An Experimental Comparison of Two Navigation Techniques for Requirements Modeling Tools

Parisa Ghazi, Martin Glinz
Department of Informatics, University of Zurich, Switzerland
{ghazi, glinz}@ifi.uzh.ch

***Abstract*—In Requirements Engineering, many modeling tasks require viewing different parts of a model concurrently. However, traditional zoom+scroll navigation uses a single focus zoom, i.e., at a given point in time, a user can zoom in on a single spot in the model only. Therefore, new focus+context navigation techniques have been proposed that allow multiple foci at the same time.**

In this paper, we report on an experiment with students where we compare the participants' performance when using a requirements modeling tool with traditional zoom+scroll navigation vs. one with so-called FlexiView navigation which is a focus+context technique with multiple foci. The participants had to perform typical modeling tasks such as searching, editing, and traversing a model. All tasks were performed on medium-sized tablets with a tool for manipulating so-called ImitGraphs. ImitGraphs are enriched node-and-edge diagrams that can mimic various diagram types such as class, activity, or goal decomposition diagrams. We found that navigation with FlexiView outperformed zoom+scroll navigation with respect to task completion time, number of mistakes, cognitive load, and user satisfaction.

***Index Terms*—Requirements engineering, Modeling tools, Navigation techniques, FlexiView, Zooming, User Interface, Empirical study, Experiment**

I. Introduction

In the process of Requirements Engineering (RE), requirements engineers create various types of requirements artifacts, e.g., textual specifications, glossaries, charts, and models [1] [2]. Despite the popularity of natural language for expressing requirements, graphical requirements models play a significant role in RE and requirements engineers spend a noticeable amount of time working with modeling tools [3]. In addition to the functional features of a tool, its user interface (UI) features influence the usability factors such as efficiency, effectiveness, and satisfaction [4] [5]. A particular UI problem is how to deal with models that are larger than the available screen. This frequently happens even with large display screens and means that navigation mechanisms such as scrolling and zooming are needed. When working with small screen devices such as tablets, navigation in the model becomes a major challenge [6].

Navigation with classic zooming and scrolling has been criticized for losing context when zooming in and its inability to view more than one part of the model in detail concurrently. To address this problem, so-called focus+context navigation techniques have been proposed [7] [8] [9] [10]. These techniques provide zooming with multiple foci and preserve the context of the zoomed-in regions by displaying it with less detail. However, the effects of using novel navigation techniques on the performance of users when carrying out typical requirements modeling tasks have not been studied empirically so far.

In this paper, we report on a study where we compared the performance of two navigation techniques in an experiment with students. For our comparison, we chose a novel focus+context technique designed for requirements artifacts called FlexiView [10] and a traditional navigation technique composed of zooming and scrolling. FlexiView provides multiple foci zooming based on a physical metaphor of magnets and springs. It uses a canvas of fixed size, so there is no need for scrolling.

Model navigation is a problem that occurs in any graphical modeling language, not just in RE. We performed our experiment in an RE context due to the importance of modeling in RE and because we eventually wanted to know whether requirements engineers can perform modeling tasks faster and with fewer errors when using a novel navigation technique such as FlexiView than when working with a traditional one.

We found in our experiment that FlexiView significantly outperformed traditional navigation with zooming and scrolling with respect to task completion time, number of errors made and overall satisfaction of users. We also found that using FlexiView reduced the cognitive load of the experiment participants when performing memory-intensive tasks and that they considered FlexiView to be easy to learn.

The remainder of this paper is organized as follows. We review navigation techniques in Sect II. Sect. III describes the goals of our study and our research questions. In sections IV and V, we present the language, tool, and modeling tasks used, while Sect. VI describes the experiment. In Sect. VII we present and discuss our results. Sect. VIII deals with the threats to validity and Sect. IX concludes.

