Improving Business Process Visualizations

Philip Effinger, Martin Siebenhaller, Michael Kaufmann

WSI-2009-02

ISSN 0946-3852

Arbeitsbereich Paralleles Rechnen
Prof. Dr. Michael Kaufmann
Wilhelm-Schickard-Institut für Informatik
Fakultät für Informations- und Kognitionswissenschaften
Eberhard-Karls-Universität
Sand 13, 72076 Tübingen, Germany
Email: {effinger,siebenha,mk}@informatik.uni-tuebingen.de

Abstract

Business processes are at the core of today's business world. They state precisely how given challenges are to be handled. Most of the effort put into business processes is either the task of designing a new process or the task of improving an existing process. In both cases, visualizations of the process supports the user in achieving his objectives. However, even visualizations of complex structures are in danger of becoming complex themselves. Thus, readability often is the victim of large visualizations.
In this report, we propose an approach for increasing the readability of process visualizations. The approach is based on a graph-geometric algorithm that performs constrained cuts on given visualizations. A side constraint of the approach is not to hide the inherent complexity of the business process. As modeling notation for business processes, the business process modeling notation *BPMN* is state of the art and it is primarily supported by our approach. Besides increasing readability of existing visualizations, we will also show how to find an initial visualization.

Keywords: Business processes, BPMN, Business Process Visualization, Graph Drawing

1 Introduction

In 2007, request for proposal (RFP) of version 2.0 of the business process modeling notation (*BPMN*) was opened by the object management group (OMG). The working title became "business process model and notation". This stands for the growing importance by *BPMN* in not only providing a comprehensive notation repository. *BPMN* 2.0 will also be capable of modeling meta models. However, there are only minor syntactical changes from *BPMN* 1.1 to 2.0.
Version 1.1 is used by more than 54 companies according to bpmn.org. While writing this report, BPMN version 1.2 was published in January 2009. However, changes from 1.1. to 1.2 merely include minor bug fixes or improved formatting.
Version 1.1 is used both in simulation tools and as a teaching language for beginners in the field of business processes. Many of these tools provide interfaces for the design of diagrams with BPMN elements. Unfortunately, they often lack the possibility of using computer-aided or automatic layouts. If the tool provides layout support, the results are often unsatisfactory since the algorithm used in place are not sufficient.

Our goal is to show an approach that both finds a new visualization for a given diagram and improves existing visualizations modeled with BPMN.

In this paper, we assume that the reader is familiar with the concept of graphs, dual graphs and planarity (see [5] for an overview).
In section 2, we give a short introduction into *BPMN* version 1.1. The techniques used for calculating an automatic layout are shown in section 3. In the following section 4, the approach for improving business process visualizations is presented.
The report closes with a demonstration and evaluation (section 5) as well as a conclusion (section 6).

2 BPMN

In order to be able to model a complete business process, we chose to support *BPMN* (business process modeling notation) [7]. *BPMN*[1] is a modeling notation that is propagated by the *OMG* (Object Management Group[2]). It offers a way to create graphical models that contain all information necessary for an optional subsequent implementation of the process model.
Currently, *BPMN* is undergoing a standardization process for version 2.0. Standardization will not be passed before the year 2010. In this report, we used version 1.1 which is still the most widespread version. *BPMN* consists of different categories of elements:

1. *Flow objects* control the flow in a process. They can be considered as nodes in a graph. The largest group among flow objects are *Events*, denoted as shown in Figure 2(a). Events can trigger actions during a process or initiate/terminate a (sub-)process. Therefore there are start events, intermediate events and end events.
   A task that has to be performed in a process is represented by an *Activity*, see Figure 2(b). Activities can be repeated once or several times (*loop activity*). In order to take decisions regarding the actual flow status, *Gateways* are introduced, see Figure 2(c). Gateways can join/fork flows depending on logical conditions.

