Proviado – Personalized and Configurable Visualizations of Business Processes*

Ralph Bobrik[1], Thomas Bauer[2], and Manfred Reichert[3]

[1] Dept. Databases and Information Systems, University of Ulm, Germany
ralph.bobrik@uni-ulm.de
[2] DaimlerChrysler Research & Technology, REI/ID, Ulm, Germany
thomas.tb.bauer@daimlerchrysler.com
[3] Information Systems Group, University of Twente, The Netherlands
m.u.reichert@cs.utwente.nl

Abstract. A monitoring component is a much-needed module in order to provide an integrated view on system-spanning and cross-organizational business processes. Current monitoring tools, however, do not offer adequate process visualization support. In particular, processes are always visualized in the way they were drawn by the process designer. This static approach is by far not sufficient when dealing with more complex scenarios where different user groups usually have different perspectives on processes and related data. In such an environment different views and personalized visualizations have to be provided. In the Proviado project we are developing a framework for realizing flexible and adaptable visualizations of business processes whose data may be scattered over multiple information systems. In this paper we focus on personalization and configuration issues, and we show how process visualizations can be adapted automatically, e.g., by applying different notations for different user groups or by altering the appearance of visualized elements depending on their execution state. For this purpose we define a visualization model which maintains all required visualization parameters.

1 Introduction

In order to streamline their way of doing business, today's companies have to support a number of business processes (BP) involving different partners, departments, and actors. In this context we have seen an increasing adoption of BP management technologies as well as emerging standards for BP orchestration (e.g., BPEL4WS) and BP choreography (e.g., WS-CDL) [1]. These technologies and standards enable the definition and execution of the operational processes of an enterprise. In connection with Web Service technology, in addition, the benefits of BP automation and BP optimization from within a single enterprise can be transferred to cross-organizational business processes as well.

A BP monitoring component is a much-needed module, particularly if process data are scattered over distributed, heterogeneous information systems. It has

* This work has been funded by *DaimlerChrysler Research & Technology.*

K. Bauknecht et al. (Eds.): EC-Web 2006, LNCS 4082, pp. 61–71, 2006.

to provide comprehensive visualization support for both model and instance data. A *process model* defines the BP activities, the control and data flow between them (e.g., represented by control/data edges connecting activities among each other or linking activities with data elements), and other process aspects (e.g., resources). A *process instance*, in turn, is executed on basis of a particular process model, but comprises additional run-time information to be displayed (e.g., activity states or application data). An example is depicted in Fig. 1.

One major shortcoming of current BP monitoring software is the static way in which business processes are visualized. Usually, a process model is displayed to users in exactly the same way as designed (or painted) by the process engineer at build time. Any adaptation of model contents (e.g., hiding automated activities) may cause major efforts for re-drawing the process model, particularly if it comprises dozens or hundreds of activities, data objects, and other graphical elements. In most cases, it is even not possible to adapt the graphical appearance of a process model to user preferences at runtime. What is needed is a runtime-adaptable BP visualization, which allows to suppress skipped execution branches, to hide certain process aspects (e.g., business documents, IT systems), or to only display activities belonging to a certain role. Further, in multi-user environments, BP visualizations should be adaptable to the preferences of process participants, e.g., allowing them to apply different notations for the same process or to vary colors and symbols for the different process elements.

In the Proviado project [2] we are developing a sophisticated framework for such adaptive and configurable BP visualizations. Areas of interest include the (semantic) integration of distributed, heterogeneous process data and their defragmentation, the provision of mechanisms for creating (dynamic) process views (e.g., by applying graph aggregation/graph reduction techniques), the automated layouting of process graphs (e.g., after changes), the use of different notations for a business process, and the provision of personalized process visualizations. In this paper we focus on configuration issues, i.e., we deal with the question how to (dynamically) configure, determine, and adapt the graphical appearance of process elements. We introduce a *visualization model* (VisModel) for this purpose, which allows for the high-level and user-friendly configuration of visualization parameters for a variety of process elements. The design of this model has been non-trivial, since it must capture different kinds of visualization parameters in a user-comprehensible and maintainable way. Examples of needed parameterizations include view definitions, links to real-time process data, layout strategies,

