Gestalt Principles Applied To Software Engineering Diagrams: An Initial Study

Krystle Lemon[1], Edward B. Allen[1], Jeffrey Carver[1], and Gary Bradshaw[2]

[1]Department of Computer Science and Engineering
300 Butler Hall, Box 9637
Mississippi State, MS 39762
+1 662-325-2756

{kdl18, carver}@cse.msstate.edu;
edward.allen@computer.org

[2]Department of Psychology
Box 6161
Mississippi State, MS 39762
+1 662-325-0550

glb2@ra.msstate.edu

ABSTRACT

Discovering root-causes of comprehension errors in software design is important to prevent their presence in software systems. This research synthesizes software engineering and Gestalt principles of similarity, proximity, and continuity for the purpose of discovering whether certain visual attributes of diagrams (dashed arrows, severe complexity, etc.) can affect the accuracy and efficiency of understanding correct relationships amongst the entities in the diagram. Twenty-seven subjects viewed diagrams of different types and answered questions about them. The experiment tested whether two dependent variables, accuracy and response time, were significantly affected by independent variables, diagram type (*simple1*, *simple2*, *complex*), Gestalt principles (good vs. bad), and forward/backward (question order). The results of this study indicated that the Gestalt principles did affect the comprehension in the *complex* diagrams.

Keywords

Gestalt principles, diagram comprehension, empirical software engineering, software architecture, cognitive science

1. INTRODUCTION

The long-term goal of this work is to find ways to prevent errors early in the software development lifecycle that may be difficult and expensive to correct later. Software engineers routinely use diagrams that depict component relationships. Such diagrams often represent the architecture of a system, modeling the structure, behavior, relationships, and constraints among components while ignoring implementation details [3]. Because software system implementation is based on the software architecture, misinterpretation of software architecture diagrams could easily result in an incorrect system implementation. Therefore, it is important that software engineers understand software diagrams to prevent errors from propagating to later phases of the software-engineering life cycle.

With the aid of cognitive science, which seeks to understand the human mind and how it learns, this work seeks to identify factors that can impair comprehension of software-engineering diagrams [4]. In cognitive science, Gestalt principles of perceptual organization deal with features that combine to form overall perception, such as the relation of figure to ground and relationships among visual features [1]. Cognitive-science research results in areas such as visual search and tracing may help uncover root-cause errors in comprehension of diagrams. This paper presents the results of a study coupling cognitive science and software engineering to investigate whether certain diagram characteristics affect comprehension by software engineers.

This paper is organized as follows. Section 2 provides an overview of diagram comprehension and related work. Section 3 describes the study. Results, discussion, and threats to validity are presented in Section 4. Lastly, Section 5 discusses the conclusions and future work.

2. DIAGRAM COMPREHENSION

Software-architecture diagrams often show the flow of information between components in a software system. Such diagrams consist of lines with arrowheads pointing in the direction of information flow, boxes representing system components, and annotations. Software engineers must comprehend all of this information and process it within their mental workspace [3]. Hungerford, et al. showed that diagrams impose less cognitive load compared to text [2]. Not surprisingly, software engineers routinely make extensive use of a variety of diagrams.

The Gestalt principles address why individuals can perceive whole elements out of incomplete elements. They address how objects are viewed in relation to figure and ground, similarity, proximity, continuity, closure, area, and symmetry [1].

This study focuses on the Gestalt principles of similarity, proximity, and continuity. The principle of *similarity* states that objects with similar characteristics belong together. The principle of *proximity* states that objects that are close together are perceived as belonging together. Finally, the principle of *continuity* states that continuous figures are perceived more often than non-continuous figures. An example of a continuous figure is when a line is perceived to pass through an object instead of viewing it as two separate lines on the object: one entering and another leaving. Therefore, some of our diagrams were drawn with these principles in mind, with solid lines with similar boxes grouped together, and with minimum spacing between them. This initial study did not try to separate out the effects of each of these principles; rather they were all taken together. This study investigated two research questions, in the context of diagram comprehension.

Hypothesis 1: Diagrams that follow Gestalt principles of similarity, proximity, and continuity offer better accuracy than diagrams that do not.

Hypothesis 2: Diagrams that follow Gestalt principles of similarity, proximity, and continuity offer faster response time than diagrams that do not.

3. THE STUDY

The study was conducted at Mississippi State University (MSU) in the Fall of 2005, using the Adobe Authorware software to administer the experiment. The 27 subjects came from the Software Architecture course (15 subjects) and the Introduction to Software Engineering course (12 subjects). These subjects were upper-level undergraduates and graduate students. The software architecture course covers basic software architecture concepts, as well as, creation and use of architecture diagrams. The Introduction to Software Engineering focuses on software engineering processes, including the diagrams that are used during those processes.