2. *Connecting objects* are used to connect flow objects. Thus, they correspond to edges in a graph. There are three different connecting objects, see Figure 2(d):

   - A *Sequence flow* object is the most common object to determine the process flow. It is used to define the direction of the flow and thus the process' sequence of flow objects.
   - A *Message flow* denotes the exchange of messages during a process' execution. The message exchange can be directed inward into the process flow or outward, e.g. for a status information or error message.
   - An *Association* represents a weak relationship among flow objects. It has no impact on the process flow.

3. *Swimlanes* are used to define a higher order among flow objects. They can be interpreted as a partitioning of flow objects into logical units, e.g. departments of a company. They are represented by vertical or horizontal stripes as depicted in Figure 2(e). So-called pools are hierarchically one step above swimlanes, and are used to group related swimlanes or to model external participants in a process.

4. *Artifacts* are a group of BPMN-elements that cannot be sorted into the categories already mentioned.

   - *Annotations* offer the possibility to add comments to flow objects as well as connecting objects, see Figure 2(f). Annotations are attached to the

[1] For most recent developments, see http://www.bpmn.org
[2] see also: http://www.omg.org

corresponding object by an association. In a graph model annotations can be handled like common nodes.

- *Data objects* represent data that is available to the process or produced by executing the process flow.
- A *Group* represents a symbolic composition of BPMN-elements. It has no semantic meaning to the process flow, but it offers a way to document or analyze process properties. In the area of graph drawing those groups are known as *clusters*.

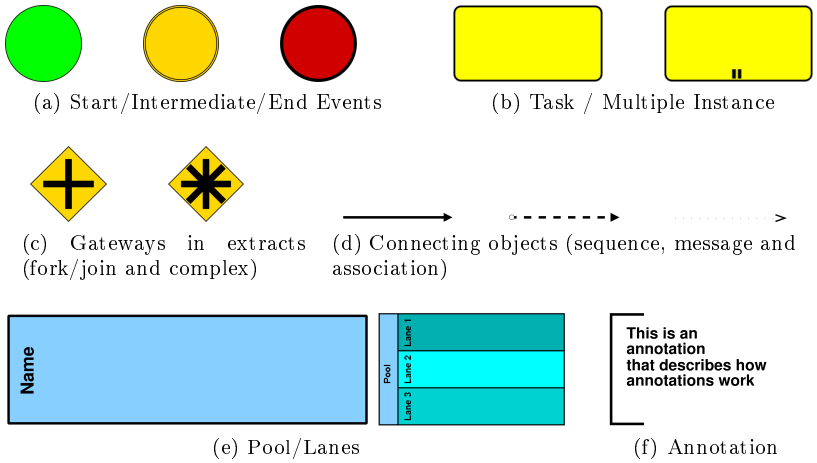


Figure 1: Overview of BPMN-elements, see also [7].

In this report, we consider graphs that consist of BPMN-elements:

Definition 1 (BPMN-graph) *A BPMN-graph is a graph $G = (V, E)$ with the following additional information:*

- *A mapping node_type : $V \to T$, where T denotes the set of types of a BPMN-element. Each node $n \in V$ is mapped to exactly one type $t \in T$.*
- *A mapping edge_type : $E \to C$, where C denotes the set of connecting objects. Each edge $e \in E$ is assigned exactly one connecting object $c \in C$.*

3 Layout algorithm

An automatic layout approach for BPMN-graphs has to support the specific requirements given by the BPMN notation, e.g. it has to support groups and swimlanes. BPMN-graphs are usually drawn using orthogonal edge routes, i.e. each edge is drawn as a sequence of horizontal and vertical line segments. Hence, we use an orthogonal layout approach for calculating the initial layout of a given BPMN-graph.

More precisely, our approach employs the implementation described in [4] that incorporates different constraints needed for the automatic layout of activity diagrams which are related to business process diagrams. The supported constraints include partitions (a generalization of swimlanes), clusters (groups) as well as a common flow direction of edges which is especially important for such diagrams. Thus, all requirements demanded by BPMN-graphs are already included.

