Journal of Software Engineering and Applications, 2023, 16, 483-495
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

The User Interfaces Transition Diagram-Editor: A Tool to Simplify User-System Interaction Modeling

Maria C. Gómez-Fuentes, Jorge Cervantes-Ojeda, Alan Badillo-Salas

Department of Applied Mathematics and Systems, Universidad Autónoma Metropolitana, Mexico City, Mexico
Email: mgomez@cua.uam.mx, jcervantes@cua.uam.mx, alan@nomadacode.com

How to cite this paper: Gómez-Fuentes, M.C., Cervantes-Ojeda, J. and Badillo-Salas. A. (2023) The User Interfaces Transition Diagram-Editor: A Tool to Simplify User-System Interaction Modeling. *Journal of Software Engineering and Applications*, **16**, 483-495.
https://doi.org/10.4236/jsea.2023.169023

Received: June 28, 2023
Accepted: September 15, 2023
Published: September 18, 2023

Abstract

The User Interface Transition Diagram (UITD) is a formal modeling notation that simplifies the specification and design of user-system interactions. It is a valuable communication tool for technical and non-technical stakeholders during the requirements elicitation phase, as it provides a simple yet technically complete notation that is easy to understand. In this paper, we investigated the efficiency of creating UITDs using draw.io, a widely used diagramming software, compared to a dedicated UITD editor. We conducted a study to compare the time required to use each tool to complete the task of creating a medium size UITD, as well as the subjective ease of use and satisfaction of participants with the dedicated Editor. Our results show that the UITD editor is more efficient and preferred by participants, highlighting the importance of using specialized tools for creating formal models such as UITDs. The findings of this study have implications for software developers, designers, and other stakeholders involved in the specification and design of user-system interactions.

Keywords

UITD, User Interfaces Flow Specification, Requirements Specification, Modelling Notation

1. Introduction

The User Interface Transition Diagram (UITD) is a formal modeling notation that simplifies the specification and design of user-system interactions. It can be used to model the flow of user interfaces that the system will have, which makes it a valuable communication tool for technical and non-technical stakeholders

during the requirements elicitation phase. Additionally, it is technically accurate, allowing it to be used reliably to start the development of the modelled system [1].

Simplicity is key when working with modeling notations, especially for non-experts [2]. The UITD provides a notation that is both complete and simple enough for non-technical stakeholders to understand [3]. It is a formal modeling notation that is easy to learn, facilitating communication between stakeholders with and without software development technical skills. Compared to other formal modeling tools, the UITD has several advantages including its simplicity, completeness, and ability to model user-system interactions in a clear and concise way. Empirical evidence about the UITD's understandability by non-technical stakeholders is also provided in [3]. By using the UITD, designers and developers can create models that are both technically accurate and easy to understand, helping to ensure that all stakeholders are aligned and satisfied with the specification of the system's requirements.

In recent years, various tools have been developed to support the creation of generalized diagrams. One such tool is draw.io, which is a well-known open-source diagramming software. Draw.io is widely used for creating diagrams, flowcharts, and other types of graphical representations. While draw.io is a versatile tool, it may not be specifically designed to handle the requirements of creating UITDs. This raises the question of how easy it is to create UITDs using draw.io, and whether it is more efficient to use a dedicated tool such as the UITD editor [4].

To investigate this question, we conducted a study in which participants were asked to create UITDs using both draw.io and the UITD editor. Our study aimed to demonstrate that the UITD editor is a useful tool for building User Interface Transition Diagrams. So, we compare draw.io as a benchmark with the UITD editor in terms of the time required to complete the task, as well as the subjective ease of use and satisfaction of the participants. We hypothesized that the UITD editor would be more efficient and preferred by participants, as it is specifically designed for creating UITDs and has specialized functionalities to simplify the editing of UITD properties. The results presented here confirm the above.

Initially, the primary purpose of the UITD editor is to expedite the process of creating UITDs for developers. Importantly, it should be noted that a subsequent phase of development envisions the UITD editor expanding its utility to encompass automatic code generation as well.

The results of our study have important implications for software developers, designers, and other stakeholders involved in the creation of systems that rely on user-system interactions.

