How to Draw a Sequence Diagram

Timo Poranen*, Erkki Mäkinen**, and Jyrki Nummenmaa

Department of Computer and Information Sciences
Kanslerinrinne 1
FIN-33014 University of Tampere
Finland
{tp,em,jyrki}@cs.uta.fi

Abstract. In this paper, we consider the aesthetic criteria and constraints related to the layouts of UML sequence diagrams. We consider the applicability of the traditional graph drawing aesthetics in drawing sequence diagrams. Because of the special nature of sequence diagrams, many of these aesthetics are not applicable. Based on our view on how these diagrams are read or viewed, we propose some new aesthetics. We also take into account the presence of usually adopted conventions and constraints.
The basic choice in producing a drawing for a sequence diagram is the linear order of the participating objects. Based on this finding and the identified aesthetics criteria, we formulate some related computational problems.

1 Introduction

The Unified Modeling Language (UML) [21] is currently the standard notation for modeling software-intensive systems. In UML, *sequence diagrams* are typically used to describe system dynamics. Sequence diagrams depict system dynamics by showing the participating objects (classes, components, etc.) in the interaction and the sequence of messages exchanged.

A sequence diagram has two dimensions: the vertical dimension represents time and the horizontal dimension represents the objects participating in the interaction. Time flows from top to bottom. Objects (or classifier roles, more generally) are shown as vertical lines (called *lifelines*) and messages as horizontal arrows extending from a sender object to a receiver object. Spacing is irrelevant, that is, only the order of messages matters, not the distance between them.

The diagrams produced by analysts are typically not very large. Based upon our experiments, they may contain something like 5-15 lifelines and up to 30 messages. If they tend to get larger, they are usually decomposed hierarchically. However, reverse engineering methods produce diagrams for which these figures

* Work funded by the Tampere Graduate School in Information Science and Engineering (TISE) and supported by the Academy of Finland (Project 51528).
** Work supported by the Academy of Finland (Project 35025).

are not necessarily true at all. They may be much larger both in terms of the number of lifelines and the number of messages.

The relative ordering of objects on the horizontal dimension has no semantic significance in standard UML, but as Rumbaugh et al. [21], p. 424 say: "it is helpful to arrange them to minimize the distance that the message arrows must cover". This paper is devoted to a study of what constitutes a good drawing for a sequence diagram and what are the related computational problems. We also discuss the applicability of some computational methods for solving this problem.

We will also discuss the quality of drawings based on (1) the basic nature of sequence diagrams, (2) some usually adopted optional constraints, and (3) aesthetic criteria to be defined in the sequel.

The rest of this paper is organized as follows. In Section 2 we give a brief introduction on sequence diagrams and their use in visualizing the behaviour of software systems. In the next section we introduce constraints for sequence diagram layouts and in Section 4 we study aesthetic criteria for these layouts. The fifth section studies the methods to obtain layouts that satisfy a given set of aesthetic criteria and constraints. Also the computational complexity related to fulfilling these criteria and constrains are considered. The sixth section gives some final conclusions.

2 Sequence Diagrams

Sequence diagrams describe how objects, or groups of objects, interact within a system [21]. Interacting objects can, for example, be classes, program components or real world instances such a customer who is buying a train ticket in the station.

Figure 1 shows a sample sequence diagram modeling the action of buying a train ticket. Throughout this paper, we assume that there is no self-referring messages and all message arrows are parallel to x-axis, that is, we do not consider such message arrows that have nonzero duration. Rumbaugh et al. [21] draw arrows having nonzero duration diagonally downward so that the receiving time is later than the sending time. Also, we assume that all lifelines describing participants are visible all the time. These restrictions are made to simplify some mathematical formalizations in Section 5.

3 Constraints

Constraints are used to provide semantic information about the meaning of the drawing in order to better reflect the features of the underlying model. These types of instructions usually cannot be automatically deduced by a diagram layout algorithm.

The drawing constraints of sequence diagrams can be divided into two classes: those that are general for all layouts and those defined using the input.

