Algorithmic Classification of Layouts of BPMN Diagrams

Elias Baalmann[1] and Daniel Lübke[1][0000−0002−1557−8804]

Leibniz Universität Hannover, Germany
e.baalmann@stud.uni-hannover.de,daniel.luebke@inf.uni-hannover.de

Abstract. Previous research is concerned with differences in BPMN diagram layout, e.g., with regards to understandability. However, layouts have neither been formally described nor their classification been automated. We aim at formalizing BPMN layouts and automating diagram layout classification for BPMN diagrams: We calculate sequence flow directions and encode them. By using regular expressions, these are clustered to diagram layouts. This results in a set of formally described BPMN layouts and a corresponding algorithm which we implemented in a tool. The results are very similar to previous work of manual layout classification on the GitHub process set. Researchers can use our definition when conducting BPMN diagram analysis and industry experts can use our tool for validating models against their layout guidelines.

Keywords: BPMN · Diagram Layout · Diagram Layout Formalization · Diagram Layout Detection · Flow Layout

1 Introduction

BPMN is the standard modeling language for business processes [1]. 2006 it was accepted as an OMG standard [4], the current version (BPMN 2.0) specifies multiple diagram types to model processes in different levels of detail [14]. Of the three specified types, only the process or collaboration diagram is considered in this paper. The BPMN is a documentation and communication tool which should allow readers to easily comprehend complex coherences. Thus, one key requirement for a model is understandability. Much research has been concerned with this topic recently, e.g., [6, 9, 8, 10–13].

One branch of BPMN understandability research is concerned with the layout of BPMN processes. The underlying hypothesis states that layout has a big impact on understandability. Besides small grained metrics like number of sequence flow crossings, the overall BPMN diagram layout has come into focus.

Up to now, layouts are only 'specified' by giving examples and appealing to the intuitive understanding of the reader ("top-down layout", "left-right layout"). This makes it hard to a) fully understand findings, b) replicate research and c) compare different research results. Furthermore, industry users cannot decide whether their diagrams are compliant to the latest research, thus preventing the implementation of scientists' recommendations for diagram layout.

J. Manner, D. Lübke, S. Haarmann, S. Kolb, N. Herzberg, O. Kopp (Eds.): 14th ZEUS Workshop, ZEUS 2022, Bamberg, held virtually due to Covid-19 pandemic, Germany, 24–25 February 2022, published at http://ceur-ws.org

In this paper, we want to outline a formal definition of prevalent flow layouts found in GitHub models and create an algorithm and followingly automated tool support for classifying BPMN diagram layouts. The term flow layout is chosen to clarify that the flow aspect of the Layout is considered. Other aspects, such as edge crossings or arrow lengths are not relevant here. Previous work, that investigated similar topics might use different terms such as "flow direction" [10] or "layout direction" [13]. But since 'direction' is to specific to describe the relatively complex layouts that are distinguished here, 'flow layout' seems to be a more adequate choice. By providing an implementation, that can classify diagrams based on formalized flow layouts we enable researchers to investigate statistics on large data sets such as "Does flow layout depend on the reading direction of the diagram author?" and practitioners to determine the most used layouts for example in their company and establish standards.

The paper is structured as follows: In the next section, we will introduce related work in the area of BPMN layouting, followed by a clear outline of our research questions in Sect. 3. In Sect. 4, we present the formalization and classification algorithm. The identified flow directions deducted from a large set of BPMN models found on GitHub are shown in Sect. 5 after which we conclude and provide an outlook.

2 Related Work

One of the first questions that arises in our context is how BPMN diagrams are laid out by practitioners. Effinger et al. [7, p. 400] state that "[i]n BPMN diagrams the flow direction is usually top-to-bottom or left-to right." This statement is empirically validated by Lübke & Wutke [13, p. 52], who found that 79.52% of BPMN diagrams from their GitHub data set are laid out left-to-right. They also identified other layouts, like most prominently, top-down layouts and more complex layouts like multiline and snake layouts.

A more theoretical approach is taken by Figl & Strembeck. [11, p. 60] who state that "[b]asically, there are four main options for the overall direction: left-to-right, top-to-bottom, bottom-to-top, right-to-left.", i.e., they take all four possible main directions as principal layout directions. However, they have also added that "zigzag models" should be subject to future research, thereby recognizing the use of more complex layouts in practice.

All modeling guidelines we found recommend left-to-right layouts, e.g., [2]. Even the BPMN specification itself favors left-to-right modeling [14, p. 42].

al. [5, p. 49] define a guideline (number 43) that process modelers should make their models long and thin by aligning all edges with a general workflow direction as much as possible.

