Novel Visual Representations for Software Metrics Using 3D and Animation

Andreas Kerren and Ilir Jusufi
Växjö University
School of Mathematics and Systems Engineering (MSI)
Vejdes Plats 7, SE-351 95 Växjö, Sweden
{andreas.kerren | ilir.jusufi}@vxu.se

Abstract: The visualization of software metrics is an important step towards a better understanding of the software product to be developed. Software metrics are quantitative measurements of a piece of software, e.g., a class, a package, or a component. A good understanding of software metrics supports the identification of possible problems in the development process and helps to improve the software quality. In this paper, we present two possibilities how novel visual representations can support the user to discover interesting properties within the metric data set. The first one uses a new interactive 3D metaphor to overcome known problems in the visualization of the evolution of software metrics. Then, we focus on the usage of 2D animation to represent metric values. Both approaches were implemented and address different aspects in human-centered visualization, i.e., the design of visual metaphors that are intuitive from the user perspective in the first case as well as the support of patterns in motion to facilitate the visual perception of metric outliers in the second case.

1 Introduction

Due to the ever increasing complexity of software and software architectures, the need to develop quantitative measurements of properties of software parts or specifications was very urgent and led to so-called *software metrics* that can serve as a input to quality assurance and various development processes as well as for a better understanding of a complex software system [Lin07]. They can be measured for binary code, source code (e.g., classes or packages), and other software representations, such as UML. Examples for popular software metrics are: lines of code (LOC), number of classes (NOC), weighted methods per class (WMC), depth of inheritance tree (DIT), and many more. From the perspective of the visualization community, the measured values of a whole software metric suite form a big multivariate data set, i.e., an entity (e.g., a class) is annotated with a set of variables/attributes (metric values).

As the development of a large software system is an evolutionary process where different versions of the source/binary code are stored in large repositories, e.g., CVS [FB03] or SVN [PCSF08], we must take this time-dependency of software metrics into account. Thus, important trends of these metrics should be highlighted in some way to give the software developers the opportunity to react in short time.

In J. Münch and P. Liggesmeyer, editors, Software Engineering 2009 - Workshopband,
pages 147-154, Lecture Notes in Informatics (LNI), vol. P-150, Kaiserslautern, Germany, 2009.

To find effective visual representations of the metric data that can evolve over time is a well-known challenge in the emerging field of software visualization. Several visualization approaches on this problem were published in the past. Because of the limited space, the authors refer to the proceedings of the ACM SoftVis Conference series, of the IEEE VisSoft Workshops, or of the IEEE International Workshops on Program Comprehension. However, we discuss one approach in Section 2.1 in more detail because it is strongly related to our own work.

We already mentioned that a large metric data set can be seen as (time-dependent) multivariate data. Thus, it is not surprising that such a data set is also interesting for the information visualization community. A first example is the work of D. Holten et al. [HVvW05]. They argue that the use of established techniques from computer graphics can enrich the palette of standard software visualization techniques to achieve more effective visual representations. In this way, they combined treemap techniques [ATS95] with color and graphical textures to represent sets of metric data. Treemaps are used to represent the hierarchical structure of the associated source code. Another, similar work is the application of so-called Voronoi treemaps for the visualization of software metrics [BDL05]. These examples show that there is a convergence between software visualization and information visualization. And as a consequence of the fact that the latter field is traditionally related to HCI, human-centered aspects also become more important for software visualization [KEM07].

In this paper, we present two novel visual representations which can support the user to discover interesting properties within a metric data set. The first approach, discussed in Section 2, uses a new intuitive 3D metaphor to overcome known problems in the visualization of the evolution of software metrics. Then, we illuminate the usage of 2D animation to represent time-independent metric values. This approach addresses another aspect in human-centered visualization: the support of patterns in motion to facilitate the visual perception of metric outliers.