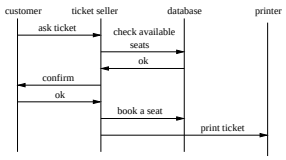


Fig. 1. An interaction between a customer, ticket seller, database and printer.

3.1 General Constraints

General constraints are commonly used graphical standards for all layouts. It is possible to find three important constraints of this class.

Table 1. General constraints for sequence diagrams.

	General constraint	Description
GC1	**Horizontal distance**:	Uniform horizontal distances between participants
GC2	**Vertical distance**:	Uniform vertical distances between message arrows
GC3	**Starting object**:	The object that starts communication is drawn to the left

The first one describes the properties of message arrows. The vertical dimension of the layout represents time and the vertical distances between messages submitted at different times are uniform. That is, when an object submits messages within different time periods, the vertical distances between corresponding message arrows do not depend on this time information. The second constraint states that the horizontal distances between subsequent lifelines are uniform. The last one indicates that the object that starts communication is drawn as the leftmost participant. These general constraints are listed in Table 1.

Of course, there might be exceptions for these principles. We give examples for each of these principles where breaking the corresponding general constraint is justified. If we want to write a longer description for submitted messages, the horizontal distance between some participants should be increased. When exact running time is important, exact starting times for messages should be read from the vertical position of message arrows. Therefore, the time axis should be metric. So far, we have not seen any cases where the last standard is not adopted, but it is not hard to imagine a case where starting object has a visually informative position, say, in the center of the drawing.

3.2 Input Dependent Constraints

The constraints in the previous section did not depend on the input. Input dependent constrains that are commonly used in traditional graph drawing [5, 13, 25] are listed in Table 2. Next we discuss the possibility to apply these traditional drawing constraints to the layouts of sequence diagrams.

The first constraint in Table 2 describes a need to place a given set of vertices to the center of the drawing. This need is justified since the center of a drawing is usually the most important and prominent place where vertices can be placed. But this argumentation does not hold for sequence diagram layouts. We think that the most important place to draw a participant in a sequence diagram is the leftmost lifeline or, sometimes, the rightmost lifeline.

The second constraint tries to keep a set of vertices in a drawing close together. This holds also for sequence diagrams: there might be a set of lifelines that belong closely together.

Table 2. Constraints in traditional graph drawing.

Constraint	Description
Center:	Place a given subset of vertices close to the center of the drawing
Cluster:	Place a given subset of vertices close together
Left-right (top-bottom):	Draw a given subset of vertices from left to right (from top to bottom)
Shape:	Draw a given subset of vertices with a predefined shape
External:	Place a given subset of vertices on the outer boundary of the drawing

Many graph drawing algorithms use different kinds of hierarchical approaches (see, for example [23]) to produce readable layouts. The basic principle behind all hierarchical graph drawing algorithms is that of dividing the set of vertices into layers and then draw these layers from left to right (from top to bottom). When we want to draw a given subset of vertices from left to right (from top to bottom) the way of assigning the vertices to different layers makes it possible to achieve this constraint. This approach is also suitable for sequence diagrams. It is natural that a set of participants needs to be drawn in a predefined order from left to right in the sequence diagram layout.

The shape constraint draws a given subset of vertices with a predefined shape. For sequence diagrams, this constraint is useful, but we prefer saying that a set of vertices will be drawn with a predefined order (permutation), than with a predefined shape.

For planar graphs, the set of external vertices plays an important role in the drawing. In the layouts of sequence diagrams, the outermost (the leftmost and the rightmost) participants play for the role of external vertices. As discussed above, especially the leftmost position is of utmost importance in these layouts.

All these five constraints share the property that they cannot be automatically deduced by a diagram layout algorithm. Hence, the user needs to give them as additional input. The same holds for the sequence diagrams. The layout algorithms cannot recognize which are important vertices and which are not.

Since the center constraint is not useful for sequence diagrams, and there is a need to place important participants in the left end of the drawing, we introduce a new constraint to replace the constraints Center and External: *Fixed place* constraint allows to place a given lifeline on a given position in the layout. Constraints for the sequence diagrams are collected in Table 3.