II. Navigation techniques

Graphical requirements models are typically larger than the available screen [6]. The user interfaces (UIs) of tools for editing and viewing such models, therefore, need to provide mechanisms for navigating in large models. Also, the UIs of requirements modeling tools should provide a mechanism for viewing different parts of a model in detail concurrently, as requirements engineers frequently need this when editing or analyzing a model. Cockburn et al. [8] reviewed different navigation techniques and categorized them. In this section, we briefly characterize typical navigation mechanisms for graphical requirements models.

2332-6441/18/$31.00 ©2018 IEEE
DOI 10.1109/RE.2018.00032

RE 2018, Banff, Canada
Research Paper

A. Explosive Zooming with Multiple Windows

The early graphical requirements modeling tools (e.g., Software Through Pictures or Teamwork) used multiple windows for navigating in hierarchically structured models. For example, when users wanted to view the details of an activity in a dataflow diagram, they clicked that activity and the tool displayed the underlying dataflow diagram or textual spec in a new window. Scrolling was used when a diagram was larger than the window displaying it. This so-called explosive zooming is easy to implement and use. It also provides a straightforward way of viewing details of different parts of a model concurrently by arranging the respective windows side by side. The big disadvantage is that users are quickly lost in a pile of overlapping windows. Also, on mobile devices with small screens, multiple windows are not equally usable.

B. Zooming and Scrolling

Navigating in a graphical structure by proportionally enlarging (zooming-in) or shrinking (zooming-out) the whole structure around a focus point, is a proven navigation technique which has its origin in graphics drawing tools. The focus point can be the center of the screen or a selected model element. Proportional zooming-in enlarges the drawing canvas, which then may become larger than the available display device. Hence proportional zooming must be combined with a scrolling mechanism. In the remainder of this paper, we speak of *ZoomScroll* when referring to the combination of proportional zooming and scrolling. ZoomScroll is today's standard technique for navigating in graphics, images, maps, etc. on smartphones and tablets. Using two-finger gestures, it is very easy and straightforward to use. However, ZoomScroll is a single focus technique, i.e., it does not allow to zoom in on more than one part of the model concurrently.

C. Focus+Context

To overcome the shortcomings of explosive zooming (loss of context) and proportional zooming (single focus only), so-called *focus+context* techniques have been developed for navigation in models [8]. The goal is to provide zooming-in on details with multiple foci, while preserving the full context around the model elements which are currently zoomed-in. The idea is to visualize the focused parts of the model with high detail while the surrounding context is visualized without any details. Furnas [7] developed fisheye views, the first focus+context visualization technique, in 1986. In Requirements Engineering, Onion graphs [11] have been proposed as a focus+context technique for visualizing large UML models. This technique visualizes the focus area with UML notation and the rest with their own notation. In ADORA [9], zooming-in works by enlarging the focused object (thus creating space for displaying information contained in that object) and pushing away all other objects that would overlap with the enlarged object otherwise. However, with every zooming-in operation, the model canvas becomes larger, so this technique also must be combined with scrolling and does not work well for small display screens. ADORA supports zooming with multiple foci. FlexiView [10] is a focus+context technique for visualizing a set of connected artifacts. FlexiView is based on a physical metaphor of magnets and springs [12] [13], using a canvas of fixed size. When a user zooms in on some region of the canvas, this is interpreted as applying a magnetic force at this point which pushes the surrounding regions towards the borders of the canvas and, as the canvas size is fixed, shrinks them. Thus FlexiView does not need scrolling. The disadvantage is that zooming in FlexiView geometrically distorts the model. The advantage is that an overview of the artifact always exists on the screen in which the relative positions of the components are preserved. FlexiView also supports zooming with multiple foci. Figure 1 shows how zooming in FlexiView works.

