Evaluating Visual Realism in Drawing Areas of Interest on UML Diagrams

Heorhiy Byelas and Alexandru Telea
Department of Mathematics and Computer Science
University of Groningen, Nijenborgh 9, 9747 AG Groningen, the Netherlands
h.v.byelas@rug.nl, a.c.telea@rug.nl

Abstract

Areas of interest (AOIs) are defined as an addition to UML diagrams: groups of elements of system architecture diagrams that share some common property. Some methods have been proposed to automatically draw AOIs on UML diagrams. However, it is not clear how users perceive the results of such methods as compared to human-drawn areas of interest. We present here a process of studying and improving the perceived quality of computer-drawn AOIs. We qualitatively evaluated how users perceive the quality of computer- and human-drawn AOIs, and used these results to improve an existing algorithm for drawing AOIs. Finally, we designed a quantitative comparison for AOI drawings and used it to show that our improved renderings are closer to human drawings than the original rendering algorithm results. The combined user evaluation, algorithmic improvements, and quantitative comparison support our claim of improving the perceived quality of AOIs rendered on UML diagrams.

1. Introduction

UML diagrams are used by system architects to describe and understand software architectures and designs, *e.g.* the structural and functional relations between the various interfaces, components, or roles [13]. Software elements that share a common property are of particular interest in system analysis, *e.g.* "all high-reliability components" or "all components introduced in the system version 2.3". We call such a set of elements an *area of interest* (AOI). AOIs can be defined using software metrics [8, 11] computed by existing analysis tools [30].

Recently, a method for visually combining AOIs and architecture (UML) diagrams [6] has been proposed. This method shows AOIs in-place on the diagrams, enabling users to correlate concerns (AOIs) with system structure (diagrams), without altering the diagram layout. The AOIs are rendered as soft, fuzzy shapes surrounding the diagram elements, by a combination of geometric and texture-based techniques (see *e.g.* Figure 9 b). Roughly, the rendering tries to imitate the way humans draw AOIs in practice using pen and paper. The rendering method results were found quite appealing by potential users [1].

However, for this method to be effective in practice, some questions still remain to be answered: Do actual users like computer-drawn AOIs comparably to hand-drawn AOIs? If not, why, and how can we improve the computer-drawn AOIs? We present here our quest to measure and improve the perceived quality of computer-drawn AOIs. We designed and executed a detailed empirical evaluation for the AOI drawings quality, and distilled salient strengths and weaknesses of the original AOI rendering [6] and of hand-drawn areas. Hand-drawn areas were found easier to understand than computer-drawn ones. Also, we found two main drawbacks of computer-drawn areas: incorrect exclusion of overlapping elements and unnatural flow-of-hand. We extended the original AOI algorithm to address these limitations. Finally, we designed a metric to compare AOI renderings, and showed that the results of our improved algorithm are closer to human drawings than the results of the original rendering algorithm.

This paper is structured as follows. Section 2 reviews related work in visualizing AOIs on diagrams and evaluating quality aspects of diagram drawings. Section 3 overviews the original AOI algorithm [6]. Section 4 details the algorithmic limitations of the original method. Section 5 presents the empirical evaluation used to compare the quality of computer and human drawings. Section 6 presents our new technique that improves the rendering of AOIs on diagrams. Section 7 presents a quantitative comparison of the human and computer drawings. Section 8 presents the results of our evaluation and proposed algorithmic improvements. Finally, Section 9 concludes the paper.

[1] The proposed method has obtained the "best paper award" at ACM SoftVis'07 [6]

2 Related Work

Visualizing areas of interest can be described with the 5-dimensional model of Marcus *et al.* [16]: task, audience, target, medium, representation. Our task is to understand how various system aspects (the AOIs) map on a system description (the UML diagram). Our audience is composed of software architects. Our visualization target is a set of diagrams, together with AOIs specified as sets of diagram elements. The visualization medium is a modified UML diagram viewer [28] that combines rendering diagrams and AOIs. The representation enriches classical UML diagrams with AOIs drawn as smooth soft-textured shapes.

Modeling tools, *e.g.* Rational Rose [13] or Together [5], are standard ways to visualize UML diagrams, but have little support for areas of interest beyond boxes. Drawing AOIs as boxes without changing the base diagram layout yields unacceptably high visual clutter. Tools such as Rigi [29], Prefuse [17], or MetricView [28] show an AOI by marking its elements with icons scaled, colored, and shaped to show metric values. Yet, inferring AOIs from such markers is hard for complex diagrams. One can also move all elements in an AOI close to each other and next draw a surrounding frame [9]. However, diagrams are often laid out manually with great care. Changing the layout upon changing of the AOIs destroys the user's 'mental map'. Also, layout adaption does not work when one needs to show several AOIs at the same time.