Next, we give an example of using constraints in sequence diagram layouts. Suppose that the participating objects can be divided into three sets depending on their role: there are controller, model or view objects in corresponding system (that is, the MVC architecture [9] is used). After this division, corresponding sets of objects could be drawn from left to right in such an order that the set of model objects are drawn first, view objects are drawn next and the set of controller

Table 3. Constraints for sequence diagrams.

Constrain	Description
C1 **Cluster**:	Place a given subset of lifelines close together
C2 **Left-right**:	Draw a given subset of lifelines from left to right
C3 **Order**:	Draw a given subset of lifelines with a predefined order
C4 **Fixed place**:	Place a given lifeline on a given position

objects are drawn last in the right. In this example we used Cluster (C1) and Left-right (C2) constraints.

4 Aesthetic Criteria

An *aesthetic criterion* is a general graphical property of the layout that we would like to have. A well chosen aesthetic criterion improves the readability of the given layout. Commonly used aesthetic criteria for traditional graph drawing (see, for example, [3, 5, 20, 23]) are listed in Table 4.

Table 4. Commonly used aesthetic criteria for traditional graph drawing.

Aesthetic	Description
Total edge length:	Minimize the total edge length
Crossings:	Minimize the total number of edge crossings
Maximum edge length:	Minimize the maximum edge length
Uniform edge length:	Minimize the variance of the edge lengths
Symmetry:	Maximize the symmetry in the drawing
Area:	Minimize the area of the drawing
Total bends:	Minimize the number of bends in the drawing
Maximum bends:	Minimize the maximum number of bends on the edges
Uniform bends:	Minimize the variance of the number of bends on the edges
Angular resolution:	Maximize the smallest angle between edges incident to same vertex
Aspect ratio:	Minimize the ratio of the smallest rectangle covering the drawing
Balance:	Distribute the vertices uniformly over the drawing area

The first two aesthetics of the traditional graph drawing coincide when they are used in sequence diagram layouts. This property is easy to see by noticing that each message arrow whose length is, say l, increments the total number of crossings by $l - 1$ (although here the edges cross with the lifelines).

In addition to the total arrow length, also other aesthetic criteria can be applied when laying out sequence diagrams. Since it is especially difficult to follow

long arrows, a natural aesthetic rule is to limit the maximal length of the arrows, or at least to decrease the number of the longer arrows. The corresponding criterion for traditional graph drawing is to minimize the maximum edge length.

The minimization of the variance of the edge length is not as obvious as the earlier criteria. In fact, this criterion often contradicts with the minimization of the total edge length. It might be impossible to shorten some of the edges. Then, minimizing the variance of the edge lengths would mean that the other edges should be made longer.

Displaying as much as possible symmetries in traditional graph drawing is perhaps the most important aesthetic criteria to achieve. For two-dimensional graph drawing, symmetry is defined by using Euclidean symmetries, that is, rotations and reflections [14]. If a sequence diagram has some symmetries, it would help to understand the structure of the diagram and the structure of the system behind corresponding participants. Therefore, we do not reject the symmetry criterion, but we leave it an open problem as to how to define symmetry in the context of sequence diagrams.

There are no bends of the edges in the sequence diagrams. Therefore all aesthetics where bends are considered are useless for our purposes. The same reasoning holds for angular resolutions, since there are no angles between message arrows.

Also there is no way to modify the width and the height of the sequence diagrams, so the aspect ratio and balance criteria can be rejected.

In addition to these criteria, we introduce three new aesthetic criteria for sequence diagrams: *sliding*, *the number of long edges* and *the subset separation*.

A natural way to view sequence diagrams is to start the viewing from the top left corner, where the leftmost participant starts the communication and then follow message arrows while sliding the diagram downward. Suppose that the diagram contains so many lifelines of participants, that the monitor screen is not able to show all lifelines at once. If there are no message arrows that have their start and endpoint outside the screen, we have no problems, but if this is not true, it is very hard to view the diagram and guess which participants are interacting. In a way we would like to slide a fixed-size window over the sequence diagram in such a way that the window always covers all the arrows between all lifelines at all visible x-coordinates. We call this the *sliding* property. See Figure 2 where the sequence diagram has a good slidability.

