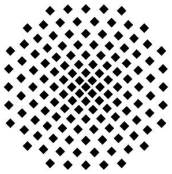

Universität Stuttgart

Fakultät Informatik, Elektrotechnik und Informationstechnik

A Prototype for View-based Monitoring of BPEL Processes

David Schumm, Gregor Latuske, and Frank Leymann

Technical Report 2011/04
March 15, 2011

Institut für Architektur von Anwendungssystemen

Universitätsstr. 38
70569 Stuttgart
Germany

CR: C.2.4, D.2.2, H.4.1, H.5.2, H.5.3

Abstract

This report describes the initial version of a tool for business process monitoring based on process viewing techniques. The tool, Business Process Illustrator (BPI), has been developed in the course of a Diploma Thesis which has been conducted at the Institute of Architecture of Application Systems. BPI is a Web-based tool for monitoring the execution of business processes. It displays the current state of a process instance in form of a process graph which is refreshed regularly. The initial version of the prototype supports regular process monitoring of processes based on the Business Process Execution Language (BPEL), plus process view transformations to reduce complexity and to ease analysis of process instances.

1 Background

In previous work, we identified the elementary forms of process view transformations based on literature study and evaluation of products [1]. From this effort, *process viewing patterns* could be obtained which provide the fundamentals for the definition of process views. In the work of [2], we built on this knowledge and described how views on processes can be used to support compliance management. Compliance refers to the entirety of all measures that need to be taken to adhere to requirements coming from laws, regulations, and internal policies [3]. The process views for compliance that we proposed in [2] consider the extraction, highlighting and hiding of process structures which are related to compliance.

The tool we present in this report builds on this previous research and implements common concepts of process views and business process monitoring, and it also provides concepts which are related to compliance management in business processes. With respect to process views for business process monitoring, the key features of the tool are (i) the omission of structures which are of little importance for understanding, (ii) the highlighting of process structures which are of high importance for particular analytical tasks, (iii) the highlighting of the execution path, and (iv) the visualization of the performance of service invocations using color grading. With respect to compliance management, the key features are the highlighting and hiding of process structures that are related to particular compliance requirements.

2 Process Views for Business Process Monitoring

In the following, we discuss the initial version of a tool for business process monitoring based on process viewing techniques. The tool has been developed in the course of a Diploma Thesis [4] which has been conducted at the Institute of Architecture of Application Systems. The source code, binaries, and an installation manual of the tool are available for download at SourceForge under the project name *Business Process Illustrator*[1], under Apache 2 license.

The Business Process Illustrator (BPI) is a Web-based tool for monitoring the execution of business processes. It displays the current state of a process instance in form of a graph which is refreshed regularly (online process instance monitoring). The initial version of the prototype supports regular process monitoring of processes based on the Business Process Execution Language (BPEL) [5], plus process view transformations to reduce process complexity and to ease analysis of process instances.

[1] Business Process Illustrator, http://sourceforge.net/projects/bpi/, 2010.

2.1 Architecture and used Technologies

BPI utilizes a Tomcat[2] application server (Java EE) as server environment. It can be accessed via any Web browser that supports Scalable Vector Graphics[3] (SVG). For the user frontend AJAX[4], JSP[5], Servlet[6] and JSF[7] are used. As process engine the Apache ODE[8] BPEL engine is supported out of the box. The initial choice of technologies in favor of Java and SVG was due to their successful usage in the Web-based custom monitoring tool in Java CAPS [10]. This tool is a showcase for Web-based monitoring of BPEL processes and it also provides simple viewing functions like the omission of `<assign>` activities and the modification of activity shapes. However, mainly due to limitations in the support of the `<flow>` construct is has not been used as a basis for development.

Figure 1 shows a generic architecture for business process monitoring on different levels of abstraction. This architecture is designed to support two different forms of process monitoring. Firstly, it support regular process monitoring of low-level models based on BPEL, plus abstraction of these models using process view transformations. In its current version, BPI fully supports this functionality and thereby forms the foundation to implement the entire architecture. The second functionality considers process monitoring across the boundaries of process models and process languages using state projections. For details regarding the architecture and the concept of state projections please refer to [6]. As depicted in Figure 1, support for state projections is ongoing work; this functionality is going to be added in the next version. Currently, the tool shows the general feasibility of process views for business process monitoring, especially the abstraction of complex business process instances to ease understanding.

