New Automatic Layout Method based on Magnetic Spring Model for Object Diagrams of OMT

Takayoshi Noguchi
Doctoral Program in Engineering
University of Tsukuba
1-1-1, Tennodai, Tsukuba-shi
Ibaraki 305-8573,JAPAN
+81 298 53 5165
nogu@softlab.is.tsukuba.ac.jp

Jiro Tanaka
Institute of Information Sciences and Electronics
University of Tsukuba
1-1-1, Tennodai, Tsukuba-shi
Ibaraki 305-8573,JAPAN
+81 298 53 5343
jiro@softlab.is.tsukuba.ac.jp

ABSTRACT

We propose to apply Magnetic Spring Model for arranging object diagrams of OMT. The main characteristics of our automatic layout method are: efficient space utilization; arranging connected classes nearer to each other; and keeping the meaning of relationships in the diagram.

We carried out experiments to compare the new method and the existing ones. In our layout method, most of the nodes are arranged at the position where users expect to, and users' thinking time of our method is shorter than the existing methods. Our layout method is useful to draw object diagrams by the graphical editor of OMT CASE tools.

Keywords

graph layout, CASE tool, OMT

INTRODUCTION

Object-oriented software development by OMT(Object Modeling Technique)[1] has received much attention and a number of OMT CASE tools have been developed. Most of these CASE tools have graphical editors for drawing diagrams. In these editors, the layout of diagrams had to be arranged by users. Among the various OMT diagrams, the object diagram is the most important one. It is the base of software development. The object diagram is represented as shown in Figure 1, where nodes express classes and edges denote relationships. It can contain three kinds of relationships: association, inheritance and aggregation (Figure 2).

PROBLEMS OF NAKASHIMA'S ALGORITHM

Nakashima developed an automatic layout system for the object diagram of OMT[4]. He applied “graph drawing algorithm” to the automatic layout system. Nakashima gave attention to the tree structures of inheritance. He treated the tree structures of inheritance as subgraphs. Then he considered a metagraph composed of subgraphs (Figure 3). He applied Walker's algorithm[6] which is the tree layout algorithm to draw a subgraph and Eades' spring model[7] which is the undirected graph drawing algorithm to draw a metagraph.

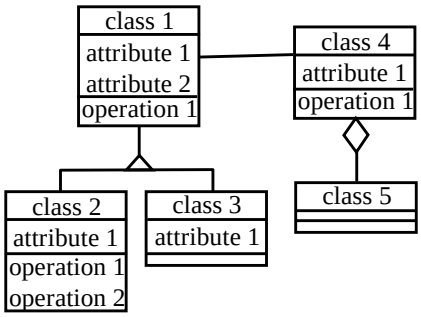

Figure 1: Object diagram

The advantage of Nakashima's algorithm is that inheritance relationship is shown clearly by using tree layout algorithm. The following problems are not considered.

- Aggregation, which must be drawn as a directed edge, was drawn as an undirected edge.
- Subgraphs were handled as rectangles in a metagraph. The area where nodes do not exist is treated as if occupied by nodes. As a result, the layout occupies too much space.
- When we arrange the layout of the metagraph, the location of nodes in subgraphs is not considered. Sometimes when there is a relationship between a node in a subgraph and a node outside of the subgraph, the distance between the two nodes becomes too long.

NEW LAYOUT ALGORITHM FOR THE OBJECT DIAGRAM

We concentrate on the directions of edges which express relationships. There are recommended directions for various relationships defined in the object diagram[1]. For example,

- The recommended directions for Association is from left to right.
- The superclass should be placed above the subclasses in Inheritance relationship. So the direction is top to down.