To simulate real-world diagrams, three diagrams, created by software-engineering students as part of a homework exercise during a previous semester of the Software Architecture course, were selected and modified. Each diagram contained named boxes (system components) and lines with arrowheads (representing information flow). The three diagrams were labeled *simple1*, *simple2*, and *complex*, where *simple1* and *simple2* diagrams where similar in number of boxes and lines in the diagrams and the *complex* diagram had approximately three times as many lines and twice as many boxes in the diagram. For each of the three diagrams, two topologically equivalent versions were produced. The *good* version of the diagram was created using the Gestalt principles (described in Section 2), where grouping, proximity, and continuity where adhered to in the drawing. The *bad* version of each diagram was simply the original diagram with the same box names and connections as the good version without consideration of the Gestalt principles. Figures 1 and 2 show the good and bad versions of the *complex* diagrams.

Each subject viewed *simple1* diagram followed by *simple2* diagram followed by *complex* diagram. Subjects were given 20

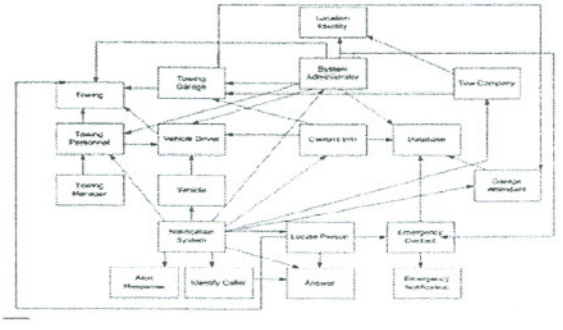

Figure 1: Complex diagram good version

questions per diagram. All students answered the same questions. However, to assess the potential effect of question order, some answered questions 1 through 60 (*forward*) (1 through 20 for each diagram) and some answered questions 60 through 1 (*backward*) (20 through 1 for each diagram). The questions asked the subjects to determine whether messages were sent between objects in the diagram (i.e. whether there was a line connecting the boxes). An example question is "Does Location Identity send messages to Towing Garage?" The subjects were split into four balanced groups as shown in Table 1.

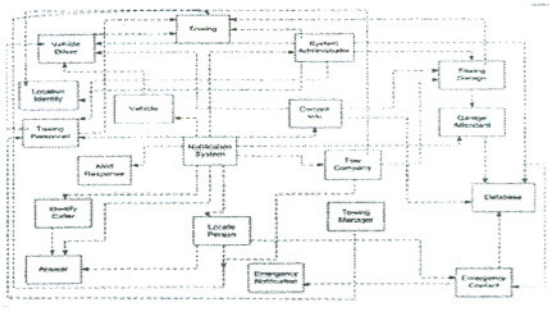

Figure 2: Complex diagram bad version

The subjects were given as much time as needed to answer

Table 1: Experiment Design

Group	Simple1	Simple2	Complex	Fwd/Backwd
1	Good	Bad	Good	Forward
2	Good	Bad	Good	Backward
3	Bad	Good	Bad	Forward
4	Bad	Good	Bad	Backward

the questions. At their own pace, they proceeded to answer the questions one by one while the associated diagram was still visible. They were not allowed to skip any questions.

4. RESULTS and DISCUSSION

This section describes the statistical analysis performed on the quantitative data in order to determine whether a significant difference in diagram comprehension exists between different types of diagrams. As this was an initial study, an alpha value of 0.1 was chosen for significance tests.

4.1 Summary of Quantitative Data

This study had two dependent variables and three independent variables. The dependent variables investigated in this study were *question accuracy* and *response time*. The independent variables were: *Gestalt principles used* (good vs. bad), *type of diagram* (*simple1*, *simple2*, or *complex*), and *question order* (forward or backward). The question order was used as a control variable to determine whether the presentation order of questions had any effect on subject response.

The goal of the statistical analysis was to determine if any of the independent variables had a significant effect on either of the dependent variables. To perform an outlier analysis, we made a scatter plot with the subjects' response time on one axis and the number of correct responses on the other axis. A visual inspection of this chart showed that the accuracy for one subject was more than two standard deviation below the mean. Therefore, this subject's data was removed as an outlier and was not included in any further analysis.

For the dependent variable *question accuracy*, most subjects were very accurate with low variability across all diagrams. Further analysis of this variable was unlikely to show interesting results. Therefore, the remainder of this section focuses on the *response time* variable.