Next new criterion is *the number of long edges*. Although it might be impossible to avoid a single long edge, it is obviously advantageous to have as few long edges as possible. This is done when obeying the minimization of the number of long edges. Recall that this criterion often contradicts the traditional criterion of minimizing the variance of the edge lengths. For this property, the minimum length of a long edge must be given.

The last new criterion, *the subset separation*, plays an important role in visualizing a system having such subsets of participants, that do not communicate together, or do communicate very little. Suppose that there are two distinct sets of participants, and one participant, a filter, who receives and forwards all

messages from those two sets. Now all communication between those two sets of participants goes first to the filter participant which forwards messages to a participant in the second set. It is natural to place this filter participant in the middle of the diagram and then place other two sets of participants to the left and to the right side. The goal of the subset separation property is to find out distinct subsets of participants that have as little as possible communication. Then these sets are drawn together to a cluster (C1) and these clusters are ordered from left to right (C2) to minimize the number of over going message arrows. Aesthetic criteria for the sequence diagrams are collected in the Table 5.

Table 5. Aesthetic criteria for sequence diagrams.

	Aesthetic	Description
A1	**Crossings**:	Minimize the total number of edge crossings
A2	**Maximum edge length**:	Minimize the maximum edge length
A3	**Uniform edge length**:	Minimize the variance of the edge lengths
A4	**Symmetry**:	Maximize the symmetry in the diagram
A5	**Number of Long edges**:	Minimize the number of the long edges
A6	**Sliding**:	Maximize the slidability of the diagram
A7	**Subset separation**:	Maximize the distinct subsets of participants

User preferences of the layout aesthetics of some classes of UML diagrams are studied by Purchase et al. [19].

5 Criteria Formalization

In this section we formalize the following aesthetic criteria: crossings (A1), maximum edge length (A2), sliding (A5) and subset separation (A7). The remaining aesthetics are also discussed, but their exact formalization is not considered here. For the basic graph-theoretical concepts used in this section, we refer to [24].

The most obvious goal is to minimize the total length of message arrows (A1). Minimizing the total arrow length can be interpreted as a graph problem in the following way: replace each participant (lifeline) with a node, and insert an (undirected) edge with capacity k between nodes u and v, if there are k messages between the participants corresponding to nodes u and v (the direction of the arrows is irrelevant here). As an example, consider the sequence diagram and corresponding weighted graph in Figure 3. The total weighted edge length sum in the diagram of Figure 3 is 18. For example, the order D-A-C-B-E of the nodes gives total length 13.

Minimizing the total arrow length in a sequence diagram coincides with arranging the nodes of a weighted undirected graph on a line. Next, we formally define the graph problem in question.

Consider an undirected graph $G = (V, E)$ with $n = |V|$ nodes and with positive edge capacities $c(e)$. A (linear) *layout* ϑ of G is a bijection $\vartheta : V \rightarrow$

$\{1, 2, \ldots, n\}$. The *optimal linear arrangement problem* (also known as the minimum linear arrangement, or the edge sum problem) calls for the layout that minimizes the weighted sum of the edge lengths, that is, the sum

$$\sum_{(u,v)\in E} c(u,v) \cdot \mid \vartheta(u) - \vartheta(v) \mid .$$

The optimal linear arrangement problem (OPT) is NP-complete, even with constant edge capacities [11] and for bipartite graphs [8]. It is unknown if OPT is solvable for weighted trees, but if the given graph is a rooted weighted tree, then the optimal layout can be constructed in $O(n \log n)$ time [1]. OPT is solvable for unweighted and undirected trees in $O(n^{\log 3/\log 2})$ time [4]; an $O(n^{2.2})$ algorithm is presented by Shiloach [22]. A collection of theoretical results for OPT and other arrangement problems is given by Díaz et al. [6]. A recent survey of the complexity and approximability results of OPT with constant edge capacities can be found in [18].

OPT has **PTAS** (polynomial-time approximation scheme [10]) for dense graphs [2], and there exists an $O(\log n \log\log n)$ approximation algorithm for arbitrary graphs [7]. However, when considering sequence diagrams, the asymptotic complexity results are of little value, since a typical instance contains only 5-15 nodes.