4 Divisions

In cases where process models become very complex and cannot easily be overlooked, it is desirable to divide the resulting diagram into smaller pieces. In the following, we give an algorithm that divides BPMN-graphs subject to constraints, e.g. the size of sheets the BPMN-graph is to be printed on or the number of parts to be computed.

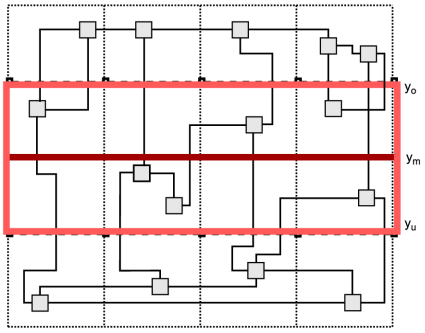
Definition 2 (Division of BPMN-Graphs) *A Division of a BPMN-Graph $G = (V, E)$ with given constraints C partitions G into node sets $V_1, \ldots, V_k$ with $k \geq 2$ such that $V_i \cap V_j = \emptyset \ \forall i \neq j$. The subgraphs induced by V_i on G have to satisfy C. Edges of $A = \{(v, w) \in E \mid v \in V_i, w \in V_j \text{ and } i \neq j\}$ are called division edges.*

A division does not necessarily correspond to a min-cut, i.e. the cuts induced by a division mainly depend on the given constraints C.
In BPMN-graphs, as for *ratio-cuts* in e.g. [6], the aim of a division is to minimiza A and to obtain subgraphs of nearly equivalent size in terms of nodes or area. Thus, we have to find suitable routes for the cuts.

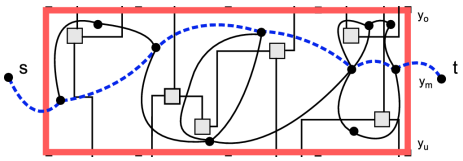
4.1 Description of the division approach

In order to find appropriate routes, we introduce the term *center band*. The *center band* is a rectangular space in the graph through which a cut runs, see Figure 2(a).

We now describe how to perform a horizontal cut, a vertical cut is performed analogously. For a horizontal cut the width of the center band is set to the width of the bounding box surrounding the graph. Height of the center band is set to a predefined value $y_o - y_u$ around the midpoint center y_m of the *center band*, see Figure 2(a). The predefined value may be given by the user or can be preset with a fraction of the graph's height. In our evaluations, we found that a value that corresponds to 10% of the graph's height is a good choice for most graphs. The *center band* should have an area size that contains at least two possible routes. The placement of the *center band* depends on how often the graph is to be divided. With two desired parts of the graph, the vertical center of the graph is chosen. With more than two desired parts, the *center bands* are inserted in geometrically equivalent distances from each other.
The core of the division algorithm is a dual graph routing inside the *center band*. The dual graph G'_D is constructed over the *cut graph* $G' = (V', E') \subseteq G$ induced by the *center band*. The embedding of G' which is needed for the construction of G'_D can be computed by means of a sweep line algorithm. Additionally we introduce two vertices s and t. Vertex s is connected to the vertex of G'_D which represents the inner face



(a) Placing a horizontal center band on the underlying graph. The red box denotes the center band. Swimlanes are depicted by dotted lines.



(b) Determining a cut using the dual graph. The dashed blue edges denote a shortest path from s to t and thus induce a cut with a minimum number of split edges in the center band.

Figure 2: Determining a cut in a graph.

containing the left border edge of the center band and vertex t to the vertex which represents the inner face containing the right border edge of the center band, see Figure 2(b). Hence, a shortest path from s to t in G'_D induces a cut inside the center band with the lowest number of cut edges (cut edges correspond to division edges).
The edges of the dual graph are weighted as follows: edges leaving the center band are set to infinite weight since we want to obtain a route inside of the center band. Alternatively, we could remove those edges. The two edges incident to s and t are set to weight 0 and the remaining edges of G'_D to weight 1. Since there is a one-to-one relation between edges of E' and edges of E'_D, a possible extension is to set higher weights for specific edges of E' that should not become division edges, if possible. Those specific weights can be set by the user. The weights are then passed to the corresponding edges of E'_D.
An example of a horizontal cut can be found in Figure 2(b). Since the weights of the edges have to be taken into account when the shortest path is calculated we use Dijkstra's algorithm [2].
Analogously to horizonal cuts, a vertical cut is performed by using a vertical *center band*, see extended example in section 5.