2 Software Metrics Using 3D

In this section, we describe our improvement of an existing Kiviat diagram approach presented by Pinzger et al. [PGFL05] at the ACM SoftVis Conference 2005. Their idea was that Kiviat diagrams, also called star plots, can be used to visualize time-dependent, multivariate data, such as software metrics gathered from several releases of source code. We will briefly discuss this work in the following.

2.1 Current Approach

Pinzger et al. developed an improvement of a standard Kiviat diagram approach (cf. for example [CCKT83]) in order to facilitate the visualization of multiple software releases and of release history data. Each value of a metric is measured across n releases and plotted in the diagram. Adjacent measures of metrics belonging to the same release are than connected via polylines. The resulting diagram consists of polygons for each release.

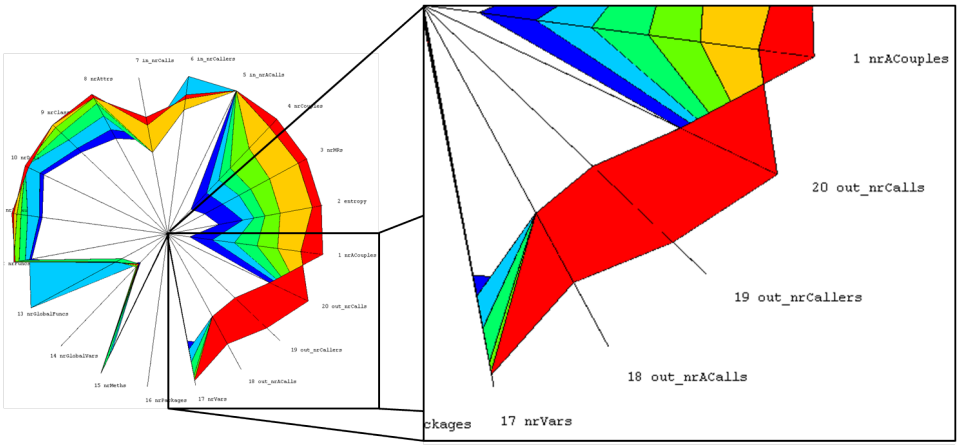


Figure 1: Screenshot example of the improved Kiviat diagram based on the work [PGFL05]. The metric values are hard to perceive due to overlaps in some parts of diagram (e.g., for the metric numbers 18, 19, and 20)

However, the values of different metrics can also decrease from release to release. This results in a polygon overlapping that hides information, see Figure 1.

This overlapping problem was solved by the authors in two ways. One solution is based on color coding the releases in order to present the time order of the releases. Another way is to introduce a new metric axis (or to use an already existing metric axis, such as release number) which encodes a sequence of releases. Since this metric will show the chronological order of releases, we might determine the increase or decrease of other metric values.

However, this solution does not solve the overlapping problem completely. We still have cases of overlaps which makes it impossible to see certain metric values, as our screenshot example in Figure 1 shows in the focused area at the right hand side. We will present our solution of this concrete problem in the next subsection.

2.2 3D Kiviat Diagrams

A first idea could be to use transparencies in order to overcome that problem. But considering the fact that individual releases are color-coded, this approach is not very useful: by introducing a transparency, overlapped colors will change (e.g., yellow over blue will yield green) which makes the values hard to perceive.

We decided to use 3D on the contrary. Instead of placing the Kiviat diagrams in a kind of stack, as discussed in the work of Fanea et al. [FCI05], we developed an *fanning-out metaphor* which is intuitive and space-filling [Guo08]. We can now interact with this diagram by *fanning out* specific metrics axes into the third dimension in the way shown in Figure 2. Thus, we can examine specific metrics per release, even those, which are hidden