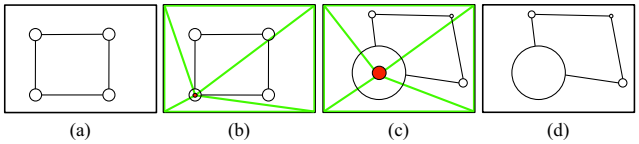


Fig. 1. FlexiView's magnifying process: (a) A graph. (b) A magnet is put on the lower left node. The green lines (not visible to users) show how FlexiView partitions the space. (c) The power of the magnet is increased, causing the focused node to enlarge and other nodes to shrink. (d) How the magnified graph is actually displayed (without the green lines).

D. Semantic Zooming

When zooming-in not just enlarges the focused area but uses the enlarged space to display additional information, we call this *semantic zooming* [14]. In ADORA, for example, zooming-in into an object results in displaying sub-models contained in that object. In FlexiView, the subject of semantic zooming can be a whole artifact or a component of an artifact. In the case of a whole artifact, semantic zooming reveals the constituents of the artifact and in the case of a component, semantic zooming reveals more details of the component.

E. Comparing navigation techniques

There is a limited number of studies on navigation techniques of requirements modeling tools such as the mentioned ADORA navigation technique [9], Onion graphs for UML diagrams [11] and hierarchical visualization of artifacts in automatic tracing tools [15]. To our best knowledge, there exists no empirical comparison of the effects of different navigation techniques on the performance of carrying out typical modeling tasks. Such a comparison is inherently difficult because navigation techniques typically come with specific tools, supporting specific modeling languages. So when comparing them, it is almost impossible to determine whether the observed effects are caused by the navigation techniques or by other differences in the tools and languages.

For our study, we decided to compare a classic navigation technique such as ZoomScroll with a focus+context navigation technique. For the latter, we chose FlexiView, as FlexiView does not use scrolling and works well on tablet devices which we wanted to use. To avoid the comparison problems mentioned above, we used a generic graphical modeling

language and created a tool for editing and viewing models that implements both ZoomScroll and FlexiView navigation.

III. Study Goals

The main goal of this study is to compare two navigation techniques for graphical requirements models with respect to the performance of users when carrying out typical modeling tasks. We particularly study whether a focus+context navigation technique such as FlexiView outperforms a classic navigation technique such as ZoomScroll. We compare these two navigation techniques in terms of efficiency and effectiveness. In addition, we want to get insight into how users rely on their memory for search path planning and working with parts of artifacts that cannot be accommodated in a single view on the screen. We framed our study goal in three research questions:

RQ1 *Can a focus+context navigation technique improve the efficiency and effectiveness of working with requirements artifacts compared to zooming and scrolling?*

RQ2 *Can a focus+context navigation technique decrease the cognitive load of working with requirements artifacts compared to zooming and scrolling?*

RQ3 *Will requirements engineers be more satisfied with using a focus+context navigation technique compared to zooming and scrolling?*

With RQ1 we aim at collecting quantitative empirical evidence, testing the hypothesis that focus+context navigation with FlexiView outperforms classic ZoomScroll navigation. In RQ2, we study how much a focus+context navigation technique influences the way users rely on their memory in typical RE tasks. RQ3 investigates the users' attitude towards using a navigation technique which is different from the ZoomScroll navigation that they use in their daily digital lives.

For answering our research questions, we designed and conducted an experiment with 24 students. A crucial prerequisite for such an experiment is the availability of a tool that supports a representative modeling language and provides both FlexiView and ZoomScroll navigation. The language and tool used in our experiment are described in the next section. Another prerequisite is the definition of modeling tasks to be carried out in the experiment. We present the four tasks we designed in Section V. Then, we present the details about our experiment in Section VI.

IV. Language and Tool

A. ImitGraphs as a generic modeling language

In our study, we used ImitGraphs [16], a generic modeling language that can mimic widely used modeling languages such as UML class or activity diagrams. Their simple appearance makes the development of experimental tools faster and their ability to imitate multiple graphical models ensures generalizable results. Using ImitGraphs drastically simplified the development of the tool required for our experiment and at the same time allowed us to carry out typical requirements modeling tasks on a model representing three types of models.