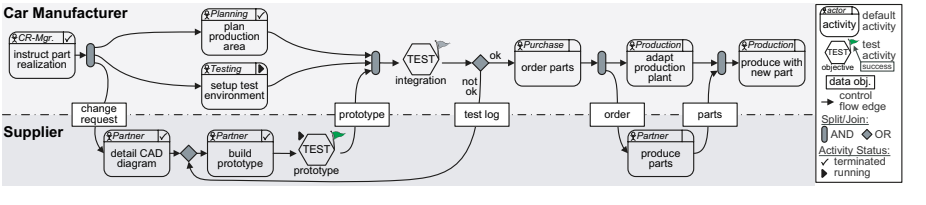

Fig. 1. Interorganizational Change Management Process (Realization phase)

preferred process notations, etc. We have elaborated the developed visualization model in several case studies in the automotive domain and implemented a powerful proof-of-concept prototype. Together with the above mentioned features the visualization model forms the key for the automated and dynamic generation of personalized process visualizations. By detaching process information from its visualization we achieve a strict separation of content and presentation, a well known approach from other areas like Web design, content management, or software development.

Section 2 starts with an example followed by the description of the requirements for realizing personalized process visualizations. Our solution approach is sketched in Section 3, and Section 4 describes our proof-of-concept prototype. In Section 5 we discuss related work. Section 6 concludes with a summary and an outlook on future research.

2 Configuration of Process Visualizations

2.1 Example

A simplified example of a cross-organizational process is depicted in Fig. 1. It shows an extract of a change management process from the automotive domain: After having decided that a certain part of a car (e.g., an electrical control unit) has to be modified, the implementation of the change is started. Related duties are then split among the car manufacturer and the part supplier. While the construction and production of the new part is in charge of the supplier, the car manufacturer must ensure that the car production facilities are adapted accordingly. For this purpose the supplier delivers a first prototype of the new part. If this prototype complies with the specifications and passes all tests ("integration tests"), it will be integrated in the production process. In order to keep track of this process a monitoring component providing site-specific visualizations of processes and related data is needed. The vision is to be able to generate adapted and personalized visualizations from given process data (see Fig. 1 and Fig. 2).

2.2 Requirements

In order to better understand the requirements for the design of our VisModel we elaborated real cases from the automotive sector. Requirements for the configuration of process visualizations are depicted in Table 1 (for details see [2]). Other requirements related to the overall visualization component (e.g., integration of process models and related runtime data, visualization of processes in different forms) can be found in [2].

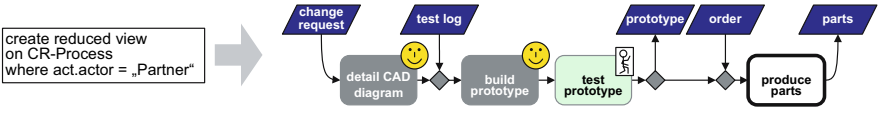

Fig. 2. Supplier's view on the process (from Fig. 1)

Table 1. Requirements for the configuration of process visualizations

Req. 1 Ability to use arbitrary symbols for visualizing a process element
Req. 2 Selection of visualization symbols based on the attribute values of the process elements
Req. 3 Precise rules for dealing with conflicting instructions regarding the visualization of process elements
Req. 4 Personalization of visualizations by adapting colors, fonts, symbols, etc.
Req. 5 Accessibility of process visualization via easy to maintain (Web) clients
Req. 6 Easy creation of new process notations and symbols
Req. 7 Easy implementation and integration with our overall architecture for BP visualization support

Req. 1 reflects the mentioned concept of separating content and presentation. In order to personalize process visualizations we must be able to easily adapt the symbol used for a certain process element. Among other things we have to specify at a high level which symbols shall be applied under which conditions. Generally, the graphical appearance of a process element depends on its properties (cf. Fig. 3). As an example consider process activities. By default we might want to represent activity nodes by a square with rounded edges. However, for activities of type *"testing"* this default representation should be replaced by a special symbol reflecting the test result with a colored flag (cf. Fig. 1). Or at the process instance level we might want to color activities depending on their state. Generally, arbitrary process data (i.e. process element attributes, instance and application data) may be used to determine the appearance of process elements. Additional complexity arises from the necessity to specify the format of a symbol dependent on process attributes (Req. 2), which are not associated with the current process element, but connected with another element via edges. Depending on the name of the actor working on an activity, for example, the color of the activity node may have to be altered, facilitating recognition of tasks belonging to the same actor. Altogether, for a complex process and its visualization this leads to a large number of dependencies or rules, describing the applicability of symbols and formats. In this scenario it is not far-fetched that two rules contradict each other. For example, let Rule 1 specify a red color for activities performed by a particular user. In contrast Rule 2 may state that running activities shall be emphasized with green color. Then it may occur that an activity fulfills both rules and it is not clear whether the activity shall be drawn red or green. This kind of conflicts between formatting instructions should be resolved in order to achieve consistent process drawings (Req. 3).

