requirement\(^{11}\).

ODRL-S reuses the definitions of Payment terms described in ODRL for defining the Financial terms license clause. Fee indicates a set of Requirements for Payments of usage, realized with the following elements [Ian02].

- **Prepay**: The amount due prior to the granting/use of the rights.
- **Postpay**: The amount due after the use of the rights.
- **Peruse**: The amount due for each use of the granted rights.

Fee is represented as an abstract entity depicted as a cloud in Figure 24 used to group similar Requirements.

The definitions for the Financial terms license clause of ODRL-S are described in Table 21 (adopted from ODRL [Ian02]).

7.3.4. Warranties, Indemnities, and Limitation of liabilities. Models and syntax

The WIL license clause\(^{12}\) defines Warranties, Indemnities and Limitations of liabilities associated with services.

---

\(^{11}\)The rights requirements in ODRL is the recognized set of preconditions that must be met in order to obtain the related permissions

\(^{12}\)The rights requirements in ODRL is the recognized set of preconditions that must be met in order to obtain the related permissions
As warranties may describe several terms on quality of issues and performance parameters, we have attempted to capture certain important criteria. The captured terms conform to general standards of SOC. Generally, the legal terms of indemnities and limitation of liabilities are difficult to interpret in machine interpretable form. However, the terms represented here are an endeavour for representing machine interpretable legal terms.
A Warranty is a promise regarding the description of services and their quality, as stated by the service provider. Warranties are represented in Figure 25 for the detailed descriptions of Warranty elements).

- **Performance**: specifies elements relating to quality of services based on temporal dimensions.
  - *Mean response time*: the mean time between the moment a request is sent and the moment that the response has been provided to the consumer.
  - *Mean process time*: the average duration of a service execution.
  - *Mean latency*: the average round-trip time between sending a request and receiving the response.
  - *Mean throughput*: a measure of the average amount of service that is provided.

- **Compliance**: specifies a set of quality aspects of the service in conformance with the rules, the law, compliance with standards, and the established SLA [MLM+06].
  - *Resolution rate*: the average time for resolving problems related to service provisioning.

- **Reliability**: specifies a set of technical measures related to hardware and/or software configuration of services and the network connections between the service requesters and providers [ZBD+03].
  - *Max utilization*: the performance measure of a service related by throughput and response time.
  - *Mean avail ability rate*: the probability of the service being accessible.
  - *Mean SLA violation rate*: the mean occurrences of breach of SLA clauses (including violations by providers and consumers).
• **Monitoring**: specifies measures that support management of services.
  
  – **Downtime frequency**: the frequency at which a service provider verifies the availability of a service.

*Third party infringements claims*, a clause of Indemnity, represents the statement provided by the licensor to the licensee to protect against the claims of a third party if any infringements over the intellectual rights (see Figure 26).

The clauses of Limitation of liability are described in ODRL-S as follows (see Figure 27).

• **Network Errors**: The license will not be liable for problems with the network.
• **Infrastructural Errors**: The license will not be liable for problems caused by the infrastructure.

The Data dictionary and schema for the WIL license clause are as shown in Table 22.

<table>
<thead>
<tr>
<th>Name</th>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warranty</td>
<td>warranty</td>
<td>An implication for the quality of service as provided by the licensor.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;xsd:element name=&quot;warranty&quot; type=&quot;o-ex:requirementType&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>substitutionGroup=&quot;o-ex:requirementElement&quot;/&gt;</td>
</tr>
<tr>
<td>Indemnification</td>
<td>indemnity</td>
<td>A defense for the licensee against allegations from a third party.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;xsd:element name=&quot;indemnity&quot; type=&quot;o-ex:requirementType&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>substitutionGroup=&quot;o-ex:requirementElement&quot;/&gt;</td>
</tr>
<tr>
<td>Limitation of Liability</td>
<td>limitationofliability</td>
<td>A limitation of liability taken for the service terms.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;xsd:element name=&quot;limitationofliability&quot; type=&quot;o-ex:requirementType&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>substitutionGroup=&quot;o-ex:requirementElement&quot;/&gt;</td>
</tr>
</tbody>
</table>

Table 22 ODRL-S Data dictionary semantics and schema for WIL

7.3.5. **Evolution model and syntax**

We define the Evolution license clause (see Figure 28) to specify the details of modifications by future releases or versions. The Evolution license clause elements are described as follows.

• **Maximum upgrades**: specifies the allowed number of upgrades\(^{13}\) to a service before the license becomes invalid.
• **Maximum versions**: specifies the allowed number of versions of a service before the license becomes invalid.

\(^{13}\) Another possible model for granting the usage of upgrades of a service is with the specification of a time period (for example, the license for a service X could require a payment, and could grant unlimited usage of current version and one year usage of any future versions; another one could grant unlimited usage of current version but new version will only be included for usage until 2010; and so on)
• **Substitutable**: indicates that the service can be substituted for by another similar service of same scope to which the licensee is allowed to use.

• **Generic**: indicates that the service can be replaced by a service of broader scope and the licensee retains the rights over the generic service.

The definitions for the Evolution license clause of ODRL-S are described in Table 23.