The remainder of this paper is organized as follows: Section 2 contains a brief introduction to User Interface Transition Diagrams and a general description of the UITD editor features. In Section 3 we briefly describe the UITD Editor characteristics. In Section 4 we describe the experimental study. Section 5 has re-

sults. Section 6 has discussion. Finally, Section 7 conclusions and future work.

2. The User Interface Transition Diagrams

The UITD expresses requirements of a system regarding its User Interface (UI) transition triggers from a source UI to a destination UI [1]. It graphically captures all the actions that the user can perform in the UI and the corresponding change of UI, if any. In a UITD, interfaces have a text name and a number for their easy identification during design and subsequent stages. Transitions have an origin and a destination user interface. Each transition has a label. Labels in transitions are composed of one user action and, when two or more transitions have the same user action, the label also contains information about the conditions that need to be met to trigger the transition.

In the UITD, each transition is labeled with a specific format: User action/<Condition>, where the user action refers to the action taken by the user in the UI, and the Condition is an optional additional piece of information that specifies any conditions that need to be met for the transition to occur. For example, a transition labeled "Click on Submit button/Form fields are all filled" indicates that the user must click the "Submit" button and all form fields must be filled out for the transition to occur.

The example UITD in **Figure 1** illustrates the flow of user interfaces and available user actions for a cultural center website that provides information about concerts, theatre plays, and movies. The UITD shows two kinds of users: associated and visitor, and the different actions they can take on the site. The associated users are able to edit the pages to add or modify events.

The transitions between UIs represent the actions that can be taken by the user, such as clicking on a button or navigating to a different page. For example, the transition labeled "Click on Login button/User has valid credentials" represents the action of clicking on the "Login" button, and the condition that must be met for the transition to occur, namely that the user must have valid credentials.

One of the benefits of using UITDs is their modularity, allowing for the diagram to be divided into sub-diagrams, making it more manageable (constructability). To illustrate the modularity of UITDs, **Figure 2** shows a sub-diagram for the login subsystem. Here, one can see that the UI #1 "Home" is presented to all users upon login. When an associated user logs in by giving correct account and password, the UI #4 "Home (Associated)" is presented. Both, Associated and Visitor, can see the UI #2 "Menu". The full options for UI #2 are explained later with the UITD subdiagram in **Figure 3**.

It is possible for an associated to select the login option in UI #2 when he/she is already in a session. In this case, UI #20 will be displayed with the message: "your session will end, are you sure?". With the "no" option the user continues in his/her current session, while with the "yes" option, the UI #3 is presented in order to login again by providing an account and password.