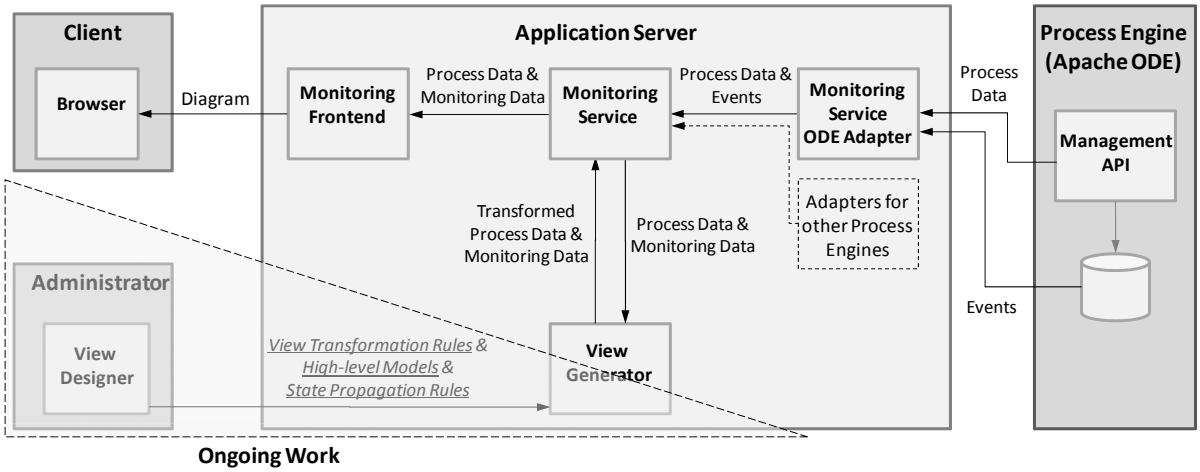

Fig. 1. Generic architecture for business process monitoring on different levels of abstraction [6], with respect to the current status of the implementation.

To be more precise about ongoing work, state projections from BPEL process instances to "Chevron" processes is going to be supported in order to implement the architecture completely. Chevron processes, for example available for process design in Microsoft PowerPoint[9], provide a high-level view of processes, similar to value chains. Figure 2 illustrates which scenario is currently supported in BPI and which part is ongoing work.

---

[2] The Apache Software Foundation: Apache Tomcat, http://tomcat.apache.org/, 2010.
[3] W3C: Scalable Vector Graphics (SVG), http://www.w3.org/TR/SVG/, 2010.
[4] Asynchronous JavaScript and XML.
[5] Oracle: JavaServer Pages Technology, http://www.oracle.com/technetwork/java/javaee/jsp/, 2010.
[6] Oracle: Java Servlet Technology, http://www.oracle.com/technetwork/java/javaee/servlet/, 2010.
[7] Oracle: JavaServer Faces, http://javaserverfaces.java.net/, 2010.
[8] The Apache Software Foundation: Apache ODE, http://ode.apache.org/, 2010.
[9] Microsoft: PowerPoint 2010, http://office.microsoft.com/en-us/powerpoint/, 2010.

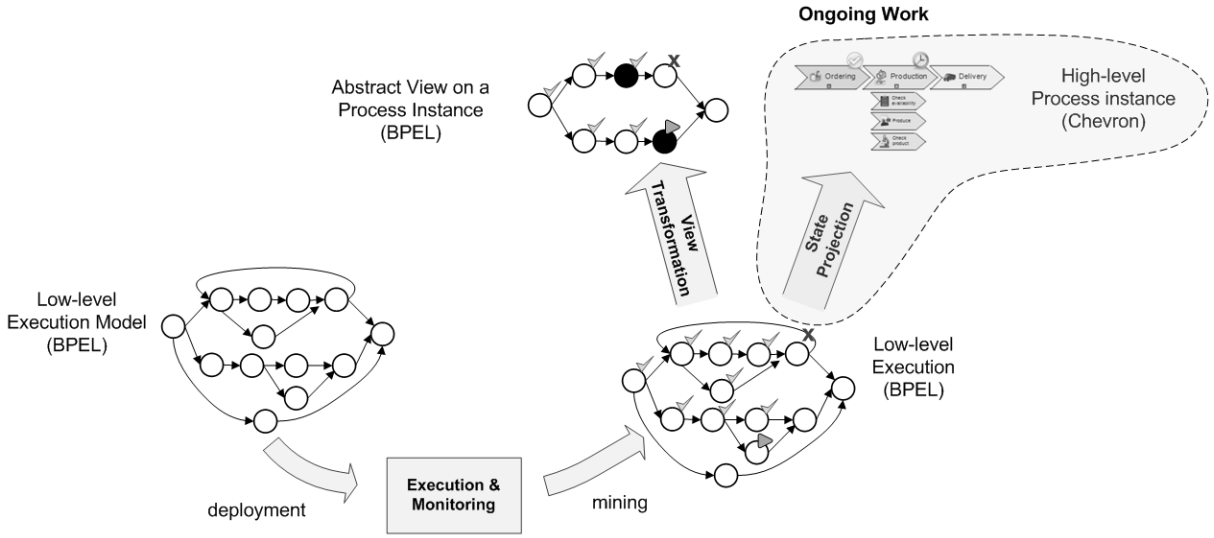

Fig. 2. Current status of the implementation with respect to supported scenarios.

2.2 Supported Language Constructs and States

BPI supports the graphical display of *all activity types* in BPEL, including `<scopes>` with `<handlers>` and `<flow>`s. BPI has been successfully evaluated with two BPEL processes containing around one hundred activities each. Arbitrary control links in `<flow>` activities can be displayed.

