Evolutionary Layout of UML Class Diagrams

J. Wolff v. Gudenberg*
Würzburg University
Informatik
D-97074 Würzburg

A. Niederle
Würzburg University
Informatik
D-97074 Würzburg

M. Ebner
Würzburg University
Informatik
D-97074 Würzburg

H. Eichelberger†
Hildesheim University
Software Engineering
31141 Hildesheim

Abstract

An evolutionary algorithm that layouts UML class diagrams is developed and described. It evolves the layout by mutating the positions of class symbols, inheritance relations, and associations. The process is controled by a fitness function that is computed from several well-known and some new layout metrics.

Keywords: UML class diagrams, evolutionary algorithms, layout metrics, graph drawing

1 The Algorithm

A neat and aesthetic drawing of a UML class diagram supports readabilty and usability. In this paper we investigate a new lightweight evolutionary non-deterministic approach for automatic layout of UML class diagrams.

1.1 Representation

We represent a diagram as a set of nodes and edges. Each node has a fixed size that does not change during the course of evolution and, hence, is not stored with each diagram. A node is represented by the x and y coordinates of its upper left corner point.

An edge connects two nodes, the coordinates of the junction points are stored relatively to the top left corner of the node rectangle. Hierarchical edges are drawn directly from point to point, whereas non-hierarchical edges are drawn orthogonally and may have (orthogonal) bends. Each bend is determined by one point. Since we only attach these edges to the left or right side of the rectangle, the number of bends is always even. In our implementation we restrict the possible number of bends to two, hence it suffices to store the common abscissa.

1.2 Mutation

The following mutations are possible, each with its own stepsize and probability:

mutation	probability
move node	0.54
move port of an edge	0.1
flip port of an edge	0.21
move bend	0.1
delete bend	0.05

An operator is chosen with the given probability in an evolution step, the move nodes operator, e.g., with a probability of 0.54.

This operator works as follows:

1. Choose a random node.
2. Move position $(x', y') = (x, y) + (dx, dy)$
   where dx and dy are normally distributed random numbers with standard deviation $\sigma_x = 100$, and $\sigma_y = 50$, respectively.
3. Check for overcuts of edges with the node. If necessary flip port.
4. According to its iteration probability p_1 go to step 1.

The other operators work similarly, all iteration probabilities are initialized with 0.7.

1.3 Fitness Function

We have considered layout metrics for UML class diagrams in more detail in [Eichelberger 2005].

The fitness function or error measure is computed as a linear combination of 9 metrics, each scaled to the range $[0, 1]$.

1. **NN** Overlappings of nodes are counted and scaled regressively. It was not helpful to compute the overlapping area.

   $$\rho_{NN} := \frac{|\text{overlaps}|}{|\text{overlaps}|+1}$$

2. **EN** Overcuts of edges and nodes are treated analogously.

   $$\rho_{EN} := \frac{|\text{overcuts}|}{|\text{overcuts}|+1}$$

3. **EE** Edge crossings are counted, the current number is then divided by the upper bound of possible crossings similar to the formula suggested in [Purchase 2002]. That bound depends on the number of edges and the number of bends.

   $$\mathsf{eb} := |\text{edges}| + 2 \cdot |\text{bends}|$$
   $$\rho_{EE} := \frac{2 \cdot |\text{crossings}|}{\mathsf{eb} \cdot (\mathsf{eb}-1)}$$

4. **H** Hierarchical edges should follow a common flow direction, from bottom to top. In our simplified view of UML class diagrams hierarchical edges means generalizations. A linear scaling is sufficient.

*email:wolff@informatik.uni-wuerzburg.de
†email:eichelberger@sse.uni-hildesheim.de

$$\rho_H := \frac{|\text{direction misses}|}{|\text{generalizations}|}$$

5. **GL** Lengths of hierarchical edges can be preassigned, for instance depending on the size of the arrow head. This metric measures the difference to that prefered length. Shortening the length is punished harder than prolonging.

   $$\rho_{GL} := 1 - \frac{\sum_{g \in generalizations} \text{mix}(\bar{g}, l_{pref})}{|\text{generalizations}|}$$

   where $\text{mix} : \mathbb{R}_0^+ \times \mathbb{R}^+ \to [0,1]$, $\text{mix}(x,y) := \frac{\min(x,y)}{\max(x,y)}$ und $\bar{g}$ the length of g.