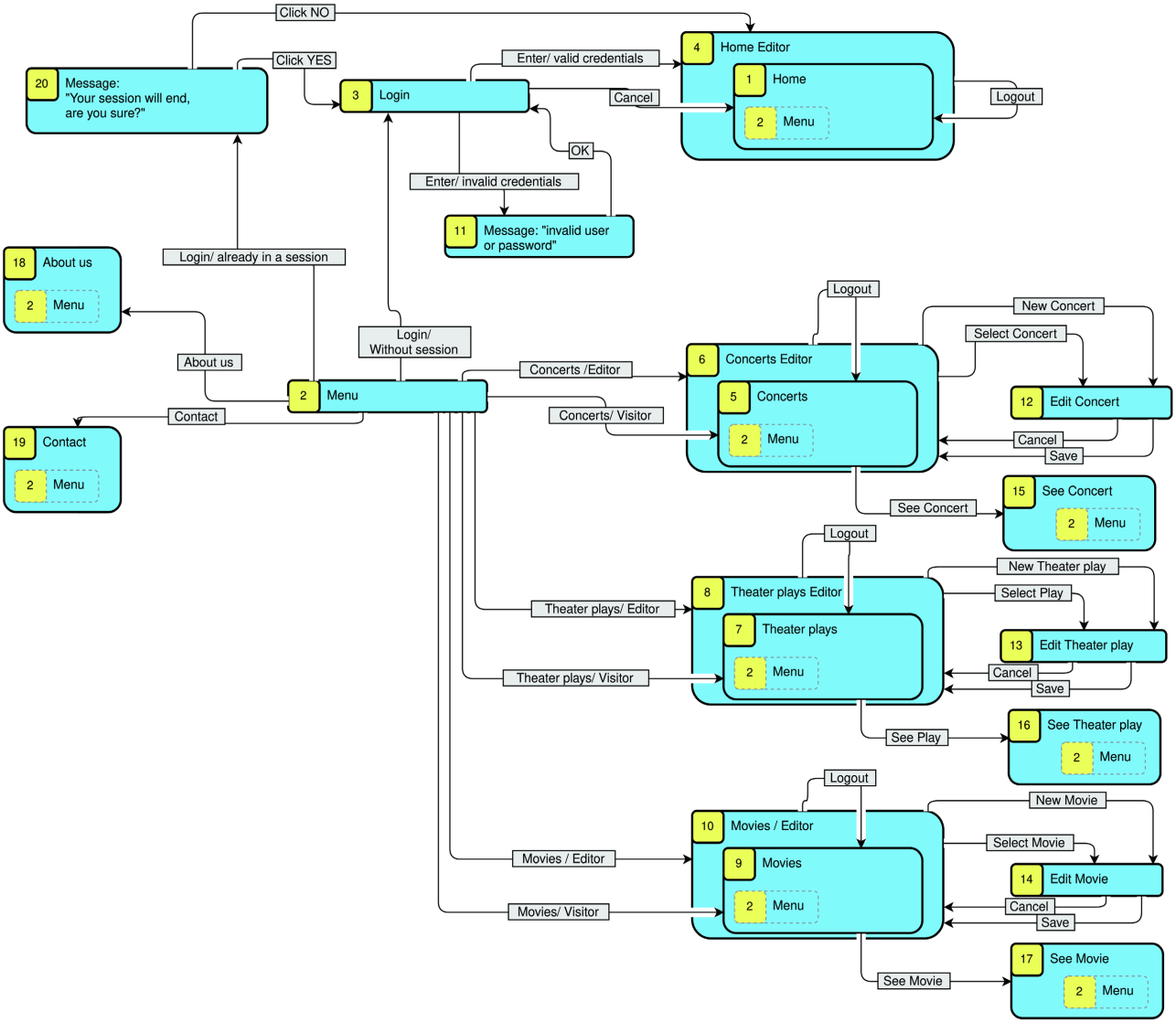

Figure 1. UITD of the site of a cultural center.

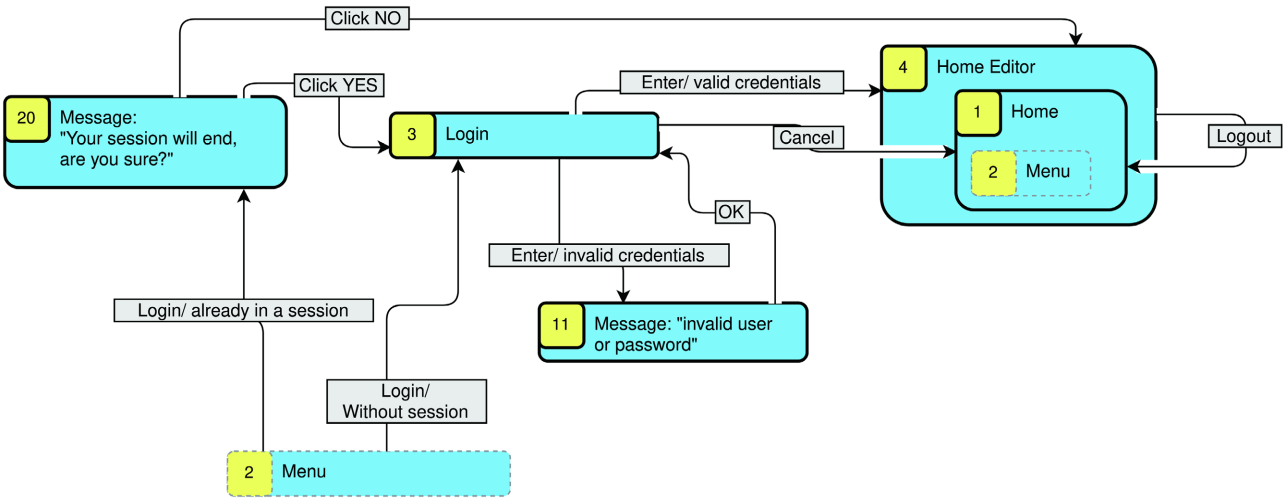

Figure 2. Login subsystem.