In [7] an algorithm for layout of processes specified in the Business Process Model and Notation (BPMN) [8] was proposed. This algorithm uses a grid in which topologically sorted activities are inserted. We were able to use this algorithm as a basis to implement the layout function for BPEL processes. However, the layout algorithm implementation for BPEL still needs to be improved in order to present very complex `<flow>` constructs in a well-arranged way.

Different execution states of activities are distinguished. Our prototype supports the activity states *Outstanding*, *Enabled*, *Started*, *Completed*, *Skipped*, *Failed* and *Recovered*. These states are displayed with different icons and opacity. In Listing 1 a simple BPEL process excerpt is shown. Figure 3 shows the corresponding diagram that is being generated for an instance of this process.

```
<process …>
   <sequence name="Sequence1">
      <receive name="Receive1" …/>
      <assign name="Assign1" …/>
      <if name="If1">
         <condition …/>
         <reply name="Reply2" …/>
         <else name="Else1">
            <sequence name="Sequence2">
               <assign name="Assign2" …/>
               <reply name="Reply1" …/>
            </sequence>
         </else>
      </if>
   </sequence>
</process>
```

Listing 1. Exemplary BPEL process.

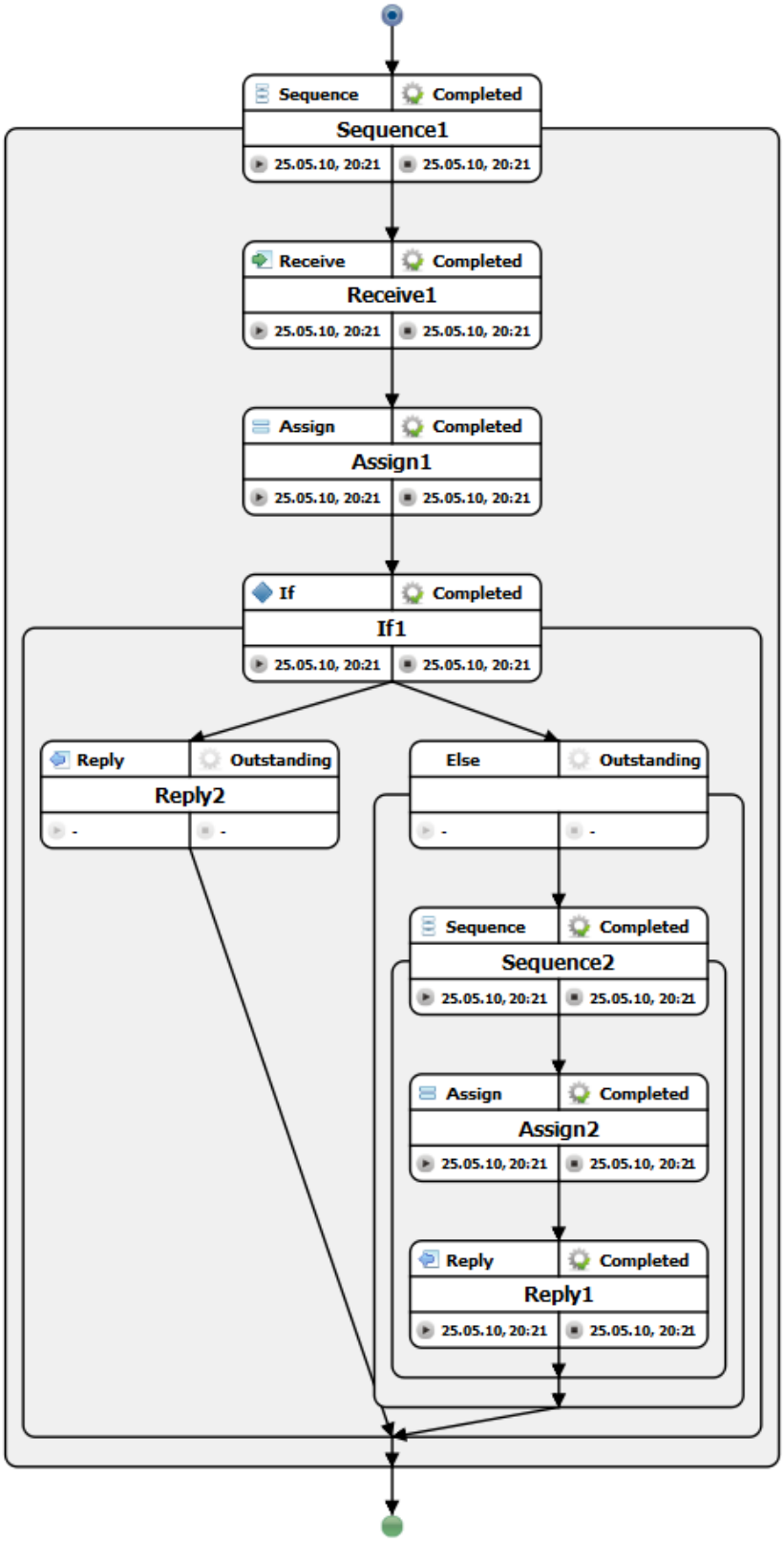

Fig. 3. Exemplary business process instance diagram.