6. **AL** Associations, i.e. non-hierarchical edges are treated similarly. We, indeed, define a prefered length for the possibly 2 horizontal pieces, and compare the pieces and the full horizontal length separately. With $\mathsf{piece(a)} := \text{mix}(\bar{a}_{h1}, l_{pref}) + \text{mix}(\bar{a}_{h2}, l_{pref}) + \text{mix}(\bar{a}_{h1} + \bar{a}_{h2}, 2 \cdot l_{pref})$ we have $\rho_{AL} := 1 - \frac{\sum_{a \in associations} \mathsf{piece(a)}}{3 \cdot |\text{associations}|}$

7. **A** The angle between a hierarchical edge and the side where it connects to a node is estimated by the ratio of the horizontal distance to the sum of distances and linearly scaled.

   Let $\triangle x$ the horizontal and $\triangle y$ the vertical distance between start and end position of a hierarchical edge $(x,y) \to (x + \triangle x, y + \triangle y)$

   $$\rho_A := \frac{\sum_{generalizations} \left(\frac{\triangle x}{\triangle x + \triangle y}\right)^2}{|\text{generalizations}|}$$

8. **MP** If multiple edges connect to the same side of a node, a balanced distribution of the edge ports is aspired. Therefore the area spanned by squares between the ports is calculated, its minimum $\text{equi}(k) := n_k \cdot \left(\frac{\bar{k}}{n_k}\right)^2$ is achieved for an equidistant partition, its maximum $\bar{k}^2$ for $n_k = 0$ where k the side of a node partioned by n_k ports into $n_k + 1$ pieces $k_0, k_1, \ldots, k_{n_k}$ with lengths $\bar{k}_i, i = 0, \ldots, n_k$.

   The scaled error per side is accumulated: $\text{side}(k) := \frac{\sum_{i=0}^{n_k} (\bar{k}_i)^2 - \text{equi}(k)}{(\bar{k})^2 - \text{equi}(k)}$ $\rho_{MP} := \frac{\sum_{k \in nodesides} \text{side}(k)}{|\text{nodesides}|}$

9. **B** The number of bends should be minimal as suggested in many graph drawing publications or in [Purchase 2002]. $\rho_B := \frac{|\text{bends}|}{2 \cdot |\text{associations}|}$

Let x be a diagram with values $\rho_m(x)$ for the metric m. The fitness function is computed as a linear combination of these values.

$$F(x) := \sum_{m=1}^{9} \omega_m \cdot \rho_m$$

m	NN	EN	EE	H	GL	AL	A	MP	B
ω	1	1	5	10	2	1	1	0.5	0.5

2 Combination with SugiBib

The context of SugiBib[Eichelberger and von Gudenberg 2003a; Eichelberger 2005] is used to retrieve the structure and the semantics of the input, and to transform the information to a graph structure optimized for the evolutionary algorithm. Then the evolutionary algorithm is executed and finally the coordinates are handed to SugiBib. This allows us to use real coordinates for the evolutionary algorithm, and also enables the reuse of the SugiBib metrics.

3 Discussion

Overall, the approach was successful, we obtain acceptable layouts of UML class diagrams in reasonable time. We have developed a set of metrics that is capable of controlling the evolutionary algorithm. The overall time, however, is slower than SugiBib by a factor of about 20.

3.1 Fitness

Lots of experiments led us to the coefficients given in the paper.

It turned out that the hierarchy should be emphasized. It is much easier to fine-tune a diagram with some overlapping nodes than one with a hierarchy mismatch. Therefore the hierarchical metric got a high factor.

The regressive scaling of the overlapping metrics causes relatively high values (≥ 0.5), hence these metrics are very frequently reduced to 0.

With larger diagrams the weight for edge crossings should be increased.

We have noticed that the consideration of subtle metrics such as the angle A improves the overall layout of a diagram a lot. So does the MP metric that is responsible for a centered position of nodes in a hierarchy.

References

Eichelberger, H., and von Gudenberg, J. W. 2003. On the Visualization of Java Programs. In *Software Visualization, International Seminar, Dagstuhl Castle, Germany, May 2001, Revised Papers*, Springer-Verlag Inc., New York, NY, USA, S. Diehl, Ed., vol. 2269 of *Lecture Notes in Computer Science*, 295–306.

Eichelberger, H. 2005. Aesthetics and Automatic Layout of UML Class Diagrams Ph.D. Thesis Fakultät für Mathematik und Informatik, University of Würzburg

Purchase, H. C. 2002. Metrics for Graph Drawing Aesthetics. *Journal of Visual Languages and Computing 13*, 5, 501–516.

Purchase, H., Colpoys, L., Carrington, D., and McGill, M. 2003. UML Class Diagrams: An Empirical Study of Comprehension. In *Software Visualization - From Theory to Practice*, K. Zhang, Ed. Kluwer, 149–178.