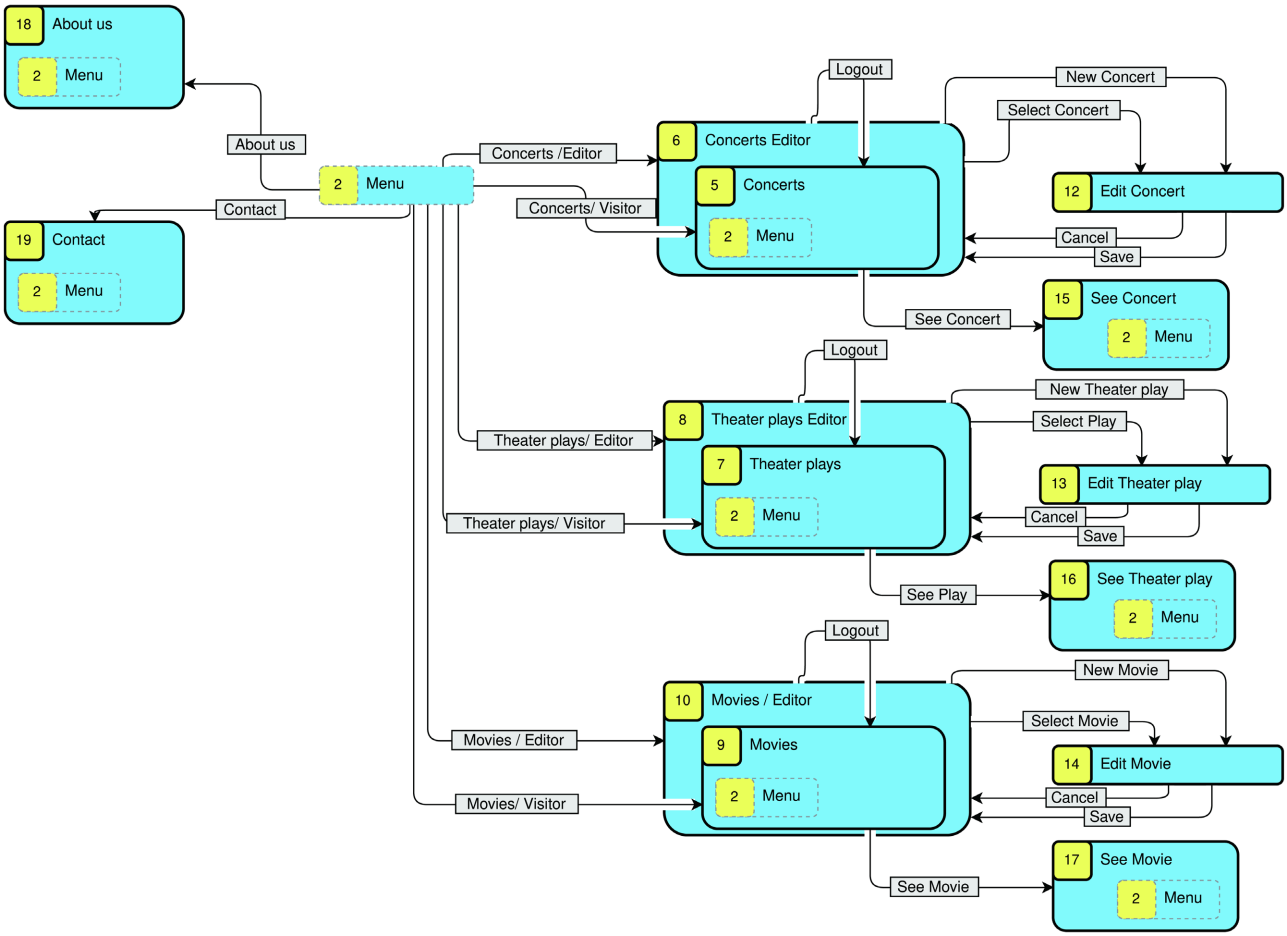

Figure 3. Site navigation with the Menu.