Performing a shortest path computation on the dual graph, we obtain the division edges for the original graph. Those edges have to be removed in order to split the graph. However, an edge removal causes information loss. Thus, we insert two replacement nodes for each such edge. Replacement nodes are a well-known construct

in the *UML* and they are also known as *connectors*. The first replacement node is connected to the source of a division edge and the second to the target. The replacement nodes are placeholders for the former edge and are marked with the name of the node they point to, e.g. the replacement node at the source is marked with the target's name and vice versa. Examples for the insertion of replacement nodes as placeholders can be inspected in section 5.

After performing a cut, the computed subgraphs are relayouted using the sketch-driven approach described in [1]. It reduces the area of the drawing without changing the user's mental map, i.e. it preserves the given embedding and shape of the edges in the subgraph. The implementation of the sketch-driven approach for *BPMN* can be inspected in existing works, e.g. in [3].

4.2 The algorithm in detail

For a given graph we first determine if a vertical and/or horizontal cut is needed to satisfy the given constraints, e.g. the prescribed size of the subgraph. If both is required we apply the cut (horizontal or vertical) which splits the lowest number of edges.

Algorithm 1: BPMN-Divisions with constraints

Data: BPMN-Graph G, constraints C
Output: Array of Graphs $[G_{sub}]$

Step 1: **if** *(C is NOT fulfilled)* **then**
  GOTO Step 2;
  else GOTO Step 3;
Step 2:
- derive center band by calculating barycentric x- and y-position(s) and set it to the predefined size
- calculate cut graph G' by means of a sweep-line algorithm
- calculate dual graph G'_D
- perform shortest path computation
- create the resulting subgraphs
- perform sketch-driven layout on each subgraph
- insert replacement nodes for divison edges
- $[G_{sub}]$.add{*subgraphs*};

GOTO Step 1 with *subgraphs* and constraints C as input;
Step 3:
return $[G_{sub}]$;

To perform a cut with prescribed number of parts, multiple non-overlapping *center bands* are introduced to search for cutting edges. Therefore, as mentioned in 4.1, the *center bands* are inserted in geometrically equivalent distances from each other.
If we want to split a drawing of a graph into sheets of fixed size, the algorithm iteratively cuts subgraphs until they fit onto a sheet. The runtime of a cut is dominated by

calculating the sketch-driven layout and thus by solving a min-cost flow problem on the planarized graph. A description of the algorithm can be inspected in Algorithm 1.

5 Evaluation

In this section we demonstrate the benefits of our visualization and the improvements on two real-world examples.

In the first case, we applied our software to the input graph shown in Fig. 3 which was drawn with the help of the tool developed in [3]. The graph considers the workflow in left-to-right perspective. The core layout produced the drawing given in Figure 4(b). The workflow is preserved and it is now drawn in the top-to-bottom perspective. All edges have orthogonal routes.
Assuming the process in Figure 4(b) is to be split, the division could be done by, for example, performing a division onto two sheets. In Figure 4(c), the *center band* is visualized by a red rectangle. Edges that are contained in the cut graph are marked (in yellow color). The corresponding cut graph is shown in Figure 4(a). In Figure 4(d) one can inspect the two resulting subgraphs. The replacement nodes are inserted and appear as blue smaller nodes. Blue nodes have no semantic in *BPMN* such that there are no interferences with the *BPMN* standard.