However, more recently, a study by Lübke et al. [12, p. 127] has shown that the understandability of large diagrams profits from more complex layouts like snake or multiline layouts in order to avoid the penalty of scrolling these diagrams on screen. For the case of smaller diagrams, this experiment found a slight advantage for left-to-right layouts in contrast to top-down layouts, affirming Figl & Strembeck's earlier experiment. However, the findings are either minimal (some understandability metrics in the former experiment) or not significant (some metrics in the former experiment and all metrics in the latter experiment).

3 Research Questions

In this paper, we want to answer the following research questions.

RQ1: How can diagrams be classified automatically?
The automatic classification of flow layouts has many applications in research to answer questions such as "does the layout choice depend on the size of the diagram" and industry for example to enforce a style guideline.

RQ2: How can flow layouts be formalized objectively?
While formalizing all identified flow layouts is beyond the scope of this paper, we want to describe how such a formalization could be realized.

RQ3: Which are the most commonly used flow layouts, and are they worth formalizing?
By analyzing a large set of diagrams, we identify the most common layouts. Afterwards, we attempt to generalize the layouts to remove any biases that may be introduced by the data set.

4 Analyzing the Direction of Sequence Flow

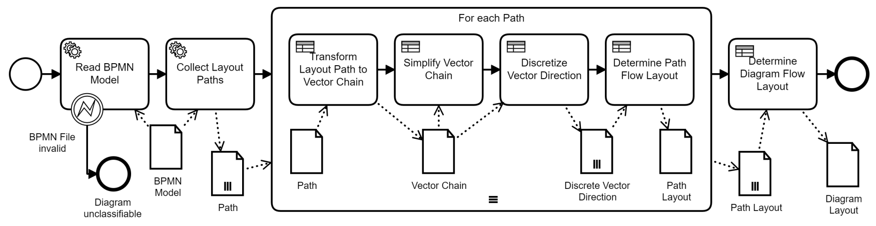

Fig. 1. Flow Layout Classification Algorithm

To classify BPMN diagrams automatically and thus answer the first research question, a modular algorithm is designed. Figure 1 illustrates the structure of the algorithm. First, the BPMN file is parsed and some sanity checks are performed to determine if the diagram can be classified at all. Since some BPMN editors do not serialize the diagrams in a standard way [1, p. 12], BPMN files exist, that are, e.g., missing layout data for the elements. For reference, an overview of the symbols and elements used in the BPMN is shown in the 'BPMN-Poster' by the BPM Offensive Berlin [3]. These diagrams cannot be classified with the current implementation. To determine the flow layout of the diagram, each path along sequence flows from any start element to any end element without loops is analyzed individually.

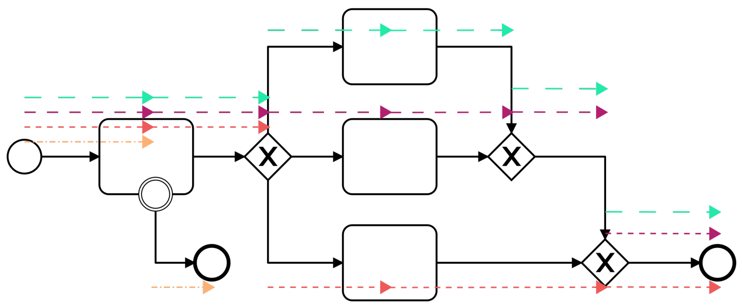

Fig. 2. BPMN Example with Vector Chain for each Layout Path in Colored Striped Arrows

The first of four tasks performed on the layout path is converting it to a vector chain. This is a list of the vectors between the centers of the flow elements on the layout path from the start to the end element. Some special cases need to be considered here. This is demonstrated by the example diagram shown in Fig. 2. Every path in the diagram is directed as straight as possible left to right. Gateways and boundary events do not allow for precisely straight layouts without overlapping the different paths.