Adapted process visualizations must be made available to users as easy as possible and deployment efforts should be minimized (Req. 5). In particular, access to process information via Web browsers should be supported. In order to achieve this we use SVG (Scalable Vector Graphics) as format for displaying processes. Usually respective Web browser plug-ins are available and installed at the client side. Further, frameworks for supporting SVG on the server-side exist as well (Req. 7). For the task of generating process visualizations standard

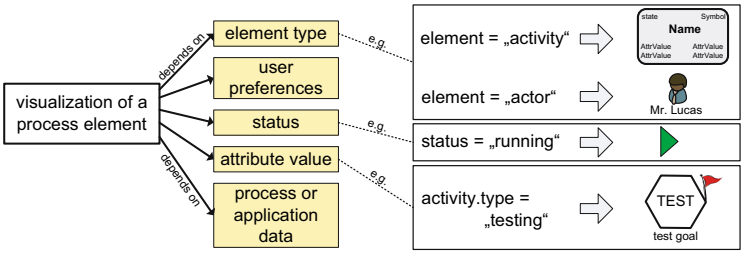

Fig. 3. Dependencies of data determining presentation

XML-based technologies (like XSLT or sXBL) allow to convert arbitrary XML data structures into SVG [3]. However, the direct application of these technologies would contradict our goal to separate content and presentation because of their complex syntax. In particular, graphical aspects are mixed with the logic for combining different templates, which results in high maintenance costs.

3 Visualization Model

Key component for specifying and maintaining visualization parameters is the Visualization Model (VisModel), whose entries are organized in an XML-based tree-structure. Fig. 4 depicts the role of this model in our overall approach for generating personalized process visualizations. The VisModel represents a logical view on the parameters for this visualization procedure. This includes, for instance, a representation of the process model to be displayed, an optional view definition reassembling the process model, a definition of the notation to be used, graphical settings regarding the appearance of process elements (e.g., colors, fonts, etc.), and information needed to access workflow or application data at runtime. In order to realize a particular process visualization we use one VisModel. Consequently, if different visualizations of a process are desired, logically, multiple VisModels have to be created. Note that the information needed for this can be gathered partially from existing information (e.g., reusing models capturing visualization profiles of a particular user group).

Fig. 4 shows the steps (S0–S3) necessary to automatically generate a process visualization. Starting point is an "integrated" process model, which correlates (fragmented) process data from different source systems in a harmonized way. First, we restrict this visualization content to that information needed by the user (S0). This is realized by a view component which applies aggregation and reduction techniques to process models (cf. Fig. 2). Step S0 is followed by formatting steps S1–S3: S1 fixes the graphical symbols designed for the different process elements. Doing so we consider information from the visualization model; S2 fills graphical symbols with real attribute values related to the process model or process instance to be displayed; within S3 formatting parameters are customized to user preferences, e.g., by coloring the process visualization in accordance to cooperate identity guidelines.

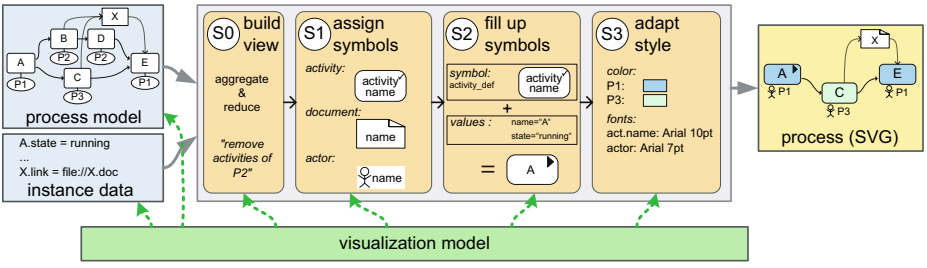

Fig. 4. Role of the Visualization Model in generating a process visualization

3.1 Template Mechanism

To enable the flexible configuration of the used process notation, we introduce a sophisticated template mechanism. Key design criteria have been Req. 1–3, with the reuse of existing templates in mind. The mechanism is subdivided in two parts: the description of the symbols and definition of their usage.