ImitGraphs are composed of three types of components: nodes, connections, and joints. Figure 2a shows a sample node. Figure 2b shows a sample connection which is composed of two joints (one at each end) and a central rectangular piece.

Different types of joints, connections and nodes should be defined for an experiment. This includes specifying properties such as orientation of the connections (vertical, horizontal or any), the connection types each node accepts, the joint types of each connection, and the anchor point from which a connection can be attached to a node (left, right, top, bottom, any combination of these or any). ImitGraphs distinguish types by color. A proper definition of components allows ImitGraphs to imitate a graphical model. Figures 2c and 2d show a simple activity diagram and its ImitGraph equivalent. Larger elements of the graphical models are simulated by multiple ImitGraph nodes (e.g., fork and join elements of the activity diagram).

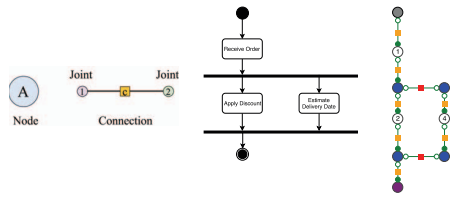


Fig. 2. ImitGraph notation: (a) a node. (b) a connection with two joints. (c) a sample activity diagram. (d) an ImitGraph equivalent to the activity diagram.

B. Tool implementation

Tablets are becoming increasingly popular as devices for creating and viewing requirements models: they are mobile, can be used in the stakeholders' work context [17] and can also be connected for collaborative modeling [18]. However, the smaller the display screen, the more challenging navigation in models becomes [6]. Therefore we decided to target medium-sized tablets (with a 10-inch screen) in order to investigate the extent to which the navigation technique used influences the performance of users when modeling requirements.

Based on the goal of this study, we specified the following requirements for our tool. (i) Supports creating and editing ImitGraphs. (ii) Supports both classic ZoomScroll and FlexiView for navigation. (iii) Is responsive and highly interactive so that it does not affect the performance of the users. (iv) Allows integration of a physics engine in order to achieve the realism we needed. (v) Is able to show smooth animations. (vi) Is able to access low-level interaction functions to record fine details of user interactions. (vii) Has a user interface similar to common modeling tools.

We decided to use the LibGDX game engine [19] for developing our tool to have adequate performance, responsiveness and interaction functions. Although we did not seek high accuracy in our physics simulation, we used the Box2D physics engine [20] which helped us to achieve realistic movements of FlexiView's magnets and springs. Accessing low-level interaction functions allowed us to precisely record detailed interactions of the users from within the tool. The output dataset contained single entries for actions such as

selecting a node and multiple entries for continuous actions such as dragging a node, both with millisecond precision.

In order to implement semantic zooming, we defined ImitGraphs to hold three different textual values named title, abstract and description in each node. Based on the size of the node on the screen and according to our defined thresholds the tool displays (i) first character of the title, (ii) title, (iii) title+abstract, or (iv) title+abstract+description.

Figure 3 shows the user interface of our tool. The central part is the graph editing area. The tool provides both Zoom-Scroll and FlexiView navigation in this area. For the purpose of our experiment, we included a hidden option to selectively disable one of them. When a node or connection is selected, its context menu appears above it. From the context menu, the users can delete the selected object, open a text editing window and create a magnet. When a node is magnetized, a slider appears at the bottom for setting the magnet's strength. For each magnet, a button is displayed in the magnet pane so that the user knows how many magnets exist and can select one of them easily by pushing its corresponding button. From the node menu and connection menu, the user can drag a node or connection and drop it on the editing area. The file menu contains buttons for creating/opening/saving files, undo and redo. The task menu contains the button that supervisors use for loading tasks, a timer and a finish button that users push when they finish the task.