One algorithm for making edges the same direction is the directed graph algorithm[2][3]. It tries to array as many edges

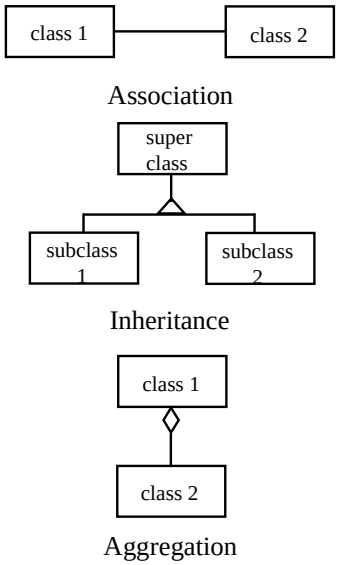

Figure 2: Graphical notation

as it can to the same direction. There is no algorithm for a graph in which many kinds of edges turn to each direction except Magnetic Spring Model[5]. We propose to apply Magnetic Spring Model[5] for the layout of the object diagram.

In Eades' spring model, we replace each edge with a spring. Repulsive forces(f_r) act upon every pair of non-neighboring nodes. Attractive or repulsive forces by the spring(f_s) act on neighboring nodes. We find the stable state of graph to arrange layout (Figure 4). Magnetic Spring Model added the rotative forces(f_m) to Eades' spring model. With this force, we control the direction of edges. The magnitudes of these three forces are given by:

$$f_s = c_s log(\frac{d}{d_0})$$

$$f_r = c_r \frac{1}{d^2}$$

$$f_m = c_m b d^{\alpha} |\theta|^{\beta}$$

α, β, c_s, c_r and c_d are the constants, d is the distance between nodes, d_0 is the natural length of the spring, b is the strength of a magnetic field, θ ($-\pi < \theta < \pi$) is the angle between the orientation of the magnetic field and the orientation of the magnetic spring.

To define the rotative forces, the notions of “magnetic fields” and “magnetic springs” are introduced (Figure 5). We replace the edges by magnetic springs and the graph is on the magnetic field. Edges turn to the north of magnetic field. The magnetic field is a virtual field, and it does not have the same nature as the real magnetic field. We can operate multiple magnetic fields together. Multiple kinds of edges are

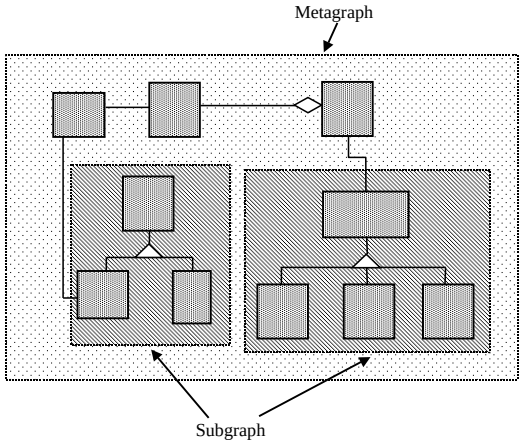

Figure 3: Hierarchical representation of object model

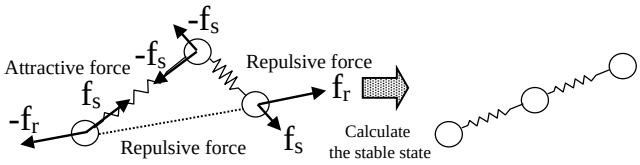

Figure 4: Layout with the spring model

arranged in each direction. There are two kinds of magnetic springs, uni-directional magnetic spring and bi-directional magnetic spring (Figure 6). In the uni-directional magnetic spring, the south of the magnetic spring turn to the north. In the bi-directional magnetic spring, the end which is nearer to the north of the magnetic field turn to the north.

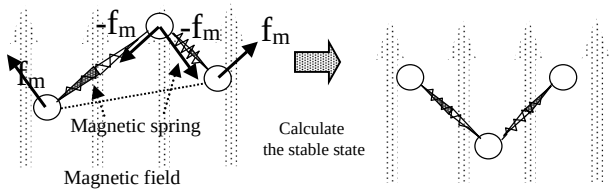

Figure 5: Layout with the magnetic-spring model

With these characteristics of Magnetic Spring Model, we can handle the directions of three kinds of relationships in the object diagram.