Describing Symbols. A template definition consists of three specification parts (cf. Fig. 6a): 1) input parameters of the template, where references to process elements are handed over; 2) representation of the symbol in SVG; 3) parameters (e.g. name of activity, activity state, starting time, etc.) to be filled with process data values. As mentioned, we adopted SVG as format for defining graphical symbols because of its XML-based syntax and the general advantages of vector graphics over raster graphics. This also allows for the easy definition of process symbols by using off-the-shelf SVG editors. In our approach each template defines exactly one symbol with its graphical characteristics (e.g. shapes and text areas). The text areas (i.e., parameters of the template) are filled by concrete values from the process model/instance to be displayed. Within the parameter sections, XPath expressions (relative to the SVG-symbol root) are used to describe the location of the corresponding text area (`location` attribute). As process data values may need to be transformed before presenting them to the user (e.g. converting an internal date format into a standard format; cf. Fig. 6b) the `value` attribute may comprise of code in a scripting language (e.g. JavaScript). Via these scriptlets it is further possible to access all kind of process data and to arrange it using arbitrary scripting functions (cf. parameter `endtime` in Fig. 6 a). For expressing special formatting options dependent on arbitrary process attributes, if-then-else or choice structures may be used. Hereby the evaluation conditions are expressed by also using JavaScript.

Fig. 5 shows an example for an activity template. The right side depicts the symbol definition based on SVG. On the left, corresponding parameter definitions are shown. Among other things they illustrate the mechanisms used to reference the locations of the data values inside the symbol. A choice construct is used to determine the correct process state symbol for activity nodes.

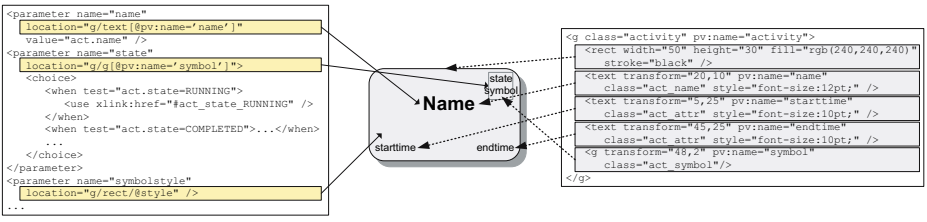

Fig. 5. Template mechanism: Definiton of a symbol

```
<template id="default_act">
  <!- input section ->
  <inputs>
    <input variable="act" type="activity">
      <descr>activity node</descr>
    </input>
  </inputs>
  <graphic>
    <!- symbol section (SVG) ->
    <symbol>
      <g class="activity" pv:name="activity">
        ...
      </g>
    </symbol>
    <!- parameter section ->
    <parameter name="name"
      location="g/text[@pv:name='name']"
      value="act.name" />
    ...
    <parameter name="endtime"
      location="g/text[@pv:name='endtime']"
      value="formatDate(act.end,'dd/mm/yyyy')"/>
  </graphic>
</template>
```

```
<if test="self.type=ACTOR">
  <template id="actor">
    <inputs>
      <input name="actor" value="self"/>
    </inputs>
  </template>
</if> <if test="self.type=ACTIVITY">
  <choose>
    <when test="self.type='testing'">
      <template ref="testing_act">
        <inputs>
          <input name="act" value="self"/>
        </inputs>
      </template>
    </when>
    <otherwise>
      <template ref="default_act">
        <inputs>
          <input name="act" value="self"/>
        </inputs>
      </template>
    </otherwise>
  </choose>
</if>
```