2.3 Selection of Process Instances to be Monitored

The user frontend provides two selection tables to select the desired process instance, see Figure 4. The upper table shows all process models which are deployed to the process engine. The lower table displays the instances available for the selected process models. Both tables have sorting and filtering options and can be minimized or maximized. The user can select the number of entries to be shown in the tables at once and switch to other pages if further entries exist.

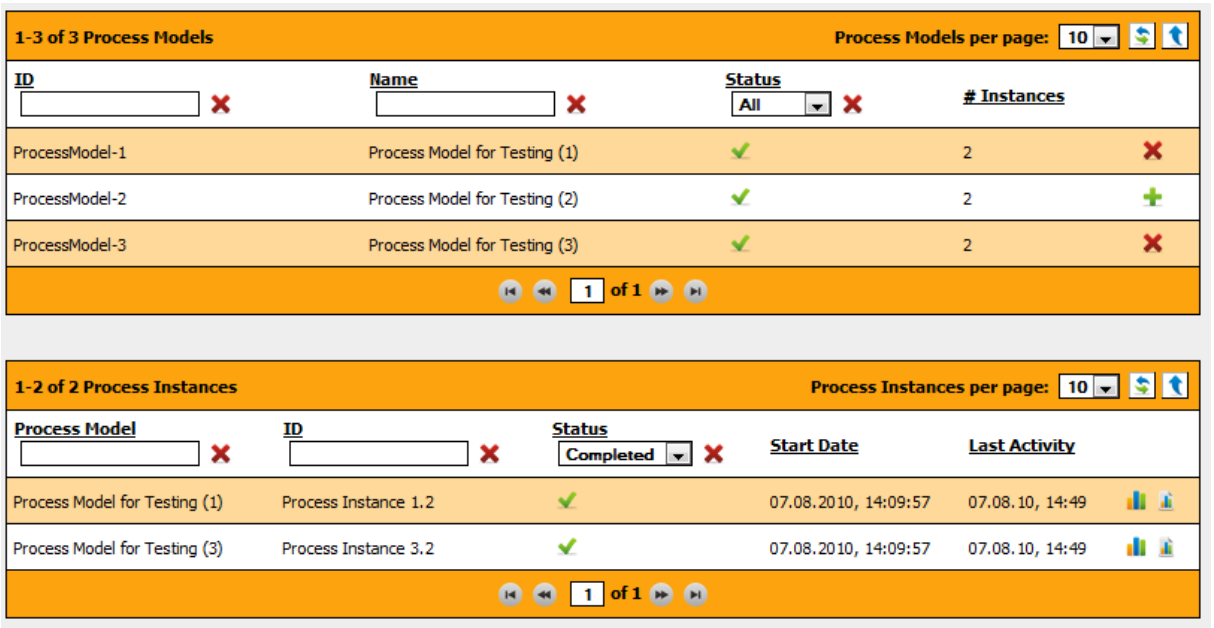

Fig. 4. Selection of process model and process instance.

2.4 Settings for Abstraction and Highlighting

The process instance diagram as well as the abstraction and highlighting settings are either displayed beneath the tables, or can be opened in a new window. An exemplary process instance diagram page is shown in Figure 5. In the illustrated process instance the left branch was executed and all activities of this branch are marked as *Completed*. The activities of the other branch are marked as *Skipped*. Generally, the current state of the process instance is displayed by the generated SVG, based on client-side refreshment requests in a predefined interval. The refreshment interval can be changed in the header of the table. Manual refreshment is also possible.

The upper part in Figure 5 shows the abstraction settings that can be used to define abstract views on a process instance. One way to define an abstract view is to manually select the activities which should be *omitted* in order to reduce the complexity of the process. Furthermore, activities that should be *highlighted* can be selected in order to emphasize particular parts. In both cases the activities can be chosen on the basis of the name or the type of an activity.

The sliders above the process instance diagram shown in Figure 5 are another way to adjust the displayed process graph. The basic idea for a model abstraction slider has been presented in [9]. This approach proposed a stepwise reduction of a process model by aggregation and omission of activities based on their relevance. However, the approach described in [9] does not discuss how to apply this concept to the specifics of BPEL as underlying process language and it does not focus on monitoring in particular. The sliders we implemented in our prototype are tailored for abstraction of BPEL process instances. Both sliders can be used independent of each other, and they can also be combined.