Methods such as metaballs [22], H-BLOB [25] and 2D implicit surfaces [2] compute AOIs as isosurfaces of some potential function, or distance field, based on the elements' locations. However, it is hard to control both the smoothness and tightness of such isosurfaces. Worse, the isosurface connectivity highly depends on a correct isovalue which cannot be easily chosen automatically [25]. Finally, distance fields and isosurfaces are computationally expensive.

A recent technique was proposed to draw AOIs by shrinking and smoothing the convex hull of the enclosed elements [6]. The features targeted by this technique are:

1. AOIs do not change a given diagram layout
2. AOIs drawing is real-time, even for large diagrams
3. AOIs correctly surround the enclosed elements
4. AOIs do not clutter the diagram or each other
5. Computer- and human-drawn AOIs should resemble

The first two requirements were satisfied by the proposed technique by construction. The technique did not correctly handle requirement 3, as we shall see in Sec. 4.1. Also, it was not measured how precisely requirements 4 and 5 are satisfied.

A separate body of related work concerns evaluating the quality of a visual depiction of system (UML) architectures. Since there are no specific studies to evaluate the quality of AOI renderings, we consider the wider range of evaluating the quality of UML diagram renderings. Purchase *et al.* have assessed the comprehensibility, aesthetics, and user preferences of UML (and similar) diagram renderings [19, 18, 20, 21]. Although valuable, such results cannot be applied directly to our problem, since AOIs are an extension of the standard UML notation. Several authors propose frameworks and methodologies to evaluate the comprehensibility and overall quality of UML models [14, 4, 1, 23]. Still, the question "what are good quality criteria for visual modeling languages" is not exhaustively answered.

An emerging conclusion is that 'bare' UML is often hard to comprehend and can perform better if extended by task-specific annotations. Our AOIs are precisely such an annotation, useful to show cross-cutting concerns atop of a given system structure. We want to construct a computer algorithm that renders AOIs similarly to good hand-drawn AOIs. The above-mentioned requirements of the technique in [6] attempt to capture the *in abstracto* quality criteria of a good AOI drawing. Yet, to assess the *perceived* quality of an AOI drawing, we need a specific study. In this paper, we present such a study and an improved AOI-rendering algorithm based on the study.

3 AOI construction

We briefly overview the two-stage construction of the AOIs from [6]. First, a contour is built around the elements (Fig. 1 b-d). Given the 2D bounding boxes $(b_{1i}, b_{2i}, b_{3i}, b_{4i})$ of the elements e_i in the AOI (Fig. 1 a), we first compute the convex hull $C = \{q_i\}$ of the corners $\{b_{ij}\}$ (Figure 1 b). The hull is next subsampled (Fig. 1 c) so that the distance $\delta = |q_i - q_{i+1}|$ between consecutive points is a given small fraction of the hull perimeter $|C| = \sum_i |q_i - q_{i+1}|$, *e.g.* $\delta = 0.01|C|$. Next, the subsampled contour $\{q_i\}$ is deformed to fit tightly the elements inside and to become smoother. (Fig. 1 d). For this, every point q_i is moved to q_i' along the normal $\vec{n}$ to the segment $(q_{i-1}q_{i+1})$, i.e. $q_i' = q_i +$

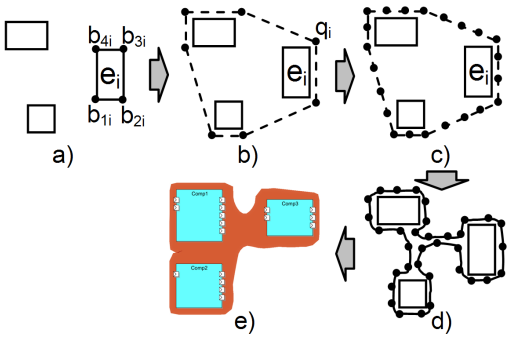

Figure 1. Area of interest algorithm

$\varepsilon_n \vec{n} + \varepsilon_s \frac{q_{i-1}+q_{i+1}}{2}$. Here, ε_n and ε_s are parameters that control the shrinking and smoothing strengths respectively. Good values are $\varepsilon_n = 0.005|C| = 0.5\delta$ and $\varepsilon_s = |q_{i-1} - q_{i+1}|/4$. If $\{q_i\}$ are in counterclockwise order, q_i moves inwards in C. This shrinks the contour, but also moves q_i towards the middle of the segment $(q_{i-1}q_{i+1})$, which is the well-known geometric Laplacian smoothing [26]. To prevent contour self-intersection, a point q_i is moved only if

$$d = \min\left(\min_{|j-i|>1} |q_i - q_j|, \min_j |q_i - p_j|\right) > 2\delta \qquad (1)$$

i.e. if the contour point q_i is farther from all corners p_j and other contour points q_j (except its immediate neighbors q_{j-1}, q_{j+1}) than 2δ. To prevent the contour to become over- or undersampled, we check the distances $\min_{|j-i|>1} |q_i - q_j|$ between the moved point q_i and its neighbors, and insert or remove points if these fall outside the range $[0.5\delta, 2\delta]$. Points are moved for N_{max} iterations.

