University of Tampere, Finland

http://www.uta.fi/

How to draw a sequence diagram

Timo Poranen, Erkki Mäkinen and Jyrki Nummenmaa

University of Tampere
Department of Computer and Information Sciences
33014 University of Tampere
email: {tp, em, jyrki}@cs.uta.fi
What’s the problem? Motivation

- Sequence diagrams (SD) describe how objects interact within a system.
- Although the diagrams produced by analysts are typically small, the reverse engineering methods may produce very large sequence diagrams.
- The basic choice in producing a drawing for a sequence diagram is the linear order of the participating objects.
- The relative ordering of objects on the horizontal dimension has no semantic significance in standard UML, but as Rumbaugh et al. say:
  
  “it is helpful to arrange them to minimize the distance that the message arrows must cover”

- What constitutes a good drawing for a sequence diagram and what are the related computational problems?
Presentation structure

1. Our framework for sequence diagrams
2. Related work
3. What constitutes a good drawing for sequence diagrams:
   - general constraints
   - input dependent constraints
   - aesthetic criteria
4. Some criteria formalization
5. Computational complexity and computation methods
6. Conclusions and open problems
Our framework for SDs

Restrictions are made to simplify formalizations:

• No self-referring messages

• All message arrows are parallel to $x$-axis (we do not consider messages having non-zero duration)

• All lifelines are visible all the time
Related work

- In the traditional graph drawing the research of the aesthetics and constraints have long history, but there is no results for SDs
- Many, but not all, of the computational problems arising from SDs are studied well in the literature
- Many suitable optimization methods exist already for corresponding combinatorial problems
Constraints

- The drawing constraints of sequence diagrams can be divided into two classes:
  - those that are general for all layouts (drawing convention) and
  - those defined using the input
- General constraints are the special rules that the drawing have to satisfy
- Input dependent constraints provide semantic information about the meaning of the drawing. Example: place “most important” participant in the leftmost position
- These types of instructions usually cannot be automatically deduced by a diagram layout algorithm
- User needs to give them as additional input
General constraints

• **Horizontal distance**: uniform horizontal distances between participants
  – Exception: longer description for messages

• **Vertical distance**: uniform vertical distances between message arrows
  – Exception: metric time axis

• **Starting object**: the object that starts communication is drawn to the left
  – Exception: starting object is drawn in the center due to some input dependent constraint or aesthetic criterion
Input dependent constraints

- **Cluster**: place a given subset of lifelines close together
  - no matter with the relative order

- **Left-right**: draw a given subset of lifelines from left to right
  - only the relative position matters

- **Order**: draw a given subset of lifelines with a predefined order

- **Fixed place**: place a given lifeline on a given position
Constraint example

- Suppose that participating objects can be divided into three sets depending on their role in the system (MVC architecture is used):
  - model (M), view (V) and controller (C) objects
- Now corresponding sets of objects could be drawn from left to right
- In this example we used cluster and left-right constraints
Aesthetic criteria

• An *aesthetic criterion* is a general graphical property of the layout that we would like to have

• Example: minimize the total message arrow length

• A well chosen aesthetic criterion improves the readability of the given layout
Aesthetic criteria for sequence diagrams

• The most important aesthetic criteria are:
  – Crossings: minimize the total number of edge crossings
  – Maximum edge length: minimize the maximum edge length
  – Sliding: maximize the slidability of the (large) diagram

• Usefull:
  – Uniform edge length: minimize the variance of the edge lengths
  – Number of long edges: minimize the number of the long edges
  – Subset separation: find out distinct subsets of participants that have as little as possible communication

• Unknown status:
  – Symmetry: maximize the symmetry in the diagram
Example on viewing large diagram 1/3
Example on viewing large diagram 2/3

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram with arrows connecting different points.
Example on viewing large diagram 3/3
Criteria formalization

• The aesthetics **crossings** and **maximum edge length** are can be defined exactly

• The exact formalization of the remaining aesthetics are more or less open problems

• Formalization can be done using elementary graph theory
From SDs to graphs

1. Replace each participant (lifeline) with a node
2. 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)
Minimizing the crossings

- Minimizing the total arrow length coincides with the optimal linear arrangement problem (OPT)

- Consider an undirected graph $G = (V, E)$ with $n = |V|$ nodes and with positive edge capacities $c(e)$

- A (linear) layout $\vartheta$ of $G$ is a bijection $\vartheta : V \rightarrow \{1, 2, \ldots, n\}$

- OPT calls for the layout that minimizes the weighted sum of the edge lengths, that is, the sum

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

is minimized
Minimization of the maximal edge length

- The corresponding graph problem is the *bandwidth* problem
- In the bandwidth problem the task is to find a layout that mini-
mizes the maximal edge length, that is,

\[ \max_{(u,v) \in E} \{| \vartheta(u) - \vartheta(v) | \}, \]

is minimized over all layouts
- Usually the optimization of one aesthetic contradicts with the op-
timization of another aesthetic!
Computation complexity

- OPT and bandwidth are NP-complete
- Some special cases are solvable in polynomial time
- For diagrams with less than nine lifelines all possible layouts can be searched exhaustive in short time
- For larger instances many optimization methods exist: hillclimbing, simulated annealing, genetic algorithms
- In general, a combination of aesthetic criterion and constraints should be optimized
- The main problem: what to compute!
Conclusions and future work

• Done:
  – We have recognized the aesthetic criteria and constraints of the layouts of UML sequence diagrams
  – Defined some aesthetic criteria formally

• Todo:
  – Many aesthetic criteria should to be defined and studied more specifically
  – Heuristic algorithms are needed for the basic aesthetics and for some of their combinations
  – Implementations of the layout algorithms for FUJABA: \(\text{http://www.uni-paderborn.de/cs/fujaba/}\)