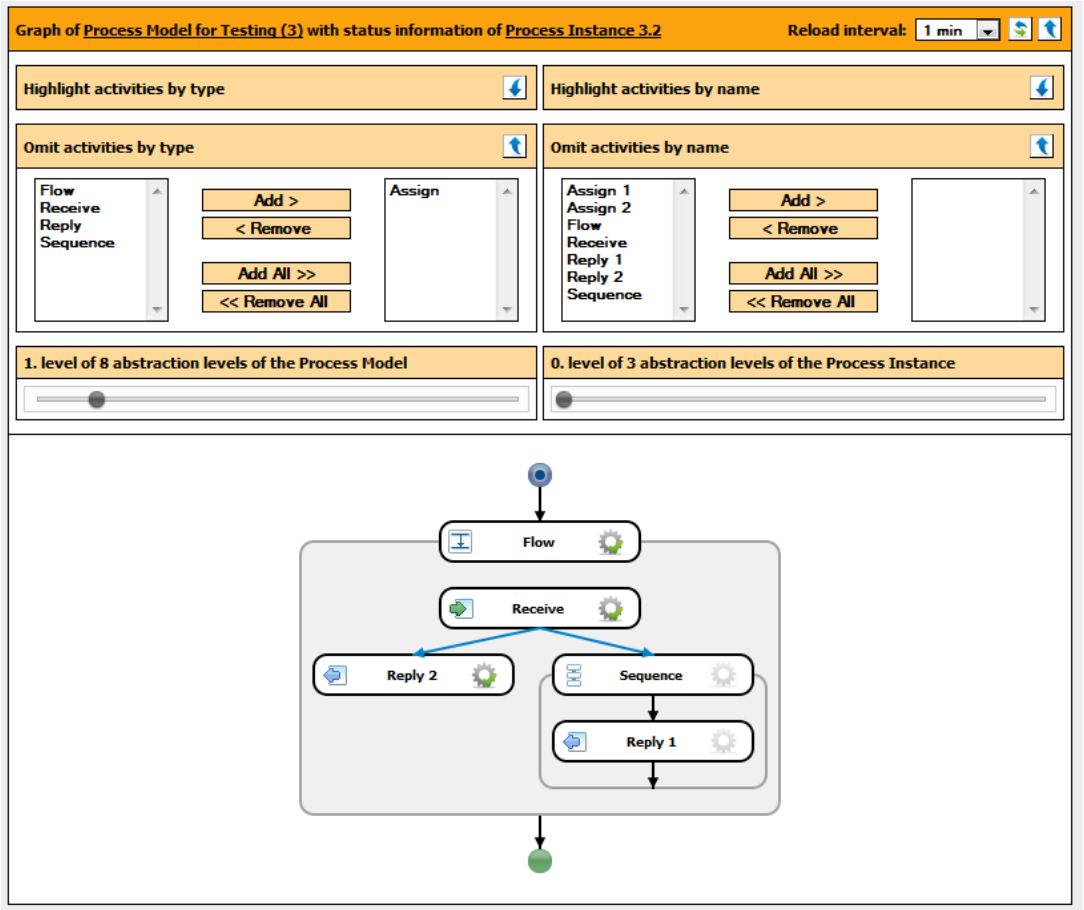

Fig. 5. Exemplary process instance diagram and abstraction settings.

The slider on the left ("abstraction levels of the Process Model") is used to change the structure of the process graph without considering runtime information. The model abstraction slider in our prototype for monitoring of BPEL processes provides at least six abstraction levels, tailored to the specifics of BPEL. On the one side, this concerns the different types of language constructs; on the other side it concerns the possibility to combine block-structured elements and graph-based elements [12]. The maximum number of levels is dynamically calculated based on the nesting depth of block structures. The following levels are provided:

- Level zero: The process graph is not adjusted.
- Level one: A compact view on the original process graph is enabled (used in Figure 5).
- Level two: `<assign>` and `<empty>` activities are omitted.
- Level three: `<throw>`, `<rethrow>`, and `<validate>` activities are omitted.
- Level four: Compensation and termination handlers are aggregated.
- Level five: `<catch>`, `<catchall>`, `<onAlarm>`, `<onEvent>` and `<onMessage>` activities are aggregated.

Starting from level six, the nesting depth of the process graph will be reduced. This means that deeply nested structures are *aggregated* and their execution states are combined.

Figure 6 shows a sample process model in three different levels of abstraction. Listing 2 shows the corresponding BPEL code of the process. In this example, only the left slider is used to adjust the process graph. The following slider values were used:

- Slider value 0: The first image (Figure 6, left) shows the original process model with the status information of a process instance. For every activity the following data is displayed: type (icon, name), activity name, status (icon, name), start and end date of the execution.
- Slider value 1: The second image (Figure 6, center) shows the process in a compact mode. Start and end dates are not displayed and names of the type and names of the status are left out.
- Slider value 6: In the third image (Figure 6, right) the compact mode is used as well. In addition, several activities are omitted. Both `<assign>` activities and the child activities of the `<sequence>` are not rendered. The links *Receive* → *Assign 2* and *Assign 2* → *Reply 2* are omitted as well, but a new link *Receive* → *Reply 2* is automatically inserted to preserve control dependency. When omitting an activity, incoming and outgoing links are rearranged.