In the second stage, we draw AOIs in two steps using the computed contours. To get filled areas, we triangulate the contour and draw it in the area's color, else skip this step. Next, we draw a soft border along the contour using a half-transparent texture brush (see [6] for details).

4 Technical Limitations

The original AOI rendering algorithm (Sec. 3) has several limitations. We discuss these next.

4.1 Ineffective exclusion

The contour constructed as described in Sec. 3 may erroneously overlap, or *include*, elements which are logically not in the AOI. In the original method such elements are marked with an *eraser*, *i.e.* a thin white border. Consider Figure 2 where elements A, B, C, and D are in the area and E, F are outside. The eraser works reasonably well if we draw *filled* areas and the overlapping elements are *completely* inside the area, *e.g.* E in Fig. 2 a. However, the eraser cue may not be salient enough to easily see that E is not in the area. For elements partially overlapping the area from which they must be excluded, *e.g.* F in Fig. 2 a, the cue is even weaker. For contour-drawn areas, the eraser works very weakly (F in Fig. 2 b) or not at all (E in Fig. 2 b). This is a serious limitation, since we saw that contour drawing is preferred to filled drawing [6]. Also, the eraser technique turned out to be very unnatural for the most users who were shown it during our evaluation (Sec. 5).

4.2 Unnatural flow-of-hand

Figure 2 shows also a second problem of the original AOI rendering technique. Close to the elements, contours are too tight. In the middle, they are too loose. Also, the contour smoothness is not optimal. The contour looks like a sharp-angled polyline. This image is clearly different from user-drawn contours, which are much smoother (see *e.g.* Fig. 9). The non-uniform tightness and angled contours create a computer-made, unnatural look, quite different from the *flow-of-hand* typical to human drawings.

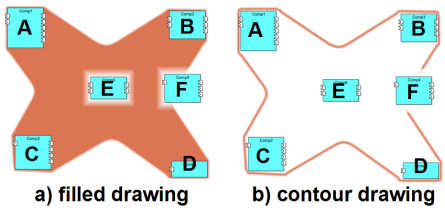

Figure 2. Element E is incorrectly shown as being inside on (b).

5 Evaluation of the AOI rendering method

At this point, several questions were raised. How bad would real users find the limitations of the AOI drawing technique (Sec. 4) in practice? Do other limitations exist? And above all, what are the qualities of an AOI drawing that users like the most, and how to simulate these in a computer rendering? We must answer these questions to estimate and improve the acceptance and usability of the AOI technique.

For this, we designed and executed an empirical evaluation. Thirty users of master, PhD, and senior software designer level were selected. All had worked before for at least a few months (up to a few years) with UML diagrams in software design activities. Our evaluation had three stages: drawing production, drawing comparison and results evaluation.

5.1 Drawing Production

First, the participants were given a complex class diagram with 110 classes marked by numbers, printed in black-and-white on an A4 paper and seven AOIs, each given as a list of class numbers, printed on a separate paper. The subjects were next asked to draw the areas as contours on this diagram, with a provided red marker pen. The subjects were told that the drawing should accurately convey which class is in which area(s) and which area contains which classes. An example drawing, done on a much smaller UML diagram with 10 classes and one AOI, was provided for illustration purposes. The experiment instructions were provided on a separate A4 sheet. No verbal indications or supervision were given during the actual work, which lasted approximately 15 minutes. The subjects worked fully independently. Figure 9 a shows a scan of the drawing done by one of the subjects.

Without the subjects' knowledge, we produced a computer drawing on the same UML diagram, using the rendering method in Sec. 3. The result is shown in Fig. 9 b.

5.2 Drawing Comparison

In the second phase, we gave to each subject two drawings: a randomly picked drawing of another subject and our computer-rendered drawing. Without giving any hint that one drawing was computer-made, we asked the subjects to complete a questionnaire (see [27]). The questions included:

1. rank the ease of understanding of the areas in each drawing on a scale of 1 (hardest) to 5 (easiest), accordingly to a Likert scale [15]
2. which is the most complex area to understand
3. list what you liked least/most in the given drawings

The questionnaire mentioned that the main quality of an AOI drawing is its understandability: the drawings should clearly show which area contains which classes.

5.3 Results Evaluation

In the third phase, we analyzed the questionnaire data. After collecting the questionnaires, we had some short discussions (10-15 minutes) with the subjects, where they freely presented their impressions, and we recorded the observations in writing. The results of this phase are summarized in Fig. 3.