APPLICATION OF MAGNETIC SPRING MODEL TO OBJECT DIAGRAMS

We consider the following problems while using the Magnetic Spring Model.

- There are three kinds of relationships in the object diagram. We have to decide the kind of magnetic field for

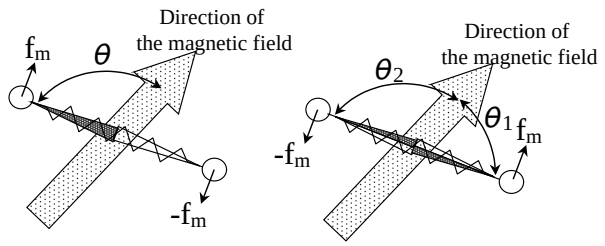

Figure 6: Uni-directional magnetic spring (left) and bi-directional magnetic spring (right).

each relationship.

- Magnetic Spring Model does not consider the size of nodes. The sizes of nodes are different in the object diagram.

The Magnetic Fields for each Relationship of the Object Diagram

There are three kinds of relationships in the object diagram. We use a compound magnetic field, which can use multiple magnetic fields. These magnetic fields do not influence each other. We decide the kind of magnetic field for each relationship based on the characteristics and meaning of the relationship.

- Association
  Association has no direction. The association between two classes can be read from either direction. There is no difference between the two classes. We adopt bi-directional magnetic spring for the association relationship. In the OMT notation, it is desirable that the nodes which are connected by the association relationship are read from left to right. we decide that the direction of the magnetic field is horizontal (Figure 7).
- Inheritance
  Inheritance is the relationship between the superclass and the subclasses. The subclasses specialize the superclass and inherit the attributes and operations of the superclass. The inheritance is expressed as a tree structure if there are multiple subclasses. We adopt uni-directional magnetic spring for the inheritance relationship. In OMT notation, superclass should be drawn on top of the subclasses. We decide that the direction of the magnetic field is in downward direction (Figure 7).
- Aggregation
  Aggregation is the relationship which is expressed as "whole-part". Some classes are parts of another class. The aggregation relationship is expressed as a directed edge from the "whole" class to its "part" classes. The aggregation has also the hierarchical structure whose depth is optional. We decide that the direction of the magnetic field be a diagonal from top-left to down-right. This is the intermediate angle between the association and the inheritance (Figure 7).

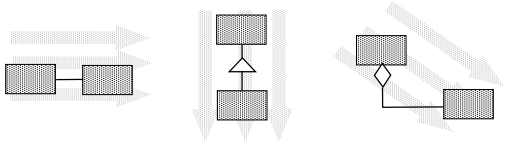

Figure 7: The magnetic field for the association(left), inheritance(center) and aggregation(right)

Size of Nodes for Magnetic Spring Model

In Magnetic Spring Model, nodes are operated as the infinitesimal points. If we use Magnetic Spring Model as it is, nodes may overlap in the object diagram. We improve Magnetic Spring Model to operate nodes whose sizes are optional. We let natural length of springs(d_0) and repulsive forces(c_r) change according to the following equations.

$$\begin{aligned}&\text{if } (d_1 + d_2 < D0)\\ &\quad \text{then}\\ &\qquad d_0(n_1, n_2) = D0;\\ &\quad \text{else}\\ &\qquad d_0(n_1, n_2) = d_1 + d_2;\end{aligned}$$

$$c_r(n_1, n_2) = (d_1 + d_2) * CR;$$

$D0$ is the shortest length of springs. CR is a constant. n_1 and n_2 are the names of nodes. d_1 and d_2 are the half length of nodes' diagonals. When we use these expressions, the length of the larger node is longer and the repulsive force of the larger node is stronger.

Result of Magnetic-Spring Model for Object Diagrams

Figures 8, 9 and 10 are the results of arranging layout with magnetic-spring model for object diagrams. Figures 8, 9 and 10 are parts of the object diagram of a window system[1].