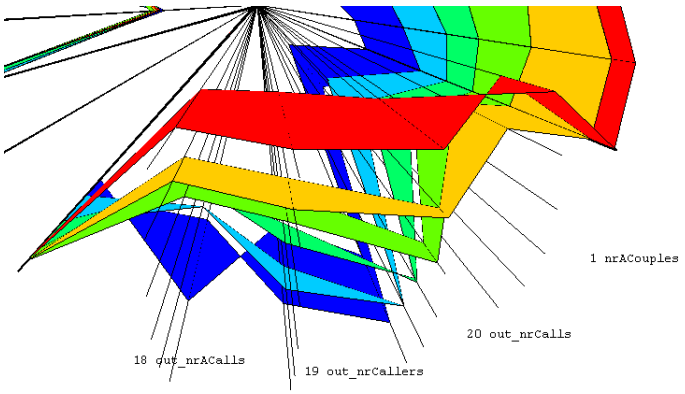
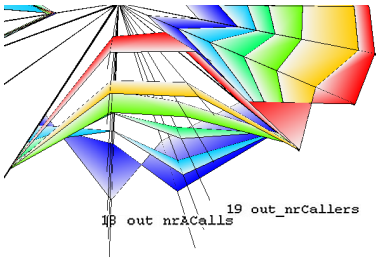
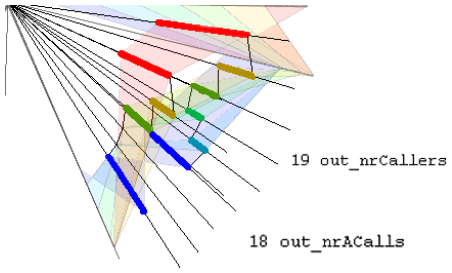


Figure 2: Users can fan out specific metric axes in order to analyze hidden information caused by overlaps (the figure only shows the underpart of one of our 3D Kiviat diagrams)



(a) A gradient helps the user to determine the directions of value changes



(b) The transparency makes it easy to filter out unnecessary data and to focus on specific ones

Figure 3: Different cut-out views of selected metric axes

by overlaps. Of course, users can rotate and move the diagram in order to get the best possible view.

Sometimes, even when our fanning-out technique is used, the direction of value changes is hard to perceive in 3D. Users have troubles to decide whether the value has increased or decreased. In order to avoid this, we introduced a gradient color coding between the releases. The darker side of the gradient shows the positive direction of a change. Thus, it is easier to see if a change is negative or positive, cf. Figure 3(a). In some cases there is also a need to filter out some metrics and to focus on specific ones. Our approach supports the use of transparency to hide unimportant metrics. The results of this kind of interaction are shown in Figure 3(b).

Our 3D Kiviat diagram approach using the fanning-out metaphor, combined with the gradient and transparency features, gives a possibility to see the data hidden by the overlaps. An important question is now whether a user accepts/understands this new metaphor together with the offered features. To get more insight into this problem, we performed a brief usability study with our students.

2.3 Usability Study

The aim of the study was to get a feedback about the interaction with our visualization approach. We had ten students in total who took part in the study. Most of them were master and bachelor students, and two were new PhD students. All of them had a computer science background and were more or less familiar with software metrics.

Each student had to finish two tasks which involved finding the changes of particular metrics throughout all releases. We used the data set presented in the original paper [PGFL05] for our study. Each student had 30 minutes to use the application before they got the tasks, which had to be finished in 15 minutes. For each task, they should show whether a specific metric has a positive or negative change between the releases: without using any of our improvements in the first task and with the help of our approach in the second one. Note, that our solution corresponds to the work of Pinzger et al.[1] if no 3D interaction is used. After finishing both tasks, they should answer a questionnaire and to give constructive comments about the prototype.

The results of the study showed—of course—that it was impossible to find the hidden information in the case of overlaps. However, after checking the answers of the tasks, we also found that not all students have answered correctly even when using our 3D interactions. Six students gave absolutely correct answers. One of them gave inverse answers, which makes us believe that he misunderstood the meaning of the gradient metaphor, that he has interpreted in an opposite way.