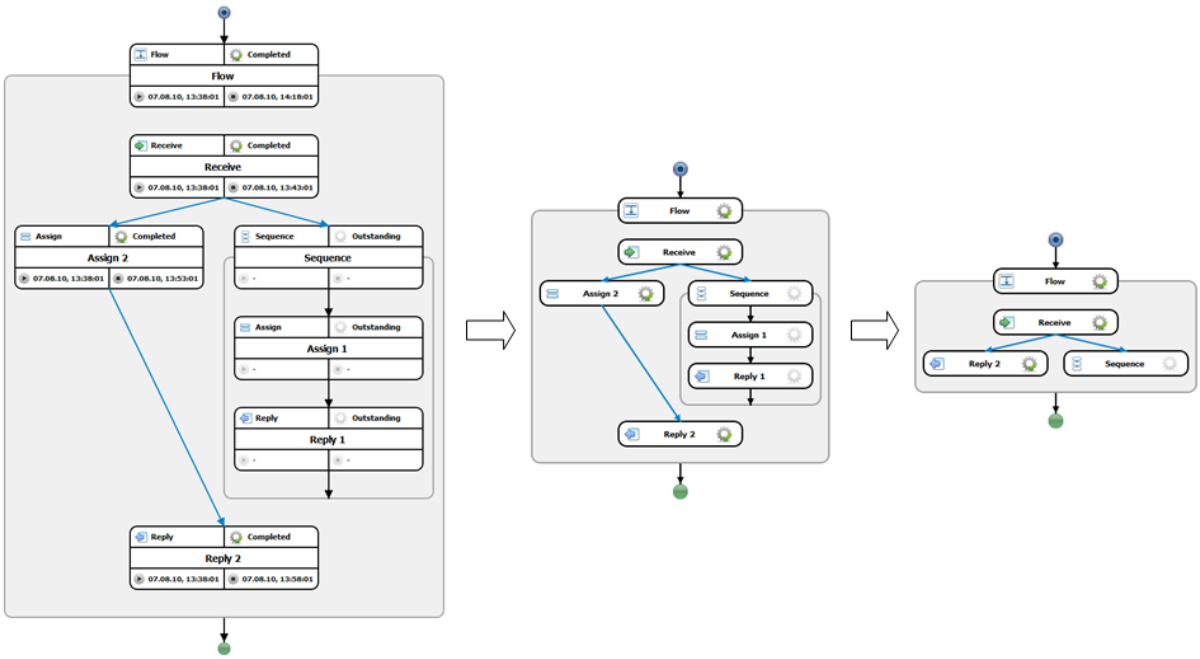

Fig. 6. Exemplary process instance diagram on different levels of abstraction.

```
<process>
   <flow name="Flow">
      <links>
         <link name="link1"/>
         <link name="link2"/>
         <link name="link4"/>
      </links>
      <receive name="Receive" …>
         <sources>
            <source linkName="link1"/>
            <source linkName="link4"/>
         </sources>
      </receive>
      <reply name="Reply 2" …>
         <targets>
            <target linkName="link2"/>
         </targets>
      </reply>
      <assign name="Assign 2" …>
         <targets>
            <target linkName="link1"/>
         </targets>
         <sources>
            <source linkName="link2"></source>
         </sources>
      </assign>
      <sequence name="Sequence">
         <validate name="Validate"/>
         <assign name="Assign 1" …/>
         <reply name="Reply 1" …/>
            <targets>
               <target linkName="link4"/>
            </targets>
      </sequence>
   </flow>
</process>
```

Listing 2. Exemplary BPEL process with a `<flow>` activity.

We have implemented another slider which focuses on functionality for business process monitoring in particular. The slider on the right (see Figure 5, “abstraction levels of the Process Instance”) considers runtime information to emphasize the status information of the selected process instance and provides runtime-specific abstraction. There exist exactly four abstraction levels.

- Level zero: The process graph is not adjusted.
- Level one: The execution path used in this process instance is highlighted in orange color.
- Level two: The highlighting is refined based on the duration of the activity; short running activities are displayed lighter than long running ones.
- Level three: Skipped activities are omitted.

In Figure 7 and Figure 8, the usage of this instance abstraction slider is exemplified in an excerpt of a process instance diagram. In Figure 7, level two has been set, combined with the compact display mode (model abstraction level one). In Figure 8, level three is set for the instance abstraction. As a consequence, dead paths are omitted.

