UML and Box-and-Line Diagrams in Software Design Comprehension

Satrio Adi Rukmono
Mathematics and Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands
email: s.a.rukmono@tue.nl

Michel R. V. Chaudron
Mathematics and Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands
email: m.r.v.chaudron@tue.nl

***Abstract*—Graphical models of software systems are used for understanding a system in communication between stakeholders as well as a blueprint used for a system's construction. Visual explanations serve as a concrete reference regarding the behavior, causality, and function of the system. However, whilst there is often agreement that a visual representation of a system is much clearer and more understandable than a textual description, there is little research regarding how standardised models, such as UML, and non-standardised box-and-line diagrams differ in their effectiveness of conveying software system design. We seek to find if there are any considerable differences in the effectiveness of UML compared to box-and-line diagrams in software design comprehension.**

***Index Terms*—uml, diagrams, software design comprehension, software architecture recovery**

I. Introduction

Graphical models of software systems are used for understanding a system, in communication between stakeholders, as well as a blueprint used for a system's construction. There is much to be said about the advantages of a graphical representation of a system. Bobek et al. [1] assert that visual explanations serve as a concrete reference regarding the behaviour, causality, and function of the system. However, whilst there is often agreement that a visual representation of a system is much clearer and more understandable than a textual description [2], [3], there is little research regarding how standardised models, such as UML, and non-standardised box-and-line (B&L) diagrams differ in their effectiveness of conveying software system design.

According to the encoder/decoder model of communication [4], [5], a receiver of a message needs to employ a decoder that is "compatible" to the sender's encoder for a communication to be successful. While this model was created in the context of technology-assisted communication, the core idea holds true for broader types of communication. In communicating software design using diagrams, for example, it is not hard to argue that a convention of what each symbol in the diagram represents will help the communicating parties understand the system of interest more easily. This is the premise behind the standardisation of modeling languages. However, in practice, non-standardised box-and-line diagrams are also used, either in place of or in conjunction with the standardised ones. Sometimes architects and developers sketch rough box-and-line diagrams on a whiteboard during discussion and then take a picture of it to send to their teammates. But their use does not seem to stop with informal setting as, for example, both volumes of *The Architecture of Open Source Applications* (AoSA) [6], [7] are littered with such diagrams as shown in Figure 1. The reasoning behind their use seems to fall on their simplicity and flexibility. In our teaching experience, students often wonder if people in real life use UML due to the perceived complexities of having to understand the meaning of many different symbols. From this phenomenon, we seek to find if there are any considerable differences in the effectiveness of UML compared to box-and-line diagrams in software design comprehension.

II. Objectives

The difference of effectiveness between the two diagram types in conveying information about software design are not obvious. While using standardised models helps in providing baseline understanding of design elements, there may be differences in the dominating type of cognitive load in accomplishing the task [8]. The objective of our study is to *analyse* the effectiveness of *UML and box-and-line diagrams* from the point of view of *software engineers* in the context of *communicating software architecture design*.

We formulate the following research question:

RQ Is either UML or B&L diagram better than the other at conveying software architecture design?

To approach this question and understand the reasoning, we define the following sub-questions:

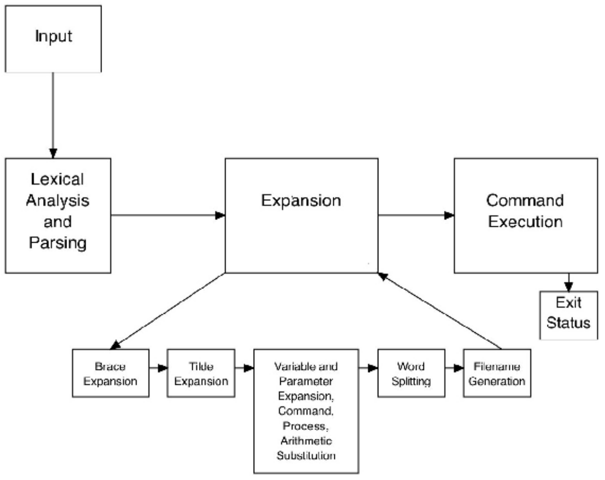
RQ$_A$ Which aspects of software architecture design are better communicated using either type of diagram?

RQ$_B$ What is the difference, if any, in the cognitive load involved in the task of comprehending software architecture design using the two types of diagram?