#	Better drawing (H or M)	Human quality (1-5)	Machine quality (1-5)	Diff	Most complex area	Confusing overlaps	Exclusion unclear	Nonuniform tightness	Lacking flow-of-hand
1	H	5	3	2	-	+			+
2	H	4	3	2	3		+		
3	H	4	2	1	-			+	+
4	H	3	3	2	3	+	+		
5	H	4	3	1	2		+	+	+
6	M	3	4	1	3	+	+		
7	H	3	2	2	2	+	+		+
8	H	3	2	1	3		+	+	+
9	H	5	3	2	2	+			
10	H	4	3	2	3	+	+		
11	M	3	4	2	2	+	+		
12	H	4	3	2	3		+	+	
13	H	5	4	2	2		+		+
14	H	5	2	1	2	+	+	+	
15	H	5	2	2	3	+	+		
16	H	5	3	2	2	+			
17	H	5	2	1	2	+			+
18	H	4	3	2	3	+			+
19	H	5	2	1	2	+	+		+
20	H	5	4	2	-	+			+
21	H	4	2	3	5	+			
22	H	4	3	1	-	+	+		+
23	H	4	2	2	3		+	+	
24	H	4	4	3	3	+		+	+
25	H	4	2	2	2	+	+		
26	H	4	2	1	3	+	+		+
27	H	5	1	1	7	+	+		+
28	H	4	3	2	3	+	+		
29	H	3	2	2	7	+			+
30	H	4	3	2	3	+			+
31	H	5	4	2	2		+		+
	A			B	C	D	E	F	G

Figure 3. Results of drawing comparison

Several points are apparent now. The machine-generated drawing (M) was of comparable, but still lower, understandability, than the human-made ones (H) (Fig. 3, column A). The perceived quality difference was quite similar for all users, *i.e.* 1 or 2 points out of 5 (Fig. 3, column B). The hardest-to-grasp areas were quite consistent, *i.e.* areas 2 and 3 (Fig. 3, column C). This matches also our opinion, and indicates that the drawings done by different users are of comparable understandability. Two drawbacks of the machine-drawn areas were most frequently named. Firstly, the eraser, used to mark elements overlapping an AOI contour but not logically part of that AOI, is not working well, as we indeed suspected beforehand (Sec. 4). We call this the *wrong exclusion* problem. For example, class 56 in not part of the drawn area, as wrongly suggested by the computer drawing in Fig. 4 a, which shows a zoomed-in detail from Fig. 9. Figure 4 b, done by a human, is however correct. This problem was found by most subjects, as shown in column E of the table.

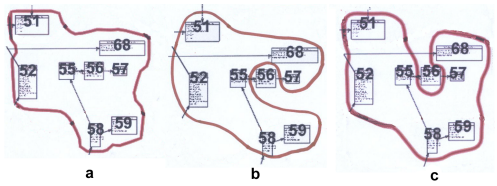

Figure 4. The initial computer rendering shows element 56 incorrectly inside the area (a), as opposed to the human drawing (b) and improved algorithm(c)

The second drawback of the computer-drawn areas concerns the contours' *tightness and smoothness*. These were perceived as being unpleasantly non-uniform (column F), and the *flow of hand*, *i.e.* similarity to the way humans draw, was lacking (column G). All users mentioned these as hindering the drawings' understandability. Finally, many subjects found the computer-drawn area *overlaps* confusing (column D). Contours which are near-tangent close to their crossing points were named hard to understand (Sec. 5.2).

6 Algorithm improvements

In Section 5.3 we identified three main problems of computer-drawn AOIs: wrong exclusion, non-uniform tightness/smoothness and confusing overlaps. We present next algorithmic improvements to address the first two.

6.1 Improved Exclusion

In previous work [6], the route of geometrically eliminating overlapping elements by modifying the contour is not followed, as it is deemed too complex to do for general diagrams. However, we found a working solution in this direction. Our idea is to edit the contour, before deforming it,

in order to exclude the wrongly overlapping elements. This works as follows (see also Fig. 5).

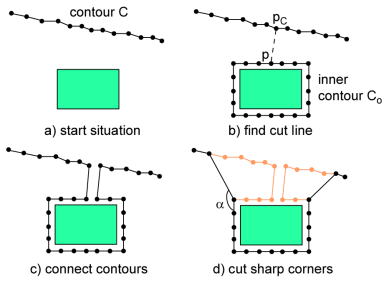

Figure 5. Geometric-based exclusion steps

First, we find the overlapping elements $O = \{o_i\}$ by testing if any of the four element corners falls within the already computed AOI convex hull C (Section 3), using a simple and robust point-in-convex-polygon test. Next, each element o_i in O is excluded in turn, as follows. A finely-sampled rectangular contour C_o is constructed around the bounding box of o_i (Fig. 5 b). Next, a short *cut line* connecting C_o with the original contour C is computed such that it does not intersect any of the elements e_i in C, as follows. We find first the closest two points $p_o \in O$ and $p \in C$. Next, we move both p_o and p_C along the inner and outer contours O and C respectively, until the line does not intersect any element. We start by moving p_o around O to the left (counterclockwise sweep) until a non-intersecting line is found or a too high distance d_{max} from the starting position, as computed along O, is reached. If no line can be drawn, we try now moving p_o to the right (clockwise sweep). If this fails too, then we move the other point p one step along the outer contour C, and repeat the inner contour sweep again. When a cut line was found (dotted line in Fig. 5b), we connect the inner and outer contours by two sampled line-segments, close and parallel to the cut line (Fig. 5c).