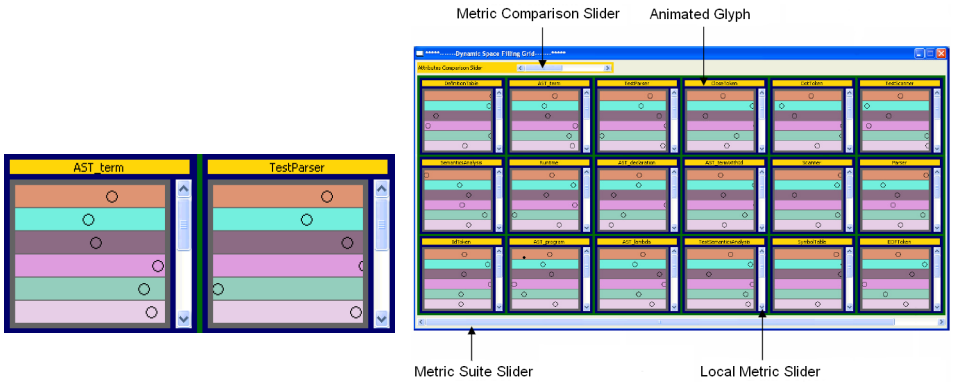
We think that the results might be better if we give them more time to use the prototype application. This was also confirmed from the results from our questionnaire analysis where some students complained about the short time provided. When the students were asked about the best feature or functionality of the prototype, most of them answered that it is fanning-out feature and the 3D manipulation of the diagram. The students agreed that through use of our 3D approach "all the releases can be understood easily". Other comments regarding the interaction possibilities were mainly positive as the reader can conclude from the results of the following two sample rating questions (rating scale: 1–5).

- "In your opinion, how natural and/or intuitive is the 3D Kiviat diagram metaphor for you?" Average rating: 3.75[2]
- "Was it easy to find the required information?" Average rating: 3.8

This was a first usability study conducted in order to get initial feedback about our approach and not a real evaluation. We wanted to know if the metaphor and other interaction techniques are intuitive and helpful for the users. The data from the study suggests that we are on the right track with changes that should be implemented in future, such as the possibility to display the current metric values.

[1] Regarding the Kiviat diagrams.

[2] Two students have not answered this question, so the result is the average of the answered ones.



(a) Metric values of two selected classes (b) Tool screenshot

Figure 4: Motion patterns to represent software metrics

3 Animated Patterns to Represent Software Metrics

The previously discussed approach is static in the sense that a non-animated visual representation was used to display the metric values. But, we can also use motion as a display technique to represent that data. This kind of dynamic representation is not well understood compared to the perception of static representations, however, it may be an alternative possibility to display at least certain aspects of a data set [War04].

Based on the fact that human beings are very sensitive to motion patterns, we started a novel project on a dynamic approach for the glyph-based visualization of software metrics. In general, glyphs have been heavily used in information visualization. They are graphical entities that convey multiple data values via visual attributes, such as shape, color, position, or size [War02]. There is a considerable literature on glyphs and a lot of tools use them. We pick out one example work for a specific domain written by Chuah and Eick [CE98]. They describe glyphs for software project management data where abstract data and time are mapped to specific visual attributes of glyphs.

We extended these approaches by moving patterns that represent metric values for Java classes [Maj08]. Figure 4(a) gives an impression about the main idea: the metric values of two classes AST_Term and TestParser are shown, whereas the different metrics are represented by colored horizontal bars. The metric values themselves were coded as circles that move horizontally. The (constant) speed of a circle represents the current value normalized over the range of the underlying metric set. Its moving direction incorporates positive or negative values respectively. We implemented a simple software prototype to get a feeling of the perceptual power, cf. Figure 4(b). Different kinds of sliders are used to focus on specific classes (Metric Suite Slider), on specific metrics of one class if the space is not sufficient within one glyph (Local Metric Slider), and on specific metrics of all specific classes (Metric Comparison Slider). Moving the Metric Comparison Slider thumb corresponds to a synchronous movement of all Local Metric Slider thumbs.