The User Interface Transition Diagram (UITD) editor includes a feature for managing complex diagrams called bolded and non-bolded User Interfaces. This feature allows for the diagram to be divided into several fragments, making it more manageable. Whenever a UI has its border in bold, it means that it is complete in the sense that all transitions connected to/from this UI are documented and visible in the current fragment. Conversely, when a UI does not have its borders in bold, it indicates that not all connected transitions are included in the fragment and consulting another fragment will be necessary to see them all. A UI can appear in multiple sub-diagrams, but it is recommended that all UIs appear in bold in at least one of the fragments. The full set of available transitions from a UI is the union of all transitions present in all fragments containing that UI. This feature simplifies the management of complex diagrams, allowing the user to focus on specific areas of the diagram as needed.

In the UITD of **Figure 3** (Main site), the UI #2 "Menu" options are shown except "login" and "logout", therefore its border is not in bold. The other UI #2 options are: "see concerts", "see theatre plays", "see movies", "contact information" and "about us".

Containing and Contained User Interfaces

The idea of containing and contained User Interfaces (UIs) is to place one UI inside another in the diagram, expressing that all transitions with origin in the contained UI are also available from the containing UI. Thus, the transition triggers that are available from the contained UI are a subset of those available from the containing UI. This feature allows the use of a UI inside several other UIs and to extend the functionality of a UI by placing it inside another one. Extending the functionality is useful when one wants to allow the extended functionality only in certain cases, such as a privileged user type.

Figure 3 illustrates an example of the containing and contained UI feature. The UI #6 "Concerts (Associated)" is a containing interface that includes the UI #5 "Concerts". The UI #5, in turn, contains the UI #2 "Menu". All the options in the UI #2 "Menu" are available from both UI #5 and UI #6.

However, visitors can only access UI #5, whereas associated users have access to UI #6, where they can add a new concert or select a concert to edit it. When doing so, the UI #12 "Edit concert" is displayed. The UI #8 "Theatre plays (Associated)" containing the UI #7 "Theatre plays", and the UI #10 "Movies (Associated)" containing the UI #9 "Movies" work the same way. When an associated user selects "logout" in UI #6, UI #8 or UI #10, the corresponding UI for visitors is displayed, that is UI #5, UI #7 or UI #9 respectively. The UI #18 "About us" and the UI #19 "Contact" are displayed the same way for associated users and for visitors.

3. The UITD Editor

The UITD editor is a User Interface Transition Diagram drawing tool designed to help software developers create models of user-system interactions with ease [4]. The tool is freely accessible through the following links:

http://148.206.168.145/EditorUITDEnglish/examples/indexF.html;
http://148.206.168.145/UITD/home.

The latter is a resource page containing additional information about UITDs. The editor provides a simple and intuitive interface and is continually being improved upon to enhance its functionality. It is a valuable resource for anyone involved in software development who needs to create user interface transition diagrams.

The UITD editor is built in JavaScript. It is based on mxGraph version 4.0.4 [5], which is an open source library created to draw diagrams.

The UITD editor is a versatile software tool that offers a variety of features to aid software developers in easily drawing a model of the user-system interactions with a UITD [4]. Some of its key features include the ability to draw a User Interface (UI) and label it with its name, with automatic generation of numeric identification. It also allows users to draw transitions and label them with the user action that triggers them and the necessary condition for that trigger. Additionally, the tool enables users to mark the border of a UI in bold to indicate that

all possible transitions to and from it are visible in the current fragment of the diagram. Other features of the UITD editor include the ability to place a UI inside an existing one (containing and contained UI), print diagrams (as a PDF or on a printer), access a user guide, and utilize a range of helpful examples.

4. Experimental Study

In this section, we describe an experimental study that aims to evaluate the efficiency of drawing UITDs using the UITD editor compared to an existing general graphic editing tool (draw.io). We followed the recommendations in [6] for preparing and reporting the study. The research question that will be answered based on our results is defined first. We then describe the selection of subjects, formulate our hypothesis, and explain the instrument design. Next, we provide an overview of the experimental procedure, followed by a detailed description of the analysis procedure.

4.1. Research Question

Our main research question is:

RQ: Is it faster to build a UITD with the UITD editor than with draw.io?