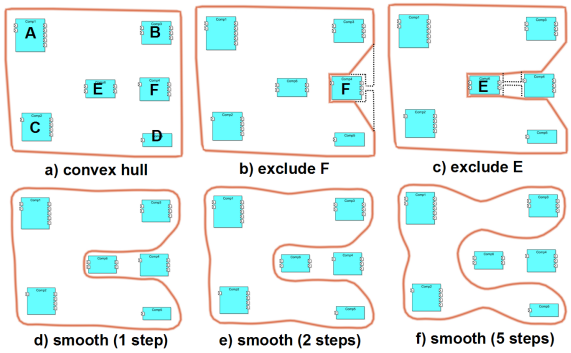

Figure 6. Exclusion of all overlapping elements (E, then F) and cutting sharp corners. After that, smoothing and shrinking

6.2 Natural flow-of-hand

Our subjects found contours sharp corners to be unnatural. We reduce these as follows. For each contour point p_i, we compute the angle $\alpha = \widehat{p_{i-1}p_ip_{i+1}}$ made by that point with its two neighbors. If α is under a minimal value α_{min} and the line $p_{i-1}p_{i+1}$ does not intersect any element e_i, then we remove p_i from the contour by connecting p_{i-1} and p_{i+1}. Good values for α_{min} are in the range $[40,70]$ degrees. We repeat this removal iteratively until no longer possible. The final result is shown in Fig. 5 d. When excluding several overlapping elements o_i, sharp corners are removed after excluding each element o_i. This gives better quality, as unnecessary sharp corners are eliminated as soon as possible. Finally, the shrinking is more stable if sharp corners are eliminated, a well-known fact from level set theory [24].

Figure 6 shows the same diagram as in Fig. 2. We see how the elements F and E are iteratively removed (Fig. 6 b,c). The red line shows the contour after exclusion and sharp corner removal. The dotted black line shows the contour after exclusion but before sharp corner removal. Figures 6 d-f show the result after a few smoothing steps. Clearly, these results are better than the original ones (Fig. 2 b). We see now very clearly what is inside, and what outside, an area. The unnatural eraser effect is now gone, and the contour resembles much more to what a human would draw. The sharp corner-cutting has the extra positive effect of smoothing the contour, yielding a more natural 'flow of hand'. We see that the improved method (Fig. 4 c) is more similar to a hand-drawing(Fig. 4 b) than the original method (Fig. 4 a).

7 Quantitative Analysis

We presented in Sec. 6 several algorithmic improvements by which we hoped to address the shortcomings of our computer rendering method. However, how to measure how well we improved as compared to the original algorithm? Repeating the user study (Sec. 5) with the *same* audience could be biased, since the users by now knew our aims, datasets, and already had some experience. Doing the same study with different subjects and/or different datasets could be done, but how to quantitatively compare subjective qualitative opinions of two different groups and/or datasets? Also, a user experiment does not precisely quantify how much closer or further our new algorithm improves the rendering. We designed a quantitative analysis for this, as follows (see also Fig. 7).

First, we extracted the area contours from all drawings, *i.e.* human and computed-generated with both the original and improved algorithm, using a simple filter-by-color thresholding technique, which was reliable as contours and diagrams were drawn with two distinct colors, *i.e.* red, respec-

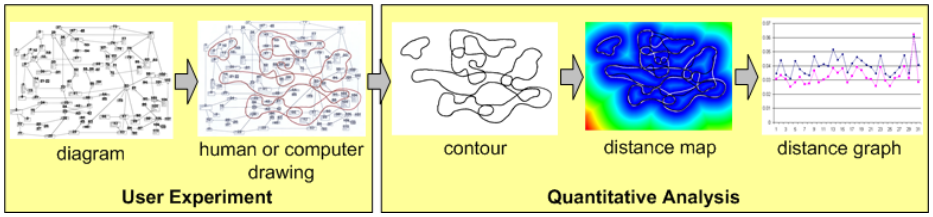

Figure 7. Quantitative analysis process

tively black. Next, we measured the difference between any two contours C_i and C_j, *i.e.* human and/or computer-drawn, as follows. For a contour C, we denote by D the distance transform, or *distance map* of C:

$$D(p) = \min_{q \in C} |p - q|, \quad \forall p \in \mathbb{R}^2 \tag{2}$$

$D(p)$ gives the distance from any point p to the closest point q on C. We compute D by solving the so-called Eikonal equation $|\nabla D| = 1$ with boundary condition $D = 0$ on all points of C, using the Fast Marching Method [24], on the pixel grid on which the scanned contour C is stored. Given a contour C_i and its distance map D_i, we define the distance d_{ij} of C_i to another contour C_j as:

$$d_{ij} = \frac{1}{2}\left(\frac{\sum_{p \in C_j} D_i(p)}{|C_j| D_{i_{max}}} + \frac{\sum_{p \in C_i} D_j(p)}{|C_i| D_{j_{max}}}\right) \tag{3}$$