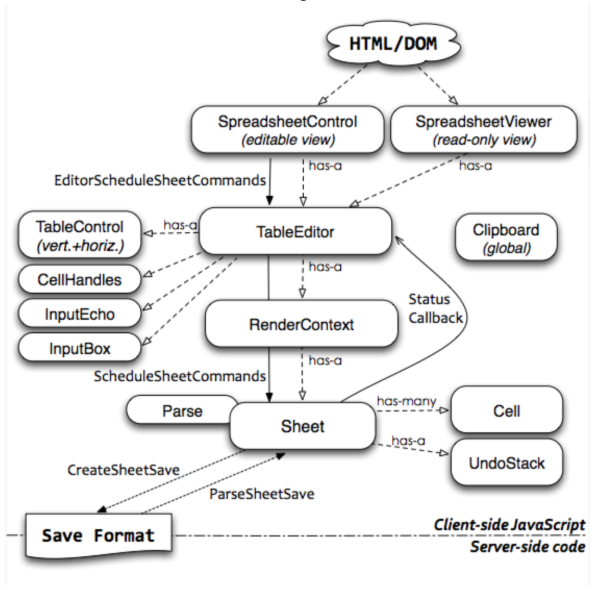
III. Considerations and Scoping

We take the following items as consideration for scoping our study and developing the diagrams and questions that we are going to use in the study.

Design aspects and architecture views. We are considering to include logical, behavioral, and execution/deployment view of the system in question.



(a) The Bourne-Again Shell (bash).



(b) SocialCalc.

Fig. 1: Non-standardized architecture diagrams of two different applications as depicted in AoSA vol. I [6].

Standardised meaning or semantics of diagram elements. Boxes and lines in box-and-lines diagram do not have standardised semantics. As such, we need to be mindful of the intended vs. perceived meaning of the diagrams we show to the study participants.

Textual companion. Diagrams are usually complemented by textual descriptions in design documents.

Mixing views. Diagrams sometimes mix abstractions or views.

Levels of abstraction. Different diagrams target different level of abstraction or detail.

Domain knowledge. To avoid bias, we plan to take a domain that is likely to be either reasonably familiar to all participants, or not familiar to most.

IV. Experiment Design

We plan to conduct the study in an online setting. Participants will be asked to answer questionnaire forms accessible via a web browser. The driving force of this method are ease of distribution and simpler logistics, possibility for reaching diverse participants, and immediacy of results for analysis.

Participants will first be asked to fill out a pre-questionnaire to understand their profile, including questions about their educational and professional background, age, gender, nationality, native language, and their preference and self-assessment of skills relating to both diagram types. This aims to understand the risk of bias involved for generalising the result of this study.

Each participant will be presented with two sets of design diagrams for two different systems accompanied with some textual description of the systems. Next, they will be asked to answer questions about the two systems. The questions will be designed around categories derived from the 4+1 view of architecture [9]. There may be more than one category of questions for each architecture view. Participants will not be made aware of this categorisation.

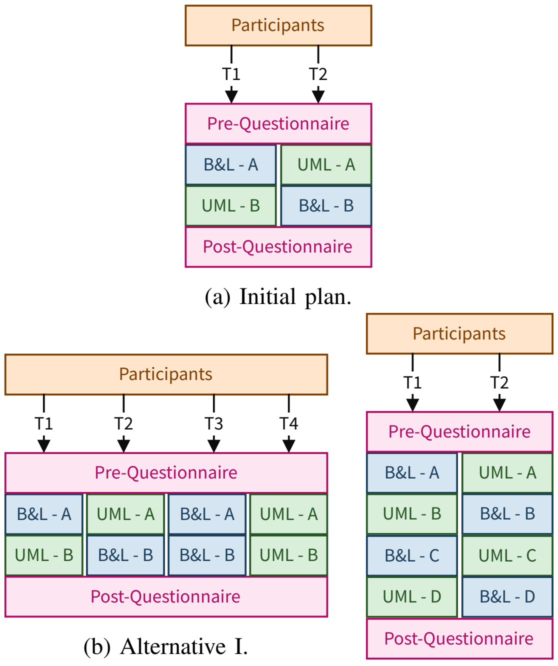
Afterwards, participants will be asked to fill out a post-questionnaire that measures the cognitive load of the design comprehension task. This post-questionnaire adopts the instrument for measuring different types of cognitive load devised by Leppink et al. [10].

A. Treatment Groups

We plan to split the participants into two to four treatment groups. The first two groups will be given different types of diagrams for each system they need to examine. We will call the systems A and B, and therefore the first group will receive UML diagram of system A and B&L diagram of system B. The second group will receive the opposite, i.e, B&L diagram of system A and UML diagram of system B. In a preliminary study using this treatment, we found that the performance scores for the second system (the B system) are lower that the first system (the A system), despite both having comparable size and complexity. To better understand this, we are considering two options in refining the experiment. The first is to include two more treatment groups, each only receiving one type of diagrams for both systems. That is, the third group will receive UML diagrams of both system A and B, and the fourth group will receive B&L diagrams for both systems. The second option is to use the original two treatment groups but add more systems, C and D, with alternating diagram types as in the original systems A and B. Figure 2 illustrates these alternatives.