Figure 8 looks like a tree structure at first. The `scroll_canvas` class makes multiple inheritance with the `canvas` class and the `scroll_window` class. The object diagram is not a tree structure. Nakashima's method can not arrange layout for this object diagram. Our method can arrange layout in which the superclass is drawn above the subclasses as shown in figure 8. Because instead of considering inheritance as a tree structure, our method decides the location of classes based on the direction of edges.

The `event` class has aggregation relationship with the `panel_item` class and the `text_item` class in Figure 9. In Nakashima's method, the inheritance tree is divided as subgraph and considered as rectangle in metagraph. Our method can draw `event` class near the `panel_item` class and the `text_item` class because of not using subgraphs.

Figure 10 is the layout of the object diagram which contains three kinds of relationships. All of the edges turn to the indicated directions. The distances between the connected nodes are not so long.

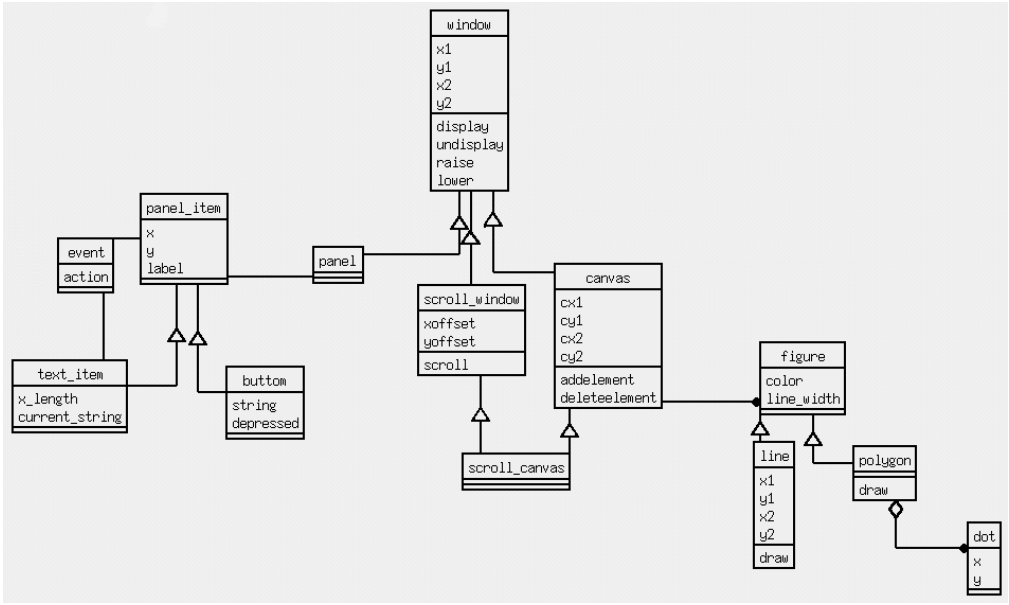

Figure 10: Result of layout 3

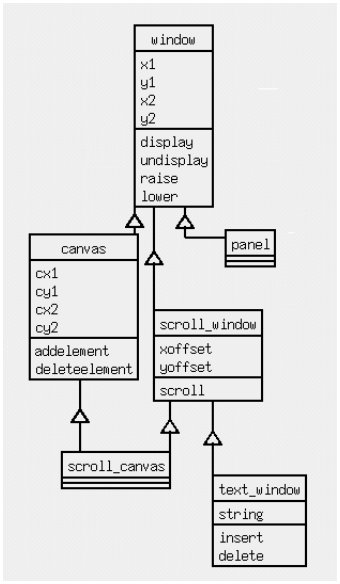

Figure 8: Result of layout 1

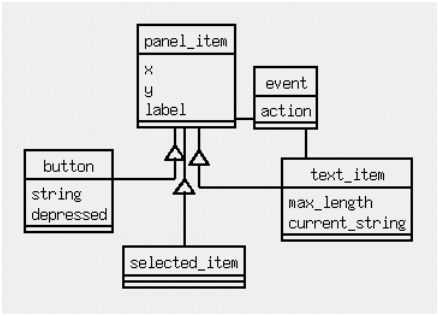

Figure 9: Result of layout 2

EVALUATION

We carried out an experiment to compare our method with the existing CASE tools. There are Rational Rose[9] which is a commercially available CASE tool and Nakashima's OMTEditor.