Here, D_j denotes the distance map of C_j, while $D_{i_{max}}$ and $D_{j_{max}}$ are the maximum values of D_i and D_j respectively over the considered images. $|C_i|$ and $|C_j|$ denote the contour lengths in pixels. The definition of d_{ij} ensures d is a symmetric function $d_{ij} = d_{ji}$, normalized between 0 and 1. Intuitively, Eqn. 3 states that the distance between the two contours C and C' is proportional with the area between the two contours, which is a perceptually good measure. Alignment and image registration problems are not an issue, since all drawings are done on the precisely the same class diagrams, rendered on identical canvases, scanned at the same resolutions. Let us stress that other more sophisticated measures, *e.g.* perceptual-based metrics [3], template-based matching [10], or contour matching using the earth mover's distance [12], can be used as well. However, using more complex distance metrics involves having a clearer idea of which features (*e.g.* angles, protrusions, concavities, flat regions) are perceptually more important for the match, an information which we do not have at the present moment.

We build now a matrix d_{ij} containing all distances between any two contours of the 31 hand-drawn ones, plus the two computer-drawn ones with the original, respectively improved, methods. As the user evaluation showed (Fig. 3), not all hand drawings were found to be of the same quality. We are actually interested to see how our computer-drawn contours compare to good human drawings. For this, we first split the 31 human drawings into three groups: good, average, poor, based the "human quality" scores of 5,4 and 3 respectively (Fig. 3). Next, we graph the distances between all 31 human drawings and the two (initial and improved) computer drawings (Fig. 8). Several observations can be made here. First, there is quite some distance variation within the same quality class. This is expected, since each quality value was assigned subjectively by just one person. Second, we see that the improved method brings the computer-generated drawings closer to the human drawings in all quality classes as compared to the original method.

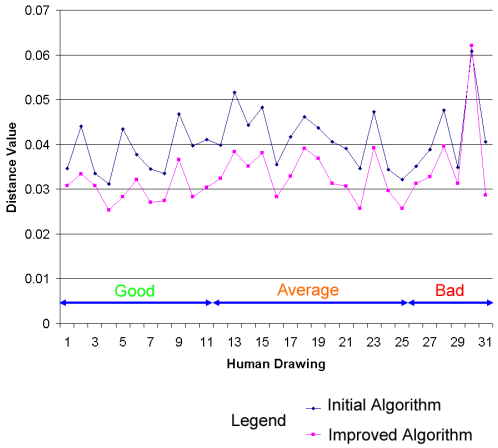

Figure 8. The distances between all 31 human drawings and two (initial and improved) computer drawings

8 Discussion

Let us consider three AOI drawings on the diagram used in our evaluation (Fig. 9): The top image is an actual scan of one of the best, most understandable, human drawings. The middle image shows the result of the original computer drawing method [6], as shown to the subjects. We here recognize all problems named so far: elements incorrectly included in surrounding areas whereas they should be outside (A,B,C), as shown in the top drawing; and contours which are tight and sharp close to the elements but loose and smooth in the middle. The bottom drawing shows

our improved algorithm. The elements A,B,C are now correctly excluded. The contours have a more uniform smoothness. Although the improved method yields better appreciated drawings, which are measurably closer to human drawings than the original method, it still has some limitations. First, our users have found near-tangently intersecting contours to be hard to understand (Sec. 5.3). To address this, we should consider a global contour rendering rather than the per-contour rendering. Second, the distance metric used to compare contours is well-known in shape analysis applications (see *e.g.* [7]), is robust to small-scale geometric noise, and rotation and scale invariant. However, it does not take into account specific quality attributes for the tasks related to AOIs. For example, we can argue that a small geometric difference between two contours is perceptually more important if located at some point where several contours overlap or intersect, than at the periphery of the drawing. Integrating perceptually driven distance metrics [3, 10] in our evaluation should lead to further insights. Finally, we are aware that we have not conducted a formal user experiment, *i.e.* a quantitative measurement of the (in)validation of a hypothesis. Our main goal was to harvest information about the differences perceived between computer- and human-drawn AOIs, and to adapt our computer drawings accordingly. If our hypothesis that human-drawn AOIs are easy to understand holds, we argue that our improved algorithm produces better drawings, since these are measurably closer to human drawings than the original computer drawings.