Since it is especially difficult to follow long arrows, the aesthetic criterion A2 limits the maximal length of the arrows. The corresponding graph problem is the *bandwidth* problem.

In the bandwidth problem the task is to find a layout that minimizes the maximal edge length, that is,

$$\max_{(u,v)\in E} \{\mid \vartheta(u) - \vartheta(v) \mid\},$$

is minimized over all layouts. This problem does not use the edge capacities. Also the bandwidth problem is NP-complete [17]; there is even no **APX** for it [27] (that is, the problem does not allow polynomial-time approximation algorithms with a constant approximation ratio) and the bandwidth problem for trees with maximum degree 3 is NP-complete [10].

Table 6. Computational complexity of the aesthetics for sequence diagrams.

Aesthetic	Problem	Graph class	Complexity class	Ref.
A1	OPT	general graphs	**NP**-complete	[11]
		bipartite graphs	**NP**-complete	[8]
		unweighted trees	**P**	[4, 22]
		rooted weighted trees	**P**	[1]
		weighted trees	Unknown	
A2	Bandwidth	general graphs	**NP**-complete	[17]
		trees ($deg \leq 3$)	**NP**-complete	[10]

Decreasing the bandwidth decreases the maximal arrow length, but it might increase the total weighted arrow length sum.

As for the sliding property (A5), in practice we want to find a reasonable-sized window and such a layout that slidability takes place. The vertical dimension and the horizontal dimension obviously have a close connection. Given the vertical dimension for the window, we want to find such a horizontal dimension for the window and such a layout, that we get slidability. Suppose that for a given vertical dimension, we have found the minimum horizontal dimension such that a layout with slidability exists. Then, increasing the vertical dimension either keeps this minimum horizontal dimension the same or increases it. Obviously, the horizontal dimension can not be smaller than the length of the longest arrow in the chosen layout. The computational complexity of these slidability problems is not known.

To divide the vertices of a graph into distinct subsets by using some criteria, is in general very hard problem. One way to characterize the separability (A7) of a graph is to find out its cut vertices. Deleting a cut vertex of the graph divides its into two or more connected components. Determining all cut vertices of a graph can be done in linear time [26]. These connected components can be drawn from left to right. If the corresponding graph is k-connected (to make the graph disconnected, at least k vertices have to be deleted), then the problem is to find a set of k vertices, whose removal makes the graph disconnected. These k vertices are drawn in the middle of the layout, and remaining connected components are drawn from left to right. It is obvious that determining the vertex separation property is useful only for small values of k.

Determining whether a given graph can be drawn symmetrically is an NP-complete problem [15]. For positive results, there are polynomial time algorithms for trees [16] and series-parallel digraphs [12] to determine maximum number of symmetries. Unfortunately, it is not clear how symmetry should be defined in the context of sequence diagrams.

Although the resource requirements for different aesthetics are often exponential, for diagrams with less than nine lifelines all possible layouts can be searched exhaustive in short time.

6 Conclusions and Open Problems

In this paper we have studied the aesthetic criteria and constraints of the layouts of UML sequence diagrams. We represented general and input dependent constraints and defined aesthetic criteria for the sequence diagrams. A part of the constraints were also formally defined.

There are several obvious subjects for future work. The status and the usability of the symmetry criterion is not clear and also other criteria should be defined more specifically.

In the future we will study heuristic algorithms for the basic aesthetics and for some of their combinations. Our future aim is to improve the quality of sequence diagrams layouts used in practical software engineering tools.