To handle these cases, the (x or y) component of the vector between the centers of the elements (where the source element is a boundary event or a split, or where the target element is a join) that points in the orthogonal direction to the direction of the split, join or boundary event is set to zero. The direction of one element is determined by the following rules depending on the element type. **1. The element is a boundary event:** the direction is horizontal if it is connected to its parent at the top or bottom side; otherwise it is vertical. **2. The element is a split:** two cases are differentiated. Provided that the split element has an incoming sequence flow, the direction is horizontal, if the absolute value of the x component of the vector for the previous sequence flow is bigger than the absolute value of its y component. Otherwise it is vertical. The second case occurs if the split element is a start element. In this case, the direction is determined by constructing a vector that points into the average direction of the outgoing sequence flows of the split element (and comparing the x and y component as above). **3. The element is a join:** here the direction is calculated similar as for a split, just in opposite order. First, the next sequence flow is considered, and, if it does not exist (the join is an end element), the average direction of the incoming sequence flows is used. Join elements pose a problem, as the vector for the next sequence flow is not determined when the direction of the join is needed. To circumvent this issue, the direction of the first sequence flow, on the path from the join to the end element that is not entering another join, is used. If no such sequence flow exists, the average direction of the outgoing sequence flows of the join is used. The colored arrows in Fig. 2 showcase the vector chains which result from this step of the algorithm for each of the four layout paths. The vertical position of the vectors illustrates how each sequence flow is converted into a vector. In reality, only the vectors (x and y components) are relevant. Due to the rules explained above, all vertical (y) components of the vectors are set to zero, resulting in four straight vector chains from left to right.

In the next step, the vector chain is simplified by combining subsequent vectors with similar directions. The angle threshold is based on the number of discrete vector directions (NODVD) and calculated by the formula $\frac{360^\circ}{\text{NODVD}}$. The vectors in the simplified chain which are a combination of at least two vectors get marked. Marked vectors are those that lay on a straight path in the diagram with at least one element between the start and the end of that path. This marking is important as it allows us to differentiate between otherwise indistinguishable flow layouts, e.g., Multiline and Snake (see Sect. 5). After that, each vector in the resulting simplified vector chain is mapped to a discrete direction. Currently, the NODVD used in the reference implementation is 16. This value was chosen because it felt natural, as a smaller NODVD like 8 would restrict the classification to much and a higher value like 32 would prevent many combinations of vectors and thus require diagrams to adhere very closely to a specific flow layout to be classified as that layout. The calculation of the angle threshold in the previous step guarantees that no two consecutive vectors have the same discrete direction.

To determine the flow layout for the path, the list of discrete vector directions is classified using regular expressions (see Sect. 5). In the end, the flow layout for the whole diagram is defined as the flow layout that occurs for most of the paths.

5 Classifying the Diagram Flow Layout

Rather than trying to describe every possible flow layout, our goal is to find commonly used layouts, formulate their distinguishing features, and build a classification algorithm that can detect these layouts and is extendable to possibly handle other layouts that are deemed worthy of classification in the future. Lübke and Wutke identified six flow layouts while manually classifying 5299 diagrams: Left-Right, Top-Down, Snake Horizontal, Snake Vertical, Multiline Horizontal and Multiline Vertical [13]. For this paper, a larger data set from GitHub (53984 diagrams) was used to identify possibly relevant flow layouts. The data set is a super set of the one used by the before mentioned authors. Because of the vast quantity, manual classification of all diagrams is unfeasible. Thus, the process shown in Fig. 3 was used. First, the algorithm described in Sect. 4 was applied to all diagrams. The discrete vector directions determined by the algorithm were used to find common flow layouts. Though we established 16 distinct vector directions, only four distinct directions named north (N), east (E), south (S) and

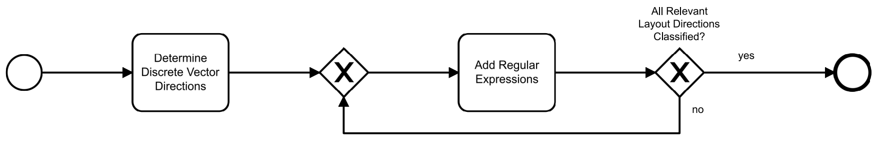
Determine Discrete Vector Directions
Add Regular Expressions
All Relevant Layout Directions Classified?
yes
no

Fig. 3. Methodology

west (W) are used in the following examples to foster comprehensibility while keeping the regular expressions manageable. The marking of the vectors (see Sect. 4) is depicted by upper-case letters for marked vectors and lower-case for non-marked vectors. Grouping the diagrams by the discrete vector directions for each layout path showed that some sequences of discrete directions occurred in multiple diagrams. For instance, 55% of all diagrams had only layout paths with the direction E and 64 diagrams had only layout paths with the sequence EsW. Manual inspection of the grouped diagrams showed that multiple direction sequences exist for the same flow layout. E.g., the sequences EsW and EsWsE would both be considered Snake Horizontal. Thus, regular expressions were constructed to classify all variations of a particular flow layout. A simplification of the regular expression for Snake Horizontal would be **EsW(sEsW)*(sE)?(s)?**. This allows for an arbitrary number of lines. This way of formalizing flow layouts with regular expressions is our way to approach RQ2.