Experiment 1

We prepared the objects diagrams which are arranged by three layout methods: Rational Rose, Nakashima's method and our method. The resulted diagrams are then arranged by seven users in OMTEditor.

We measured the quantity of the work. If the quantity of the work is less, the layout of object diagrams is better. We gathered three kinds of data for the quantity of the work.

Movement distance The total distance of moving nodes.

Operation time The time take to amend the layout of an object diagram.

Movement frequency The total number of times of moving nodes.

We prepared the following two object diagrams for users.

The object diagram 1 The object diagram of the liquor warehouse[10]. (number of nodes: 9, number of edges: 12)

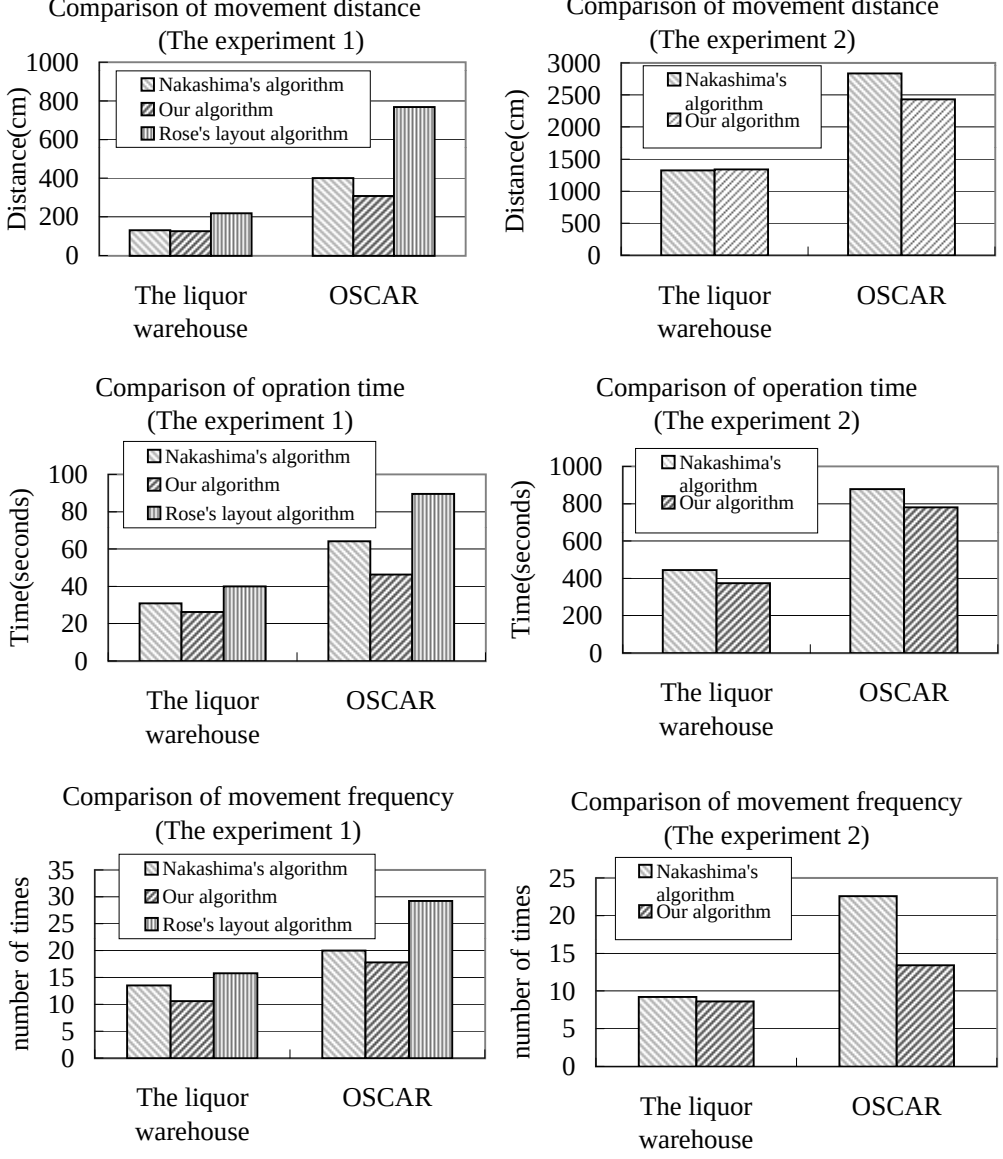

Figure 11: Result of the experiment

The object diagram 2 The object diagram of the animation system "OSCAR"[1]. (number of nodes: 14, number of edges: 14)

Experiment 2

Seven users drew the two diagrams from the beginning which were used in Experiment 1. They were allowed to use the layout method many times while drawing the diagrams. The experiment was carried out using our layout system and Nakashima's layout system. We gathered the same kinds of data as in Experiment 1.

Result of the Experiment

Figure 11 is the result of the experiment. Each value is the average of the values gathered by the seven users. In each value, minimum and maximum data are excluded.

In Experiment 1, our algorithm's results are better enough than the results of Rational Rose. Comparing our algorithm's result and Nakashima's algorithm's result, the results in the case of the liquor warehouse are almost the same. We think that the reason of these results is that the object diagram of the liquor warehouse has only association relationships. In relate to OSCAR, our algorithm took less time than Nakashima's algorithm, though the movement distance and the movement frequency are almost the same. We think that the reason of this result is that users' thinking time of our algorithm is shorter than Nakashima's one.