References

1. D. Adolphson and T.C. Hu. Optimal linear ordering. *SIAM Journal on Applied Mathematics*, 25(3):403–423, 1973.
2. S. Arora, A.M. Frieze, and H. Kaplan. A new rounding procedure for the assignment problem with applications to dense graph arrangements. In *37th Annual Symposium of Foundations of Computer Science*, pages 21–30, 1996.
3. C. Batini, L. Furlani, and E. Nardelli. What is a good diagram? In *4th International Conference on the Entity Relationship Approach*, pages 312–319, 1985.
4. F.R.K. Chung. On optimal linear arrangements of trees. *Computers and Mathematics with Applications*, 10(1):43–60, 1984.
5. G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis. *Graph Drawing*. Prentice Hall, 1999.
6. J. Díaz, J. Petit, and M. Serna. A survey of graph layout problems. *ACM Computing Surveys*, 34(3):313–356, 2002.
7. G. Even, J. Naor, S. Rao, and B. Schieber. Divide-and-conquer approximation algorithms via spreading metrics. In *36th Annual Symposium of Foundations of Computer Science*, pages 62–71, 1995.
8. S. Even and Y. Shiloach. NP-completeness of several arrangements problems. Technical report TR-43, The Technion, Haifa, 1978.
9. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Orientated Software*. Addison-Wesley, 1995.
10. M.R. Garey, R.L. Graham, D.S. Johnson, and D.E. Knuth. Complexity results for bandwidth minimization. *SIAM Journal on Applied Mathematics*, 34(3):477–495, 1978.
11. M.R. Garey and D.S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W.H.Freeman, 1979.
12. S. Hong, P. Eades, A. Quigley, and S. Lee. Drawing algorithms for series parallel digraphs in two and three dimensions. *Lecture Notes in Computer Science*, 1547:198–209, 1998.
13. C. Kosak, J. Marks, and S. Shieber. Automating the layout of network diagrams with specified visual organization. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(2):440–454, 1994.
14. R.J. Lipton, S.C. North, and J.S. Sandberg. A method for drawing graphs. In *1st Annual ACM Symposium on Computational Geometry*, pages 153–160, 1985.
15. J. Manning. Computational complexity of geometric symmetry detection in graphs. *Lecture Notes in Computer Science*, 507:1–7, 1989.
16. J. Manning and M.J. Atallah. Fast detection and display of symmetry in trees. *Congressus Numerantium*, 64:159–169, 1989.
17. C. Papadimitriou. The NP-completeness of the bandwidth minimization problem. *Computing*, 16:263–270, 1976.
18. J. Petit. *Layout Problems*. PhD thesis, Universitat Politècnica de Catalunya, 2001.
19. H.C. Purchase, J-A. Allder, and D. Carrington. Graph layout aesthetics in UML diagrams: user preferences. *Journal of Graph Algorithms and Applications*, 6(3):255–279, 2002.
20. H.C. Purchase, R.F. Cohen, and M. James. Validating graph drawing aesthetics. *Lecture Notes in Computer Science*, 1027:435–446, 1996.
21. J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1999.

22. Y. Shiloach. A minimum linear arrangement algorithm for undirected trees. *SIAM Journal on Computing*, 8(1):15–32, 1970.
23. K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.
24. M.N.S. Swamy and K. Thulasiraman. *Graphs, Networks and Algorithms*. John Wiley and Sons, 1981.
25. R. Tamassia, G. Di Battista, and C. Batini. Automatic graph drawing and readability of diagrams. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):61–79, 1988.
26. R.E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.
27. W. Unger. The complexity of the approximation of the bandwidth problem. In *39th Annual Symposium of Foundations of Computer Science*, pages 82–91, 1998.

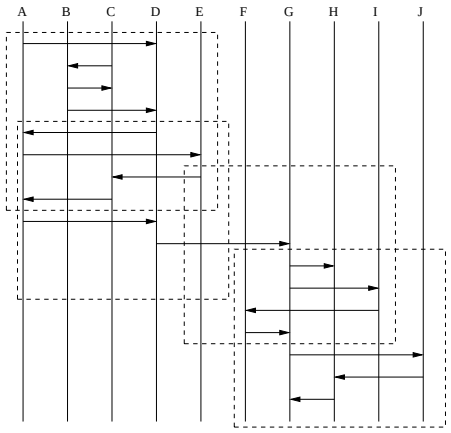
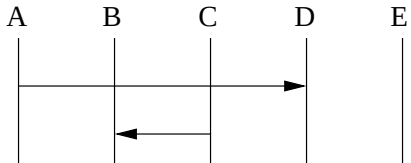


Fig. 2. An example of viewing large sequence diagram with the sliding property.


A
B
C
D
E