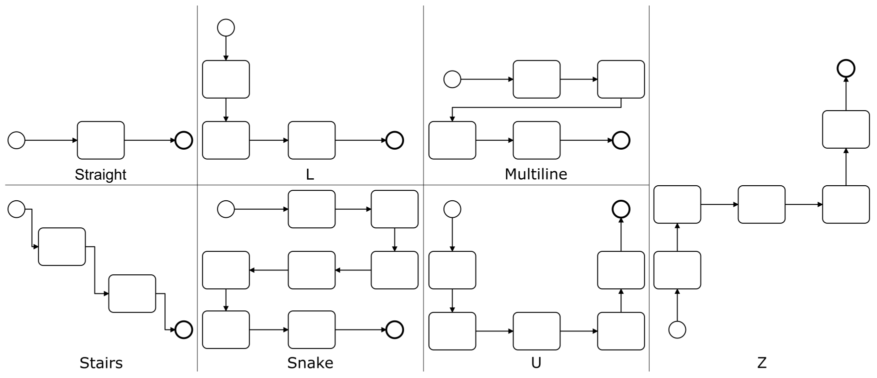

Fig. 4. The Seven Categories of Flow Layouts

The seven categories of flow layouts that have been identified to answer RQ3, are: Straight, L, Multiline, Stairs, Snake, U and Z (Fig. 4). Multiple variants of flow layouts exist for each of these categories. Left-Right, Top-Down, Right-Left and Down-Top are the four variants of the straight category. Other categories can have more distinct flow layouts. One example is the multiline category. Eight variants can be differentiated as shown in Fig. 5. Even though not all

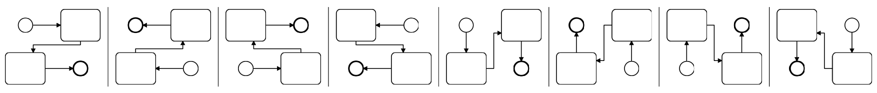

Fig. 5. Variants of Multiline Layouts

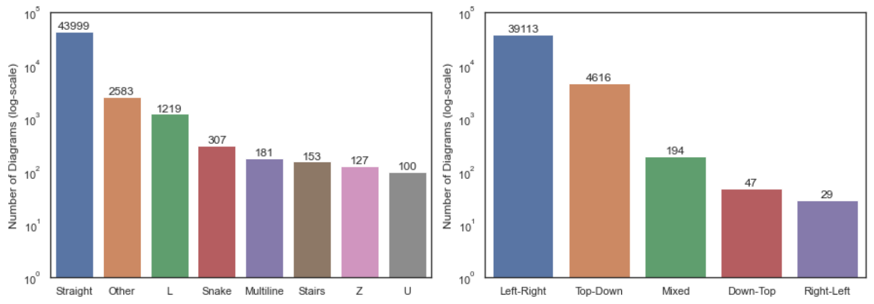

Fig. 6. Distribution of Flow Layouts in GitHub Data Set. Layout Categories (left) and Variants of Straight Flow Layout (right).

these variants occur in the data set, they are all possible multiline layouts and should thus be identifiable. This extension allows us to generalize the usability of the classification by removing biases introduced by the data set as best as possible.

Figure 6 illustrates the distribution of the automatic classification for the large data set. The left diagram shows the seven flow layouts, the right diagram the four variants of the Straight flow layout. Diagrams that could be analyzed (no file reading error, no missing layout information,...) but not classified are shown as Other. The Mixed category in the right chart contains diagrams where two thirds of the layout paths where classified as Straight but no single variant of the Straight category occurred for this many paths. Not analyzable Diagrams are not shown, 5315 of the 53984 .bpmn files where not analyzable as they contained some syntactic error or missed layout information etc. The charts demonstrate how strongly the Straight flow layout is favored especially in the Left-Right and Top-Down variants (note the logarithmic scale).