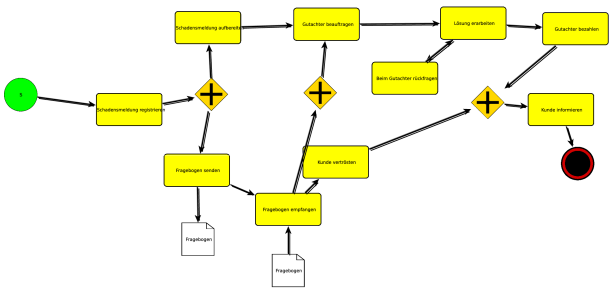


Figure 3: A sketch of a BPMN-graph showing a process model of notification of claim.

Figure 5(a) is an extended example of Figure 4(b). We added the role of the insurant (in a new swimlane). The former example described only the role of the insurer. The sequence is equivalent to the former example. First, we derive a layout (see Figure 5(b)) from the sketch. In this example, a vertical division (see Figure 5(c)) is performed since the roles (insurer/insurant) of the process represent an intuitive separation.
Finally, the divided subgraphs are shown in Figure 5(d).

Note, that although all figures use the same scale, the sum of the area of the resulting subgraphs is significantly smaller than that of the input graph. This positive effect is attributed to the application of the sketch-driven layout algorithm after cutting the graph.

6 Conclusion

In this report, we showed how to enhance a basic orthogonal layout approach in order to visualize business processes modeled with BPMN. Therefore, we applied the concept of divisions.

As demonstrated in the evaluation section, the approach offers an effective way to improve a given orthogonal layout by reducing its complexity and increasing interactivity with the user in the, in general, static field of layout computation. In combination with sketch-driven layout, see [1], interactivity can even reach a level for high usability that enables a user to model and layout in one comprehensive and efficient tool.

Future research might comprise the support of higher user interactivity for drawing business processes as well as more sophisticated concepts for improving visualizations and simultaneously reducing complexity of visualizations.
Exploiting the modeling language's semantics, layouts can be computed tighter to the needs of a specific modeling language as BPMN.

References

[1] U. Brandes, M. Kaufmann, D. Wagner, and M. Eiglsperger. Sketch-driven orthogonal graph drawing. *Lecture Notes In Computer Science*, 2528, 2002.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.

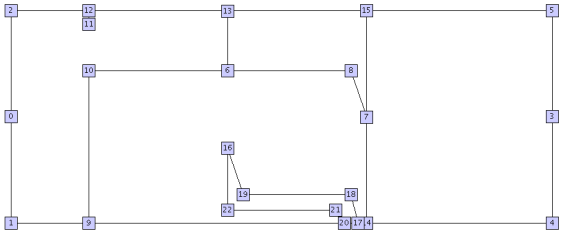
[3] P. Effinger. Automatisches layout von geschäftsprozessen. Diplomarbeit, Eberhard-Karls-Universität Tübingen, Wilhelm-Schickard-Insitute, Sand 13, May 2008.

[4] M. Siebenhaller and M. Kaufmann. Drawing activity diagrams. Technical report, Wilhelm-Schickard-Institut, 2006.

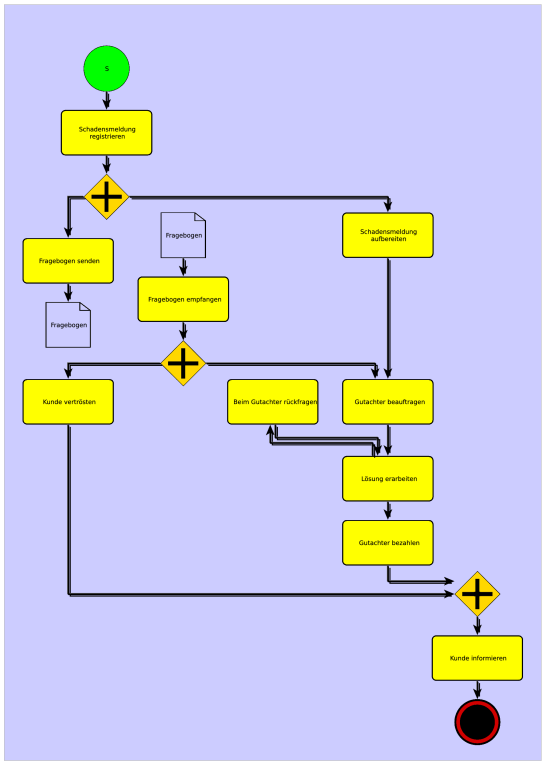
[5] R. Tamassia, G. DiBattista, P. Eades, and I. Tollis. *Graph Drawing*. Prentice Hall, 1999.