9 Conclusions

We have presented an attempt to render areas of interest (AOIs) on UML diagrams so they resemble easy-to-understand human drawings. First, we compared an existing AOI rendering method [6] with human-drawn results by. We found some aesthetic and correctness limitations of the existing method, as well as what users consider to be a good drawing. The method in [6] scored averagely in aesthetics and overall quality. Next, we designed several rendering improvements. The geometric exclusion technique creates contours which correctly and visibly exclude all elements. The corner-cutting technique, which eliminates sharp corners, creates a smooth, evenly-fitting, natural flow-of-hand drawing style. Finally, we quantitatively measure how close computer-generated renderings are to human drawings, and saw that our proposed improvements effectively brought the computer-made drawings closer to typical (good) human drawings.

Our work is one of the first attempts to measure empirically and quantitatively the quality of UML diagram annotations, such as areas of interest, and to use these measurements to improve computer-generated annotations. We found that user studies, although highly time-consuming and laborious, are indispensable instruments to get deeper understanding of what makes a visualization good. There was strong consensus among diverse users, who did not communicate with each other, about what is nice and less nice in the AOI drawings. Designing the proposed rendering improvements followed naturally, once we understood what the users liked to see. Hence, we believe that we succeeded to address here requirements 3 (correctness) and 5 (measured similarity with human drawn areas) from the desirable AOI drawing requirements et (Sec. 2). Requirements 1 and 2 were already addressed by the original rendering method.

We next consider to design a rendering method that mimics even closer (good) human drawings, by performing a machine-learning process that optimizes the proposed distance metric by varying the computer method's rendering parameters. We also want to address requirement 4 (limited cluttering) on complex diagrams, by a combination of geometric, shading, and texture techniques.

References

[1] J. Aranda, N. Ernst, J. Horkoff, and S. Easterbrook. A framework for empirical evaluation of model comprehensibility. In *Proc. Intl. Workshop on Modeling in Software Engineering (MiSE)*, pages 7–15, 2007.

[2] M. Balzer and O. Deussen. Exploring relations within software systems using treemap enhanced hierarchical graphs. In *Proc. VISSOFT*, pages 89–94. IEEE Press, 2005.

[3] S. Berretti, A. D. Bimbo, and P. Pala. Retrieval by shape similarity with perceptual distance and effective indexing. *IEEE Transactions of Multimedia*, 2(4):225–239, 2000.

[4] A. Bobrowska. A framework for empirical evaluation of model comprehensibility. In *Proc. SOFSEM*, pages 72–81, 2005.

[5] Borland. Together. 2005. `www.borland.com/together`.

[6] H. Byelas and A. Telea. Visualization of areas of interest on software architecture diagrams. In *Proc. ACM SoftVis*, pages 105–114, 2006.

[7] L. F. Costa and R. M. Cesar. *Shape Analysis and Classification: Theory and Practice*. CRC Press, 2001.

[8] N. Fenton and S. Pfleeger. *Software Metrics: A Rigorous and Pracical Approach*. Chapman & Hall, 1998.

[9] E. Gansner and S. North. An open graph visualization system and its applications to software engineering. *Software: Practice & Experience*, 30(11):1203–1233, 2000.

[10] D. M. Gavrila. Multi-feature hierarchical template matching using distance transforms. In *Proc. ICPR*, pages 439–444, 1998.

[11] N. Gill and P. Grover. Component-based measurement: A few useful guidelines. *ACM SIGSOFT Software Engineering Notes*, 28, 2003.

[12] K. Gtauman and T. Darrell. Fast contour matching using approximate earth mover's distance. In *Proc. CVPR*, pages 220–227, 2004.

[13] IBM. *Rational Rose*. 2005. `www.306.ibm.com/software/rational`.

[14] J. Krogstie. Evaluating uml using a generic quality framework. In *UML and the unified process*, pages 1–22. Idea Group Inc., 2003.

[15] R. A. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 140, 1932.

[16] A. Marcus, L. Fend, and J. I. Maletic. 3d representations for software visualization. In *Proc. ACM SoftVis*, pages 27–36, 2003.

[17] Prefuse. The prefuse information visualization system. 2007. `http://prefuse.org`.

[18] H. C. Purchase, D. A. Carrington, and J.-A. Allder. Empirical evaluation of aesthetics-based graph layout. *Empirical Software Engineering*, 7(3):233–255.

[19] H. C. Purchase, D. A. Carrington, and J.-A. Allder. Graph layout aesthetics in uml diagrams: User preferences. *Journal of Graph Algorithms and Applications*, 6(3):255–279.

[20] H. C. Purchase, L. Colpoys, and D. A. Carrington. Uml collaboration diagram syntax: an empirical study of comprehension. In *Proc. VISSOFT*, pages 13–22, 2002.