6 Conclusion & Outlook

By analyzing a large data set of BPMN diagrams, we demonstrated that there are many flow layouts which are used for multiple diagrams. Subsequently, we identified seven categories of commonly used layouts. Formalizing flow layout with the use of regular expressions on discretized vector chains for each layout path is sensible. The formalization allowed us to create an algorithm and a tool implementation that showed promising results and can, e.g., be used by researchers to answer questions such as how diagram layouting differs between less experienced users and experts of BPMN. The tool can also be used by teams in the industry to validate models against their layout guidelines. This paper provides a concise overview of our work but fails to describe every detail of the complex subject that is layout detection. Examples of aspects that were considered but not explicitly reported in this paper are: the impact of swimlanes or subprocesses on flow layouts and how an accuracy score can be determined to indicate how exact a diagram is adhering to a particular flow layout. Furthermore some parts of the parameterization of the algorithm where chosen by feel and might appear arbitrary. For example determining the optimal NODVD based on more scientific metrics than 'does it feel natural' could be an interesting topic for future work to investigate.

References

1. Allweyer, T.: BPMN 2.0: Introduction to the standard for business process modeling. BOD - Books on Demand, Norderstedt, 2nd, updated and extended edition edn. (2016)
2. Birchler, A., Bosshart, E., Märki, M., Opitz, P., Pauli, J., Rigert, B., Sandoz, Y., Schaffroth, M., Spöcker, N., Tanner, C., Walser, K., Widmer, T.: eCH-0158 BPMN-Modellierungskonventionen für die öffentliche Verwaltung. WWW: https://www.ech.ch/dokument/fb5725cb-813f-47dc-8283-c04f9311a5b8 (September 2014)
3. BPM Offensive Berlin: BPMNPoster – www.bpmb.de (2011), http://www.bpmb.de/index.php/BPMNPoster
4. Chinosi, M., Trombetta, A.: Bpmn: An introduction to the standard. Computer Standards & Interfaces **34**(1), 124–134 (2012). https://doi.org/10.1016/j.csi.2011.06.002
5. Corradini, F., Ferrrari, A., Fornari, F., Gnesi, S., Polini, A., Re, B., Spagnolo, G.O.: Quality assessment strategy: applying business process understandability guidelines for learning, http://pumax.isti.cnr.it/dfdownloadnew.php?ident=cnr.isti/cnr.isti/2015-TR-03&langver=i&scelta=Metadata
6. Corradini, F., Ferrari, A., Fornari, F., Gnesi, S., Polini, A., Re, B., Spagnolo, G.O.: A guidelines framework for understandable BPMN models. Data & Knowledge Engineering **113**, 129–154 (2018). https://doi.org/10.1016/j.datak.2017.11.003
7. Effinger, P., Siebenhaller, M., Kaufmann, M.: An interactive layout tool for BPMN. In: 2009 IEEE Conference on Commerce and Enterprise Computing. pp. 399–406 (2009). https://doi.org/10.1109/CEC.2009.36
8. Effinger, P.: Layout patterns with BPMN semantics. In: Dijkman, R., Hofstetter, J., Koehler, J. (eds.) Business Process Model and Notation. pp. 130–135. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
9. Effinger, P., Jogsch, N., Seiz, S.: On a study of layout aesthetics for business process models using BPMN. In: Mendling, J., Weidlich, M., Weske, M. (eds.) Business Process Modeling Notation. pp. 31–45. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
10. Figl, K., Strembeck, M.: On the importance of flow direction in business process models. In: 2014 9th International Conference on Software Engineering and Applications (ICSOFT-EA). pp. 132–136. IEEE Computer Society, Los Alamitos, CA, USA (2014). https://doi.org/10.13140/2.1.3445.8247

11. Figl, K., Strembeck, M.: Findings from an experiment on flow direction of business process models. In: Kolb, J., Leopold, H., Mendling, J. (eds.) Enterprise modelling and information systems architectures. pp. 59–73. Gesellschaft für Informatik e.V, Bonn (2015)
12. Lübke, D., Ahrens, M., Schneider, K.: Influence of diagram layout and scrolling on understandability of BPMN processes: an eye tracking experiment with BPMN diagrams. Information Technology and Management **22**(2), 99–131 (2021). https://doi.org/10.1007/s10799-021-00327-7
13. Lübke, D., Wutke, D.: Analysis of prevalent BPMN layout choices on GitHub. In: Manner, J., Haarmann, S., Kolb, S., Herzberg, N., Kopp, O. (eds.) Proceedings of the 13th European Workshop on Services and their Composition (ZEUS 2021), Bamberg, Germany, February 25-26, 2021. CEUR Workshop Proceedings, vol. 2839, pp. 46–54. CEUR-WS.org (2021), http://ceur-ws.org/Vol-2839/paper9.pdf
14. Object Management Group: Business process model and notation (BPMN), version 2.0, https://www.omg.org/spec/BPMN/2.0/PDF