[6] S. Wang and J. Siskind. Image segmentation with ratio cut. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(6):675–690, 2003.

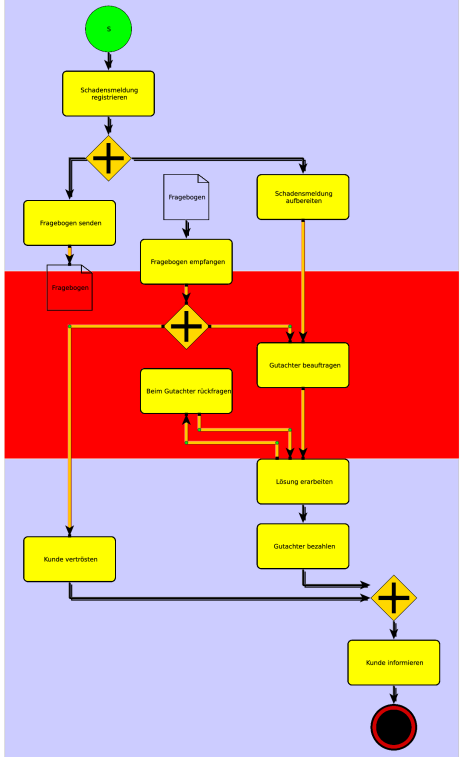
[7] S. A. White. Introduction to bpmn. *bpmn.org*, May 2004.



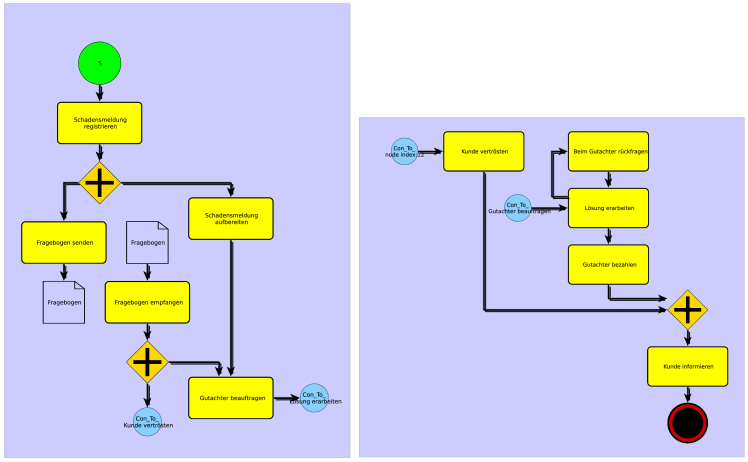
(a) Cut graph for example in Figure 4(c)



(b) Layouted graph.

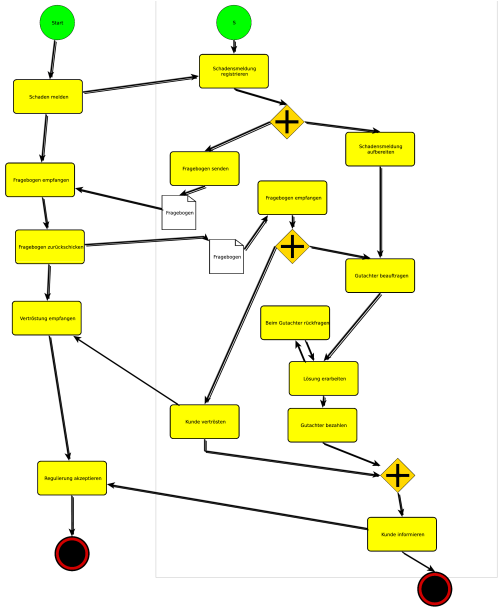


(c) Process before divison; the *center band* is marked red.