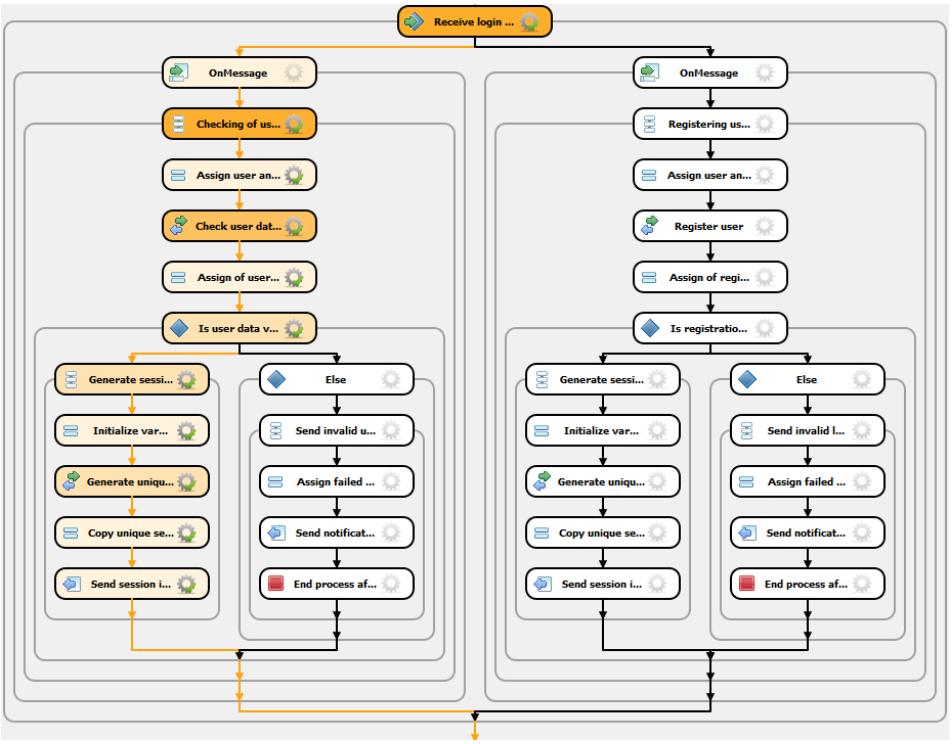

Fig. 7. Excerpt from a process instance diagram, using model abstraction level one, combined with instance abstraction level two. The execution path is highlighted and the duration of activities is shown based on the intensity of colours.

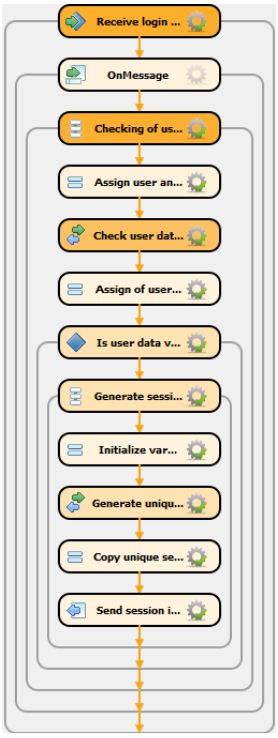

Fig. 8. The same excerpt, using model abstraction level one, combined with instance abstraction level three. The execution path is highlighted, the duration of activities is shown based on the intensity of colours, and skipped activities are omitted.

2.5 Example of a Diagram generated for an Instance of a Complex Business Process

Figure 7 shows a diagram generated for a business process which has been developed in the course of COMPAS project [11]. In this diagram, only the compact mode has been selected to demonstrate the capability of the prototype to deal with complex business processes.