Fig. 6. (a) Definition of templates (b) Usage of templates

Defining Usage. Having defined a set of templates the next challenge is to specify under which conditions these templates shall be applied. Main complexity in this context is to define the correlations between process elements and available templates (cf. Fig 3). As an example consider the process from Fig. 1 where we want to use a rectangle with rounded edges for displaying activities and a special symbol for representing "test activities". We first considered using logic rules for this task, but withdrew this idea. First, we would have obtained a large number of rules to be maintained. Second, specification of such rules would have been a complex task (e.g., precise conditions for firing rules would have had to be specified). Third, rules are not guaranteed to be conflict-free; i.e., there might be two rules, one defining a red background color and the other specifying a green color. A general conflict resolution strategy for this case would be very complex to realize. The solution we have chosen instead is depicted in Fig. 6b. Using "if-then-else"-like statements together with a first-occurrence-wins policy, it is ensured that the template to be applied can be determined unambiguously at runtime. The algorithm we developed in this context traverses all elements of the process model and assigns the corresponding symbols to them.

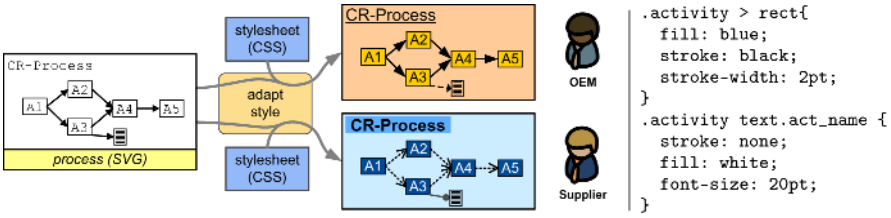

Fig. 7. Use of stylesheets for adapting format parameters

3.2 Formatting a Process Visualization

The task of formatting a process visualization is subdivided into 3 steps (cf. Fig. 4). In Section 3.1 we already explained how to describe unambiguously which symbol has to be used for visualizing a certain process element. This definition is interpreted in Step S1 where symbol templates are assigned to the process elements. In Step S2 the parameter values of the templates are calculated according to the scriptlets contained in the templates. Consequently, the placeholders are substituted by concrete values from the process model (instance). During Step S3 graphical attributes of the process elements are adapted according to users' preferences (cf. Fig 7). In this step, colors, fonts, line styles, etc. can be modified using Cascading Style Sheets (CSS). The latter complement SVG graphics' formatting capabilities. In contrast to the previous steps, S3 is executed by the rendering engine of the SVG viewer. By using templates for the coarse layout and stylesheets for the personalization of graphical attributes we gain additional flexibility, which allows us to easily adapt the final appearance of process visualizations. This is realized by providing more than one stylesheet for a particular VisModel. The stylesheet to be used is selected lately at runtime, e.g., depending on the organizational unit that requested the process visualization. It is even possible to hide process elements in this late stage using CSS-attributes.

Key to deal with the requirements from Table 1 is the described template mechanism. It enables great flexibility defining the appearance of process elements and also promotes their reuse (Req. 1–3). The look of the resulting process graphs can be customized further using stylesheets (Req. 4). By adopting SVG easy deployment of a visualization component considering available Web-browser plug-ins becomes possible (Req. 5). Further, the availability of frameworks for generating SVG server-side is useful in our context. Thus Req. 5 and 7 are met. In addition to this, SVG allows for the easy definition of process symbols using standard editors (Req. 6). The implementation efforts could be reduced by harking back to existing libraries (e.g. for JavaScript and XPath) (Req. 7).

4 Proof-of-Concept Implementation

We have implemented the described concepts in a powerful proof-of-concept prototype. Fig. 8 depicts sample screens showing the same process in two different execution states and with different appearance. Fig. 8a shows a visualization of

the process from Fig. 1, which is similar to what is offered by current process design tools. In Fig. 8b the same process is depicted in another style and with progressed execution state. Reducing the number of elements and streamlining notation often leads to an improved readability as in the given case. This is particularly suitable for monitoring components highlighting activity execution states using different colors. Similarly, another VisModel using colors for identifying participating actors and varying border styles for representing execution states can be defined. Additional information on less important process attributes can be visualized using tool-tips (cf. Fig. 8b). Since we apply SVG, processes can be monitored with standard Web browsers. Usually they include respective plugins providing basic operations (e.g., zooming) by default. Advanced SVG features like animation and scripting have enabled us to interact with users in a sophisticated way, e.g., by replaying process execution with animated state transitions. Our demonstrator is implemented using Java and other standard techniques like XML, XPath, CSS, and JavaScript.