4.2. Selection of Subjects

To ensure the suitability of our study, we recruited volunteers from the computer engineering undergraduate program at the Autonomous Metropolitan University in Mexico City, all of whom had prior experience working with UITDs and were willing to draw UITDs using two different tools. We believe this selection of volunteers was appropriate for our study, as they were familiar with the concepts and techniques involved in creating UITDs.

The sample group had 62 subjects.

4.3. Experimental Procedure

Participants were instructed to create two UITDs, the login subsystem in **Figure 2** and the main site in **Figure 3**, using both the UITD editor and draw.io. These UITDs are fragments of the full UITD in **Figure 1**. Participants were asked to record the time it took them to create the diagrams using each tool. Draw.io was chosen for comparison as it is a popular and powerful charting tool with a free version available. Following the diagram creation task, participants were asked to complete the following questionnaire.

4.4. Instrument Design

The questionnaire that we use as an instrument is the following:

1) Which tool did you use first during the study?

a) UITD editor.

b) Draw.io.

2) Please report the time it took you to make the requested diagrams (without

reporting the seconds):

a) With the UITD editor

Diagram 1:

Diagram 2:

b) With Draw.io:

Diagram 1:

Diagram 2:

3) How much experience did you have with draw.io?

a) I did not know it previously

b) Very little experience

c) Little experience

d) Some experience

e) A lot of experience

4) Select the UITD Editor features that you consider to be an advantage compared to draw.io:

a) No advantage

b) Automatic drawing of User Interfaces (UI) with number and name

c) The automatic arrow labeling with condition/action

d) The drawing of a UI within another UI

e) The functionality to bold all UI borders

5) Do you have any suggestions for improvements to the UITD Editor?

4.5. Hypotheses Formulation

To determine whether there was a significant difference between the mean time spent drawing UITDs with the UITD editor versus draw.io, we utilized a dependent sample *t*-test (paired sample *t*-test) [7]. This statistical test was chosen because it compares the mean difference between two sets of paired observations. Each subject in our study was measured twice, resulting in pairs of observations. Specifically, each subject i was associated with the time spent using draw.io (t_{other_i}) and the time spent using the UITD editor ($t_{UITDeditor_i}$) to draw UITDs. We define the differences between two paired samples as:

$$d_i = t_{other_i} - t_{UITDeditor_i} \quad 1 \le i \le n \tag{1}$$

where

n is the sample size

The null hypothesis for the paired sample *t*-test assumes that there is no significant difference between the means of the two paired samples. In other words, the true mean difference μ_d between the paired samples is equal to zero. Thus, our null hypothesis can be stated as:

H_0: $\mu_d = 0$

The alternative hypothesis is that there is a significant difference between the means of the two paired samples. In this case, we are interested in determining whether the mean time spent using the UITD editor is lower than the mean time spent using the alternative tool. Therefore, the mean difference must be positive

and our alternative hypothesis is:

H_1: $\mu_d > 0$ (upper-tailed)

We want to determine with which of the hypothesis (null or alternative) the experimental data are more consistent.

4.6. Analysis Procedure

The test statistic for a paired sample t-test is given by Ec. (2)

$$t = \frac{\bar{d}}{\left(s_d / \sqrt{n}\right)} \tag{2}$$

where

$\bar{d}$ is the average of the differences d_i given by Ec. (1)

s_d is the sample standard deviation of the differences, given by Ec. (3)

$$s_d = \sqrt{\frac{\left(d_1 - \bar{d}\right)^2 + \left(d_2 - \bar{d}\right)^2 + \cdots + \left(d_n - \bar{d}\right)^2}{n-1}} \tag{3}$$

Now we state how the assumptions to perform hypothesis testing with paired sample t-test are met:

- *The dependent variable must be continuous*. In our study, the dependent variable is the time spent, which is a continuous variable.
- *The subjects must be independent.* The subjects made their diagrams individually, and their measurements did not affect each other.
- *Each pair of measurements must be obtained from the same subject*. We used the differences in the time spent by each subject to draw the diagrams.
- *The dependent variable should be approximately normally distributed.* We performed an Anderson-Darling normality test on the differences in the time spent, and the p-value was greater than 0.05, indicating that the data is not statistically different from a normal distribution. Therefore, the assumption of normality is met.