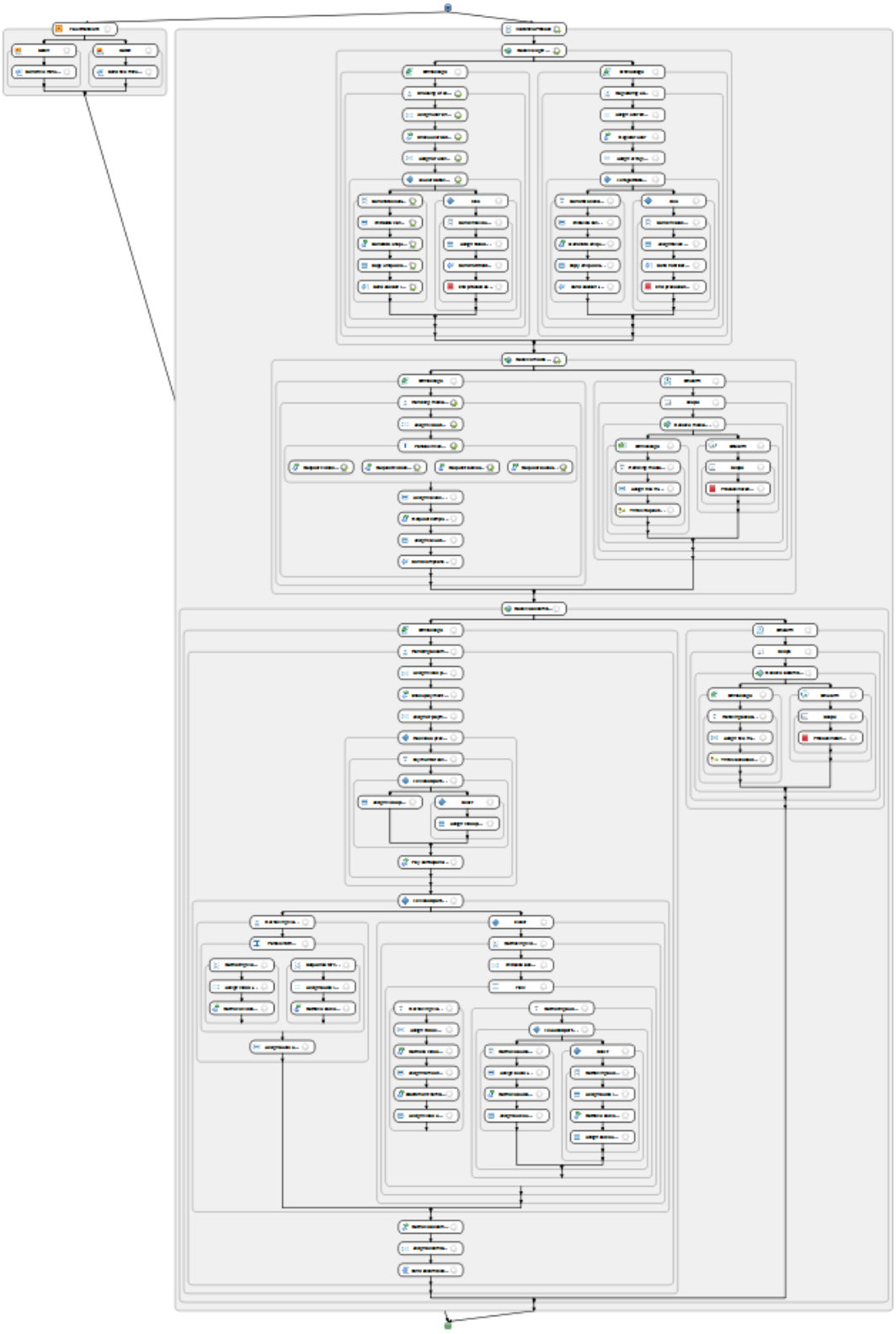

Fig. 7. Exemplary complex business processes instance diagram in compact mode.

Acknowledgements

The work published in this article was partially funded by the COMPAS project (FP7-215175), under the EU 7th Framework Programme ICT Objective. Many thanks go to Steve Strauch and Huy Tran for their efforts in the development of the complex usage scenarios. These scenarios were very helpful in the evaluation of the prototype.

References

[1] D. Schumm, F. Leymann, A. Streule. Process Viewing Patterns. Proceedings of the 14th International IEEE Enterprise Distributed Object Computing Conference (EDOC 2010), IEEE Computer Society Press, 2010.

[2] D. Schumm, F. Leymann, A. Streule. Process Views to Support Compliance Management in Business Processes. Proceedings of the 11th International Conference on Electronic Commerce and Web Technologies (EC-Web), Springer, 2010.

[3] F. Daniel, F. Casati, V. D'Andrea, S. Strauch, D. Schumm, F. Leymann, E. Mulo, U. Zdun, S. Dustdar, S. Sebahi, F. de Marchi, M. Hacid. Business Compliance Governance in Service-Oriented Architectures. Proceedings of the IEEE Twenty-Third International Conference on Advanced Information Networking and Applications (AINA'09), IEEE Press, 2009.

[4] G. Latuske. Sichten auf Geschäftsprozesse als Werkzeug zur Darstellung laufender Prozessinstanzen (in German). Diplomarbeit Nr. 3036, Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, 2010.

[5] Organization for the Advancement of Structured Information Standards (OASIS). Business Process Execution Language 2.0 (BPEL), OASIS Standard, 2007.

[6] D. Schumm, G. Latuske, F. Leymann, R. Mietzner, T. Scheibler. State Propagation for Business Process Monitoring on Different Levels of Abstraction. Proceedings of the 19th European Conference on Information Systems (ECIS 2011), 2011.

[7] I. Kitzmann, C. König, D. Lübke, L. Singer: A Simple Algorithm for Automatic Layout of BPMN Processes. Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing, IEEE, 2009.

[8] Object Management Group (OMG). Business Process Model and Notation (BPMN). OMG Available Specification, Version 2.0, January 2011.

[9] A. Polyvyanyy, S. Smirnov, M. Weske. Process Model Abstraction: A Slider Approach. Proceedings of the 12th International IEEE Enterprise Distributed Object Computing Conference (EDOC 2008), IEEE, 2008.

[10] Oracle. Java CAPS Business Process Monitor: Custom Monitoring Tools with Java CAPS, 2010, available at http://wikis.sun.com/display/JavaCAPS/Graphical+Business+Process+Monitoring, accessed 15/03/2011.

[11] COMPAS: Compliance-driven Models, Languages, and Architectures for Services, http://www.compas-ict.eu/, 2011.

[12] O. Kopp, D. Martin, D. Wutke, F. Leymann. The Difference Between Graph-Based and Block-Structured Business Process Modelling Languages. In: Enterprise Modelling and Information Systems. Vol. 4(1), Gesellschaft für Informatik, 2009.