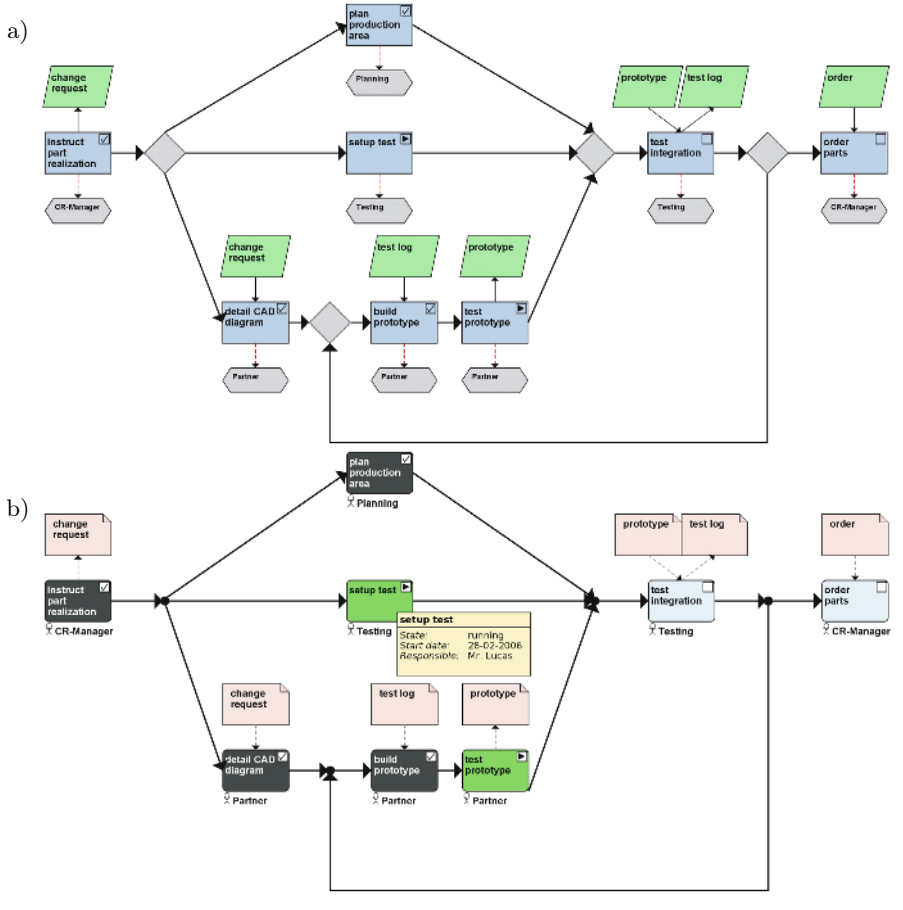

Fig. 8. Screenshots showing two different presentations of the same process

5 Related Work

There are numerous Workflow Management systems (WfMS) which enable the definition, execution, and monitoring of business processes. In particular, the modeling and monitoring components of these WfMS provide visualization support for model and instance data. However, only those process data can be visualized which are under control of the WfMS (e.g., WBI Monitor [4]). A more open approach is followed by process performance management (PPM) tools (e.g. ARIS PPM [5]), which support the monitoring of processes whose data is scattered over multiple information systems. Altogether these tools show limitations with respect to their visualization component. Neither can the visualization of a process be personalized nor can it be adapted to the current context. In particular, processes are always displayed as drawn by the process designer, and the discussed requirements are not met. Furthermore, at the process instance level visualization support is mainly restricted to the control flow perspective, whereas other process aspects (e.g., data flow, resources, application data, etc.) cannot be displayed. Finally, the Web interfaces of current tools are rather poor and also not adaptable to users' needs. For these reasons, current monitoring tools are mainly used by administrators and developers. A more flexible, but also more complex approach is offered by generic visualization software; e.g., ILOG JViews is not bound to a specific WfMS or workflow meta model and therefore enables more flexible, but also more complex to realize process visualizations.