4.7. Validity Threats

Skill Bias: One potential threat to validity is the presence of bias resulting from varying levels of diagram-making skills among participants in different groups. To mitigate this, we employed a within-subjects design, requiring all participants to create diagrams using both the UITD editor and draw.io. By assessing each individual's performance on both tools, we minimized the impact of skill disparities and ensured a fair comparison. The use of a dependent sample t-test further enabled us to gauge the mean differences in performance on an individual basis.

Experience Bias: Another threat involves bias stemming from the experience gained by participants when creating the first diagram, potentially affecting their performance in subsequent attempts. To counter this, we implemented a counterbalancing approach. Half of the participants were instructed to begin with the UITD editor, while the other half began with draw.io. This approach equalized the impact of experience across both tools and allowed us to examine any varia-

tions introduced by the order of tool usage.

By addressing these potential threats, we aimed to enhance the internal validity of our study, ensuring that the observed differences in performance could be attributed to the inherent qualities of the tools rather than external factors.

4.8. Power of the Test (A Priori)

A power analysis was conducted using the G*Power 3 software [8]. We used the matched pairs t-test for the inequality of two dependent means, one tail.

For the a priori power analysis, the G*Power software indicated that a sample size of 45 is needed to achieve a statistical power of 0.95, while a sample size of 90 is needed to achieve a power of 0.99.

5. Results

5.1. The Two Parts of the Study

In order to achieve a sufficient number of subjects, that is, the number indicated in the a priori power of the test, our study was divided into two parts.

First part of the study. In the first group, 34 subjects participated, and we obtained the following results:

- For UITD 1 (**Figure 2** Login), the t-statistic value was 3.009206, and the p-value was 0.002494.
- For UITD 2 (**Figure 3** Main site), the t-statistic value was 1.911171, and the p-value was 0.032355.

Since both p-values for the paired sample t-test were less than the standard significance level of 0.05, we can reject the null hypothesis H_0.

During this part of the study, a significant finding was that 50% of the participants suggested the incorporation of shortcuts such as ctr-c/ctrl-v for copy/paste and ctrl-z/ctrl-y for undo/redo. Based on this feedback, we decided to include these requested features in the tool before continuing with the subsequent part of the study.

Second part of the study. The results of the second group, which consisted of 28 individuals, are presented below:

- For UITD 1 (**Figure 2** Login), the t-statistic value was 2.020355, and the p-value was 0.026687.
- For UITD 2 (**Figure 3** Main site), the t-statistic value was 2.060819, and the p-value was 0.024538.

Since the p-values for the paired sample t-test were less than the standard significance level of 0.05, we reject the null hypothesis H_0.

According the a priori power analysis, 45 samples are needed to get a statistical power of 0.95 (see section 4.9). So, adding the samples of the first part of the study with those of the second, we obtained 62 samples, and results are the following:

- For UITD 1 (**Figure 2** Login), the t-statistic value was 3.614776 and the p-value was 0.000305.

- For UITD 2 (Figure 3 Main site), the t-statistic value was 2.758436 and the p-value was.003828.

5.2. Power of the Test (Post Hoc)

We used the matched pairs t-test for the inequality of two dependent means, one tail in the G*Power software. For the post hoc power analysis, we calculated the required parameters following the guidelines in [8].

For a sample size of 62, we obtained that, for UITD 1 (Figure 2 Login), the achieved statistical power was 0.9731597. And, for the UITD 2 (Figure 3 Main site), the achieved statistical power was 0.8605469.

5.3. Opinions about the Advantages of the UITD Editor

None of the participants stated that they had not found advantages with the UITD Editor, 81% stated that the automatic drawing of User Interfaces (UI) with number and name is an advantage of the UITD Editor, 58% found as an advantage the automatic arrow labeling with condition/action, 68% stated the drawing of a UI within another UI as an advantage, and 44% marked the functionality to bold all UI borders as an advantage. This is reported in Table 1.

6. Discussion

Now we provide a thorough analysis of the study's findings, addressing their significance and implications for user interface design and modeling. We also highlight the strengths and limitations of the study and suggest potential directions for future research.