![Figure 28 ODRL-S Evolution license clause](image)

<table>
<thead>
<tr>
<th>Name</th>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evolution</td>
<td>evolution</td>
<td>Use of a service over time as physical environment and functionality change.</td>
</tr>
</tbody>
</table>

```xml
<xsd:element name="evolution" type="o-ex:permissionType"
substitutionGroup="o-ex:permissionElement"/>
```

**Table 23 ODRL-S Data dictionary semantics and schema for Evolution**
7.4. DSL for service licensing in the WatchMe Scenario

In this subsection we provide an example of how to use DSL for licensing to express the compliance requirements stated in the WatchMe scenario (Section 1.1.1). In such scenario each video provider has its own license, defining the possible combinations with audio allowed by the license contract between the video and audio providers, called Composition permission (Table 1). Additionally, each one of the licenses has to include the different plans that WatchMe offers to customers. For example, Video Provider1 has two licenses expressed in the Licensing DSL textual syntax: Pay-view Plan (Listing 2) and Time-based Plan (Listing 3).

The specification of the Video Provider1 Per-view license states that a customer assigned in that plan can acquire only $n$ possible streams at price $p$ (lines 18-21). Besides that, it also involves the Composition permission, which declares that video from Video Provider1 can be combined with audio from audio provider 1 or from audio provider 2 before streaming video and relating audio stream to customer (lines 38-41).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<o-ex:rights xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
    xmlns:o-dd="http://odrl.net/1.1/ODRL-DD"
    xmlns:mvno="http://watchMe.net/streaming-services"
    xmlns:sl="http://www.w3.org/2007/XMLSchemaList">
    <o-ex:offer>
        <o-ex:asset o-ex:id="watchMe-video-service">
            <o-ex:context>
                <o-dd:uid>urn:watchMe:service:watchMe-Provider1-PerUse_service</o-dd:uid>
                <o-dd:version>1.0</o-dd:version>
            </o-ex:context>
            <o-ex:permission>
                <sl:composition/>
            </o-ex:permission>
            <o-ex:permission>
                <o-dd:play>
                    <o-ex:requirement>
                        <o-dd:plan>
                            <o-dd:type>Pay-per-view plan</o-dd:type>
                            <o-dd:amount o-dd:currency="EUR">p</o-dd:amount>
                        </o-dd:plan>
                    </o-ex:requirement>
                    <o-ex:constraint>
                        <o-dd:unit o-ex:type="watchMe:NumberOfStreams">
                            <o-dd:count>
                                n
                            </o-dd:count>
                        </o-dd:unit>
                    </o-ex:constraint>
                    <o-ex:requirement>
                        <o-dd:combinations>
                            <o-dd:type>ApprovedAudioProvider1</o-dd:type>
                        </o-dd:combinations>
                    </o-ex:requirement>
                </o-dd:play>
            </o-ex:permission>
            <o-ex:permission>
                <o-dd:play>
                    <o-ex:requirement>
                        <o-dd:combinations>
                            <o-dd:type>ApprovedAudioProvider2</o-dd:type>
                        </o-dd:combinations>
                    </o-ex:requirement>
                    <o-ex:constraint>
                        <o-dd:ApprovedAudioProviders>
                            <o-dd:AudioProvider1>A1</o-dd:AudioProvider1>
                            <o-dd:AudioProvider2>A2</o-dd:AudioProvider2>
                        </o-dd:ApprovedAudioProviders>
                    </o-ex:constraint>
                </o-dd:play>
            </o-ex:permission>
        </o-ex:asset>
    </o-ex:offer>
</o-ex:rights>
```
Figure 29 illustrates in a graphical way by means of license clauses the same concerns expressed using textual syntax.

The specification of the Video Provider1 Time-based license states that a customer assigned in that plan can acquire any number of times any possible streams from StartDate till EndDate of the plan (lines 18-21 and 45-48). Besides that, it also involves the Composition permission, which declares that video from Video Provider1 can be composed with audio from any audio providers (lines 32-35).

Listing 2 Licensing DSL of Provider1 - Pay-view plan

```xml
<?xml version="1.0" encoding="UTF-8"?>
<o-ex:permission xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
                 xmlns:o-dd="http://odrl.net/1.1/ODRL-DD">
  <o-dd:play/>
  <o-ex:permission/>
  <o-ex:offer/>
  <o-ex:rights/>
  <o-ex:requirement/>
  <o-ex:constraint/>
  <o-ex:plan/>
  <o-ex:unit/>
  <o-ex:constraint/>
  <o-ex:plan/>
  <o-ex:requirement/>
  <o-ex:constraint/>
  <o-ex:combination/>
  <o-ex:setOfAudioProviders/>
  <o-ex:audioProvider1/>
  <o-ex:audioProvider2/>
</o-ex:permission>
```
8. Conclusions

In this document we have described the final goal-oriented model for warehousing compliance and business data. The data model supports business and compliance analysis. We have also presented and described interactions between the components of the compliance
governance runtime architecture. Finally, we have provided a model and a DSL of compliance to licenses.

9. Reference documents

9.1. Internal documents


[D5.2] “Initial goal-oriented data model”, ver. 1.0 of 2008-12-30.


9.2. External documents