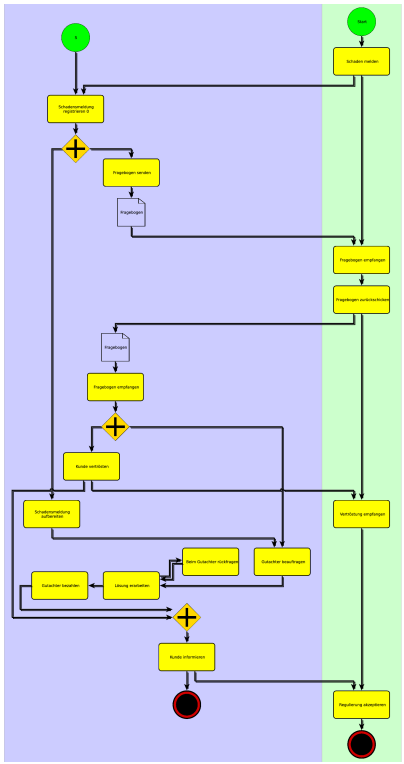


(d) Subgraphs after divison; Replacement nodes (blue) are inserted for removed division edges.

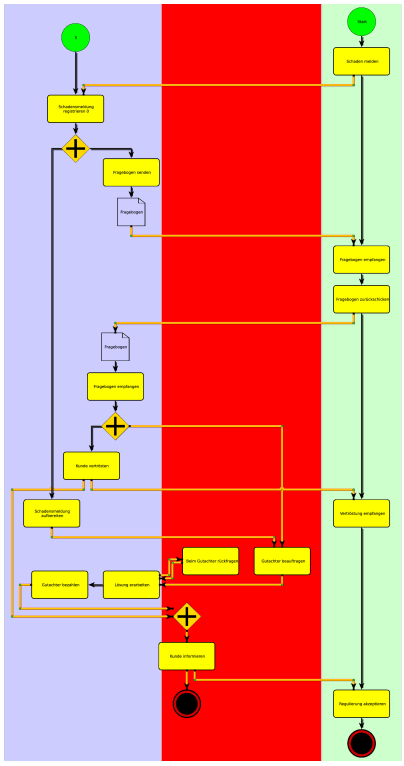
Figure 4: Example for a BPMN-Layout and division for an insurance process for notifications of claims.



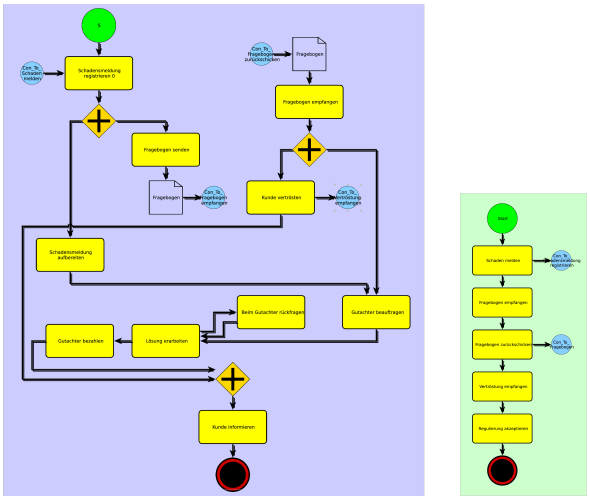
(a) A sketch showing a *BPMN* notation with two roles in notification of claim.



(b) Computed layout graph.



(c) Layout graph before vertical divison; the *center band* is marked red.



(d) Subgraphs after divison; Again, replacement nodes (blue) are inserted for division edges.

Figure 5: Example of a BPMN-Layout and a vertical division for a two-role (insurer/insurant) insurance process for notifications of claims.