To demonstrate the effectiveness of the UITD editor, we aimed to reject the null hypothesis (H0) and accept the alternative hypothesis (H1) with a significance level of 0.05. H1 suggests that the average time spent creating a User Interface Transition Diagram using the UITD editor is less than the average time spent using draw.io. The results of the paired sample t-test confirmed this hypothesis, as the p-values were found to be below the standard significance level of 0.05, leading us to reject H0.

Furthermore, our a priori power analysis indicated that a sample size of 45 was sufficient to achieve a statistical power of 0.95. The post hoc power analysis provided insights into the performance of the UITD editor compared to draw.io. With 62 samples, we can be 97% confident that the UITD editor is faster for

Table 1. UITD Editor advantages.

UIT editor traits				
No advantage	Automatic drawing of User Interfaces (UI) with number and name	Automatic arrow labeling with condition/action	drawing of a UI within another UI	functionality to bold all UI borders
0%	81%	58%	60%	44%

creating the smaller diagram (UITD 1) and 86% confident that it is faster for creating the larger diagram (UITD 2) than using draw.io.

The study derives its strength from meticulous methodological design that adheres to established protocols and data analysis techniques. Employing a framework that involves individual participants being compared to themselves enhances result reliability by mitigating variations in skills and prior experiences. Furthermore, incorporating participant feedback and iteratively refining the tool highlights its adaptability to cater to users' requirements, thus enhancing its practical utility.

7. Conclusions

Our study demonstrated that the UITD editor outperforms draw.io in terms of efficiency and accuracy when drawing User Interface Transition Diagrams. The incorporation of recommended shortcuts, such as ctr-c/ctrl-v for copy/paste and ctrl-z/ctrl-y for undo/redo, further enhanced the usability of the editor. While the comparison was limited to draw.io, we believe that the specialized features of the UITD editor would provide it with an advantage over other general-purpose drawing tools.

It is worth noting that the UITD editor is part of a larger toolset, Architector, which we are developing to automate the process of generating web application skeletons with interface navigation based on UITDs. In future work, we plan to explore the benefits of Architector and publish it as a tool for developers.

In summary, our findings suggest that the UITD editor is a valuable tool for developers seeking to model the flow of user interfaces efficiently and accurately.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Gómez, M.C. and Cervantes, J. (2013) User Interface Transition Diagrams for Customer-Developer Communication Improvement in Software Development Projects. *Journal of Systems and Software*, **86**, 2394-2410. https://doi.org/10.1016/j.jss.2013.04.022

[2] Van der Linden, D., Hadar, I. and Zamansky, A. (2019) What Practitioners Really Want: Requirements for Visual Notations in Conceptual Modeling. *Software & Systems Modeling*, **18**, 1813-1831. https://doi.org/10.1007/s10270-018-0667-4

[3] Cervantes-Ojeda, J., Gómez-Fuentes, M. and Chacón-Acosta, G. (2022) Can Non-Developers Learn a Simplified Modeling Notation Quickly? *Journal of Software: Evolution and Process*, **34**, e2481. https://doi.org/10.1002/smr.2481

[4] Cervantes-Ojeda J, Badillo-Salas, A. and Gómez-Fuentes, M.C. (2021) Specialized Tool for Editing User Interface Transitions Diagrams (UITD). 2021 9*th International Conference in Software Engineering Research and Innovation* (CONISOFT), San Diego, USA, 25-29 October 2021, 10-16. https://doi.org/10.1109/CONISOFT52520.2021.00014

[5] MxGraph. https://jgraph.github.io/mxgraph/

[6] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B. and Wesslén, A. (2012) Experimentation in Software Engineering. Springer Science & Business Media, Berlin. https://doi.org/10.1007/978-3-642-29044-2

[7] Ross, A. and Willson, V. L. (2017) Paired Samples T-Test. In Ross, A., ed., *Basic and Advanced statistical Tests*, Brill, Leiden, 17-19. https://brill.com/display/book/9789463510868/BP000005.xml

[8] Faul, F., Erdfelder, E., Lang, A.G., Buchner, A. (2007) G* Power 3: A Flexible Statistical Power Analysis Program for the Social, Behavioral, and Biomedical Sciences. *Behavior Research Methods*, **39**, 175-191. https://doi.org/10.3758/BF03193146