In Experiment 2, the results of the liquor warehouse are almost the same as Experiment 1. In the case of OSCAR which contains not only association relationships but also inheri-

tance and aggregation relationships, our algorithm is better than Nakashima's algorithm. Particularly our algorithm's movement frequency is fewer than Nakashima's one. We think in our algorithm most of the nodes are arranged at the position where users expect to.

CONCLUSION

We have proposed a new layout method using Magnetic Spring Model for object diagrams of OMT.

Our method can arrange the object diagram in which all relationships are implemented based on the characteristics and meaning of the relationship. Connected classes are arranged near to each other.

We have carried out experiments to compare the new method and the existing ones. From the results of the experiments, our method has better layout capability than the existing methods.

REFERENCES

1. James Rumbaugh et al: *Object-Oriented Modeling and Design* Prentice Hall, Inc (1992)
2. Giuseppe Di Battista and Roberto Tamassia: *Algorithms for Plane Representations of Acyclic Digraphs* Theoretical Computer Science, 66, pp.175-198 (1988)
3. Giuseppe Di Battista, Roberto Tamassia and Ioannis G. Tollis: *Area Requirement and Symmetry Display in Drawing Graphs* Proc. ACM Symp. on Computational Geometry, pp.51-60 (1989)
4. Satoshi Nakashima: *Automatic layout of diagrams by Object Modeling Technique* The Master's Thesis, Univ. of Tsukuba (1997), (*in Japanese*)
5. Kozo Sugiyama, Kazuo Misue: *Graph Drawing by Magnetic Spring Model* Journal of Visual Languages and Computing, Vol.6, no.3, pp.217-231 (1995)
6. John Q. Walker: *A node-positioning algorithm for general trees* Software Practice and Experience 20-7.pp685-705 (1990)
7. Peter Eades: *A Heuristics for Graph Drawing* Congressus Numerantium, Vol.42, pp.149-160 (1984)
8. Kozo Sugiyama: *Automatic graph drawing methods and their applications* The Society of Instrument and Control Engineers (1993), (*in Japanese*)
9. Rational Rose: http://www.rational.co.jp/products/rose/index.html, (*in Japanese*)
10. Toshiharu Yamazaki: The explanation of program planning technique number 2 IPSJ Magazine, Vol.25, no.11, pp.1219 (1984), (*in Japanese*)