[21] H. C. Purchase, R. Welland, M. McGill, and L. Colpoys. Comprehension of diagram syntax: an empirical study of entity relationship diagram notations. *International Journal of Human-Computer Studies*, 61.

[22] J. Rilling and S. P. Mudur. On the use of metaballs to visually map code structures and analysis results onto 3d space. In *Proc. WCRE*, pages 299–306. IEEE Press, 2002.

[23] J. Rumbaugh. Notation notes: Principles for choosing notation. *Journal of Object-Oriented Programming*, 12(4), 1999.

[24] J. Sethian. *Level set methods and fast marching methods*. Cambridge Univ. Press, 1999.

[25] T. Sprenger, R. Brunella, and M. Gross. H-blob: A hierarchical clustering method using implicit surfaces. In *Proc. Visualization*, pages 61–68. IEEE Press, 2000.

[26] G. Taubin. Geometric signal processing on polygonal meshes. In *EUROGRAPHICS STAR Reports*, 2000.

[27] A. Telea. *AOI* user study. 2007. `www.win.tue.nl/~alext/ARCHIVIEW/experiment.html`.

[28] M. Termeer, C. Lange, A. Telea, and M. Chaudron. Visual exploration of combined architectural and metric information. In *Proc. VISSOFT*, pages 21–26. IEEE Press, 2005.

[29] S. Tilley, K. Wong, M. Storey, and H. Müller. Programmable reverse engineering. *Intl. J. Software Engineering and Knowledge Engineering*, 4(4):501–520, 1994.

[30] J. Wust. *SDMetrics*: The software design metrics tool for *UML*. 2005. `www.sdmetrics.com`.

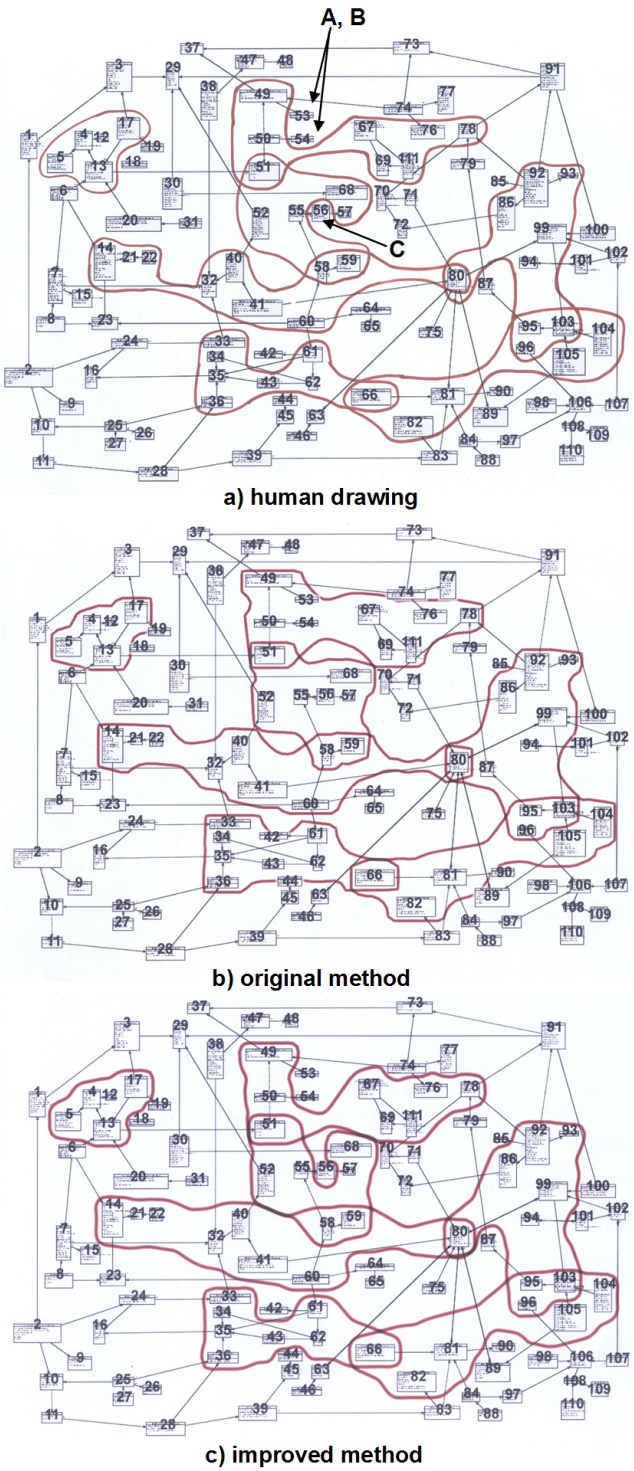

Figure 9. Comparison of AOI renderings. The improved rendering method (c) yields results closer to the human-drawn AOIs (a) than the original method (b)