The area of Information Visualization deals with the use of computer-supported, interactive, visual representations of data to amplify cognition [6]. Several approaches are available for the visualization of general graph structures [7]. However, there is literature dealing with BP visualization [8,9,10,11]. Most approaches focus on special aspects rather than providing a complete picture. Examples include the layouting of certain process graphs [9], the mapping to SVG [10,12], and the adaptation of the set of displayed process elements [13]. ArchiMate [11] is more ambitious and supports different visualizations and viewpoints of enterprise architectures for different user groups. However, most of the discussed requirements have not been addressed by these approaches.

6 Summary and Outlook

We have presented an approach for the personalized visualization of process model and process instance data. For this we have introduced a VisModel that comprises all configuration parameters providing adaptable BP visualizations. Key concept is the flexible definition of process symbols independent from process model data. This has been realized based on a powerful template mechanism. Due to lack of space we have explained basic concepts by means of simple examples rather than providing formal considerations.

The configuration of process visualizations, i.e., the specification of a VisModel, is a complex task that requires writing XML-code. We plan to build a sophisticated tool that allows for the graphical definition of a VisModel as well

as for template reuse. Template generation will be facilitated integrating an SVG editor. Layouting general process graphs is another complex task [14]. This and other challenges have been factored out in this paper. At the moment we opt for using existing positioning information of process elements, but we aim at replacing this workaround with sophisticated layout algorithms. Layouting will be introduced to the formatting task from Fig. 4 (after Step S2), where the resulting graph objects can be used to calculate the adequate layout. Automatic layout even gets more important when taking into account more advanced issues like view mechanisms or different visualization forms of process data (e.g., swim lanes, gantt charts, etc.). Finally, in Proviado several other activities are on their way to accomplish all tasks depicted in Fig. 4. This includes view generation (S0), access control, and process data integration.

References

1. Havey, M.: Essential Business Process Modeling. O'Reilly Media (2005)
2. Bobrik, R., Reichert, M., Bauer, T.: Requirements for the Visualization of System-Spanning Business Processes. In: Proc. 16th Int. Workshop on Database and Expert Systems Applications (DEXA), Copenhagen, Denmark (2005) 948–954
3. Jolif, C.: Comparison between XML to SVG Transformation Mechanisms. In: Proc. SVG Open'05, Enschede, Netherlands (2005)
4. IBM: IBM WBI Monitor V. 4.2.3. (2003) IBM Report.
5. IDS Scheer AG: ARIS Process Performance Manager (PPM). White Paper (2004)
6. Card, S.K., MacKinlay, J.D., Shneiderman, B.: Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann, New York (1999)
7. Herman, I., Melançon, G., Marshall, M.S.: Graph Visualization and Navigation in Information Visualization: A Survey. IEEE Transactions on Visualization and Computer Graphics **6** (2000) 24–43
8. Luttighuis, P.O., Lankhorst, M., van de Wetering, R., Bal, R., van den Berg, H.: Visualising Business Processes. Computer Languages **27** (2001)
9. Six, J.M., Tollis, I.G.: Automated Visualization of Process Diagrams. In: Proc. 9th Int. Symp. on Graph Drawing (GD '01), Vienna, Austria (2002)
10. Koolwaaij, J.W., Fennema, P., van Leeuwen, D.: SVG for Process Visualization. In: SVG Open 2003, Vancouver (2003)
11. Steen, M., Akehurst, D., ter Doest, H., Lankhorst, M.: Supporting Viewpoint-Oriented Enterprise Architecture. In: 8th Int. Enterprise Dist. Object Computing Conf. (EDOC), Monterey, California (2004) 201–211
12. Mendling, J., Brabenetz, A., Neumann, G.: EPML2SVG - Generating Websites from EPML Processes. In: Proc. of the 3rd GI Workshop on Event-Driven Process Chains (EPK 2004), Luxembourg (2004)
13. Streit, A., Pham, B., Brown, R.: Visualization support for managing large business process specifications. In: Proc. 3rd Int. Conf. Business Process Management (BPM). Volume 3649 of LNCS., Nancy, France (2005) 205–219
14. Rinderle, S., Bobrik, R., Reichert, M., Bauer, T.: Business process visualization - use cases, challenges, solutions. In: 8th International Conference on Enterprise Information Systems (ICEIS'06), Paphos, Cyprus (2006)