B. Participants

We are populating a list of candidates to participate in the study. Currently, the list includes software engineering students of different levels (bachelor's, master's, PhD's, and engineering doctorates) and professionals in software engineering industry. The candidates come from different countries, including Indonesia, Malaysia, Netherlands, Slovakia,



(a) Initial plan.

(b) Alternative I.

(c) Alternative II.

Fig. 2: Treatment group alternatives.

Sweden, and Uganda, unavoidably with different levels of English language fluency.

We discern the apparent difference in how people with different genders think, as popularly suggested by Pease & Pease's book *Why Men Don't Listen & Women Can't Read Maps* [11], to have a possible impact on how someone navigates software design diagrams. However, the candidate list that we currently have is dominated by males. We seek to diversify this for better representations before conducting the study.

C. Variables

The study will involve variables as described in Table I. As previously mentioned, question categories will be derived from the 4+1 view of software architecture, that includes *logical*, *process*, *development*, and *physical* views as well as *scenarios*. Diagram types are split into two groups, UML and B&L. We have not decided on which types of UML diagrams we should include, but at the very least it should include class diagrams as it is one of the most prominent type of UML diagram.

The three variable categories concerning participants will be gathered using the questionnaires.

TABLE I: Experiment variables.

	variable	*type*	*source*
Question Characteristics			
1.	Question category	nominal	experiment design
2.	Diagram type	nominal	experiment design
Participant Characteristics			
3.	Educational background	ordinal	pre-questionnaire
4.	Experience	ordinal	pre-questionnaire
5.	Modeling skills	interval	pre-questionnaire
6.	English language fluency	nominal	pre-questionnaire
7.	Gender	nominal	pre-questionnaire
Participant Scores			
8.	Score for UML	ratio	questionnaire grading
9.	Score for B&L	ratio	questionnaire grading
10.	Time per question	ratio	questionnaire timestamp
Participant Cognitive Load			
11.	Extraneous processing	interval	post-questionnaire
12.	Essential processing	interval	post-questionnaire
13.	Generative processing	interval	post-questionnaire

References

[1] E. Bobek and B. Tversky, "Creating visual explanations improves learning," *Cognitive research: principles and implications*, vol. 1, no. 1, pp. 1–14, 2016.

[2] R. Jolak, M. Savary-Leblanc, M. Dalibor, A. Wortmann, R. Hebig, J. Vincur, I. Polasek, X. Le Pallec, S. Gérard, and M. R. Chaudron, "Software engineering whispers: The effect of textual vs. graphical software design descriptions on software design communication," *Empirical Software Engineering*, vol. 25, no. 6, pp. 4427–4471, 2020.

[3] B. Tversky, "Multiple models. in the mind and in the world," *Historical Social Research/Historische Sozialforschung. Supplement*, no. 31, pp. 59–65, 2018.

[4] C. E. Shannon, *The mathematical theory of communication, by CE Shannon (and recent contributions to the mathematical theory of communication), W. Weaver*. University of illinois Press Champaign, IL, USA, 1949.

[5] C. Meinel and H. Sack, *Digital communication: Communication, multimedia, security*. Springer Science & Business Media, 2014.

[6] A. Brown and G. Wilson, *The Architecture of Open Source Applications: Elegance, Evolution, and a Few Fearless Hacks*. Lulu, 2011, vol. 1.

[7] ——, *The Architecture of Open Source Applications, Volume II: Structure, Scale, and a Few More Fearless Hacks*. Lulu, 2012, vol. 2.

[8] R. E. Mayer, *Multimedia Learning*, 3rd ed. Cambridge University Press, 2020.

[9] P. B. Kruchten, "The 4+ 1 view model of architecture," *IEEE software*, vol. 12, no. 6, pp. 42–50, 1995.

[10] J. Leppink, F. Paas, C. P. Van der Vleuten, T. Van Gog, and J. J. Van Merriënboer, "Development of an instrument for measuring different types of cognitive load," *Behavior research methods*, vol. 45, no. 4, pp. 1058–1072, 2013.

[11] A. Pease and B. Pease, *Why Men Don't Listen & Women Can't Read Maps: How to spot the differences in the way men & women think*. Hachette UK, 2016.