We will use this implementation as a basis for the development of more advanced dynamic glyphs. The current ones are, for example, very space-consuming which should be improved. A first small usability test has shown that the perception of such animated glyphs seems to be better compared with a version using textures for the representation of metric values, but only if the number of animated objects per class is small ($\approx$ five circles). Especially metric outliers are quickly discovered.

4 Conclusions and Future Work

In this paper, we presented two novel visual representations including their prototypical implementations, which can support the user to discover interesting properties within a metrics data set. The first one uses a new interactive 3D metaphor to overcome known problems in software metric visualization. The second one is on the usage of 2D animation to represent the metric values.

Both approaches are ongoing projects of our research group and need to be carefully evaluated. Especially the dynamic approach is at the beginning of our investigations to bring motion perception findings into software visualization. Our future work will be directed to analyze the possibilities that this approach will offer, to work on space-filling variants, and to extend it to alternative display technologies.

Acknowledgments: We would like to thank Guo Yuhua for implementing the 3D approach to visualize software metric evolutions described in Section 2. Furthermore, we are thankful to Raja Majid for his prototypical implementation of animated patterns shortly discussed in Section 3.

References

[ATS95] T. Asahi, D. Turo, and B. Shneiderman. Visual Decision-Making: Using Treemaps for the Analytic Hierarchy Process. In *CHI 95 Conference Companion*, pages 405–406, 1995.

[BDL05] M. Balzer, O. Deussen, and C. Lewerentz. Voronoi Treemaps for the Visualization of Software Metrics. In *Proceedings of the 2005 ACM symposium on Software Visualization (SoftVis '05)*, pages 165–172, New York, NY, USA, 2005. ACM.

[CCKT83] J.M. Chambers, W.S. Cleveland, B. Kleiner, and P.A. Tukey. *Graphical Methods for Data Analysis*. Wadsworth, Belmont, CA, 1983.

[CE98] Mei C. Chuah and Stephen G. Eick. Information Rich Glyphs for Software Management Data. *IEEE Computer Graphics and Applications*, 18(4):24–29, 1998.

[FB03] Karl Fogel and Moshe Bar. *Open Source Development with CVS*. Paraglyph Press, 3rd edition, 2003.

[FCI05] E. Fanea, S. Carpendale, and T. Isenberg. An Interactive 3D Integration of Parallel Coordinates and Star Glyphs. In *Proceedings of the 2005 IEEE Symposium on Information Visualization*, pages 149–156, 2005.

[Guo08] Y. Guo. Implementation of 3D Kiviat Diagrams. Bachelor's thesis, Växjö University, Sweden, 2008.

[HVvW05] D. Holten, R. Vliegen, and J.J. van Wijk. Visual Realism for the Visualization of Software Metrics. In *Proceedings of Visualizing Software for Understanding and Analysis (VISSOFT 2005)*, pages 1–6, 2005.

[KEM07] Andreas Kerren, Achim Ebert, and Jörg Meyer, editors. *Human-Centered Visualization Environments*. LNCS Tutorial 4417. Springer, Heidelberg, 2007.

[Lin07] R. Lincke. *Validation of a Standard- and Metric-Based Software Quality Model*. Licentiate thesis, Växjö University, Sweden, 2007.

[Maj08] R. Majid. Dynamic and Static Approaches for Glyph-Based Visualization of Software Metrics. Master's thesis, Växjö University, Sweden, 2008.

[PCSF08] C.M. Pilato, B. Collins-Sussman, and B.W. Fitzpatrick. *Version Control with Subversion*. O'Reilly Media, Inc., 2nd edition, 2008.

[PGFL05] M. Pinzger, H. Gall, M. Fischer, and M. Lanza. Visualizing Multiple Evolution Metrics. In *Proceedings of the 2005 ACM Symposium on Software Visualization (SoftVis '05)*, pages 354–362, St. Louis, Missouri, 2005.

[War02] Matthew O. Ward. A Taxonomy of Glyph Placement Strategies for Multidimensional Data Visualization. *Information Visualization*, 1:194–210, 2002.

[War04] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, 2nd edition, 2004.