Three separate ANOVA tests were run for the *simple1*, *simple2*, and *complex* diagrams to isolate the effects of the independent variables for each diagram type. The ANOVA test for response time for *simple1* diagrams was not significant. The ANOVA test did show an interaction between the variables *forward/backward* and *good/bad* with $F_{(27,1)}$ =5.896 and p=0.023. Further analysis was done using independent sets t-test, isolating the effects of the *forward/backward* variable and the *good/bad* variable. The *forward/backward* variable was not significant. The *good/bad* variable was not significant alone.

The *simple2* diagram ANOVA statistical tests yielded insignificant results for all variables and further t-tests confirmed this. The ANOVA tests used both the *accuracy* and *response time* as dependent variables to model the significance of *the diagram type* variable, *good/bad* variable, and the *forward/backward* control variable. A similar analysis was completed for the *complex* diagrams and the results showed that the good/bad variable was very significant where $F_{(27,1)}$ = 0.001 and p=0.004.

4.2 Discussion of Results

The results obtained suggest that the range of complexity among the diagrams presented in the experiment was not sufficient to obtain a useful variation in accuracy results. Consequently, Hypothesis 1 remains unevaluated.

Figure 3 depicts the distribution of response times. The statistical tests showed that the Gestalt principles for the *complex* diagram significantly affected the response time variable. Therefore, we conclude that the comprehension time of good diagrams was faster than that of bad diagrams providing support for Hypothesis 2. This conclusion encourages future experimentation to determine which Gestalt principles can be applied in software engineering to make more comprehensible diagrams and what features should be avoided.

4.3 Threats to Validity

By using two different question orderings a potential threat caused by the ordering of the questions was assessed. The analysis showed that question order had no effect on either accuracy or time. By balancing the presentation of good and bad diagrams, the influence of individual abilities was balanced.

Although we had only three pairs of diagrams, decreased response times from *simple1* to *simple2* diagrams suggest that a learning effect occurred between the subjects viewing the first and second diagrams. This threat to validity was not addressed. The increase of time from the *simple2* to *complex* diagram could correspond to subjects having a higher mental workload because the complexity increases, as expected. This study did not evaluate the effect of complexity. Due to the fact that the subjects were students, a threat to external validity is present. However, the students' tasks were similar to professional subjects' tasks.

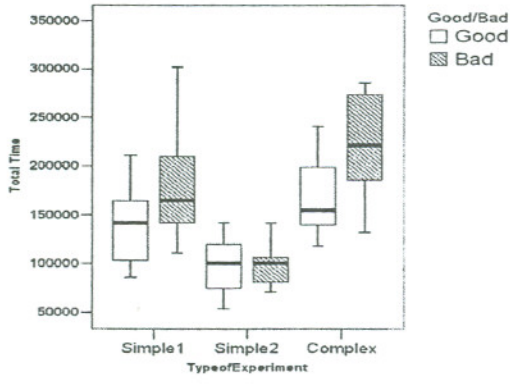

Figure 3: Distribution of response time in milliseconds

5. CONCLUSIONS

This initial experiment investigated the basic question of whether certain factors affect diagram comprehension. The results of this study showed that the Gestalt principles did affect the comprehension in the *complex* diagram. Gestalt principles of perceptual organization show promise in easing the task of comprehending software-engineering diagrams. This research aims at helping software engineers to determine what type of artistic approaches to consider when designing the software architecture diagrams to help avoid errors in the later stages of the software-engineering lifecycle.

Future work will include studies that analyze various types of diagram features and the effect of each Gestalt principle.

6. ACKNOWLEDGMENTS

Our thanks go to Ginger Cross, student subjects, MSU ESE research group, Byron Williams, and F. Chevonne Thomas. This work was supported in part by NSF grant CCR-0132673.

7. REFERENCES

[1] J. A. Anderson, *Cognitive Psychology and Its Implications*, W. H. Freeman and Company, New York, New York, 1990, 66-68.

[2] B. C. Hungerford, A. R. Hevner, and R. W. Collins, "Reviewing Software Diagrams: A Cognitive Study," *IEEE Transactions on Software Engineering*, vol. 30, no. 2, Feb 2004, 84-95.

[3] T. Klemola and J. Rilling, "Modeling Comprehension Processes in Software Development," *Proceedings: First IEEE International Conference on Cognitive Informatics*, Aug 2002, 329-336.

[4] J. Reason, *Human Error*, Cambridge University Press, Cambridge, United Kingdom, 1990.

[5] P. Thagard, *Mind: Introduction to Cognitive Science*, MIT Press, Cambridge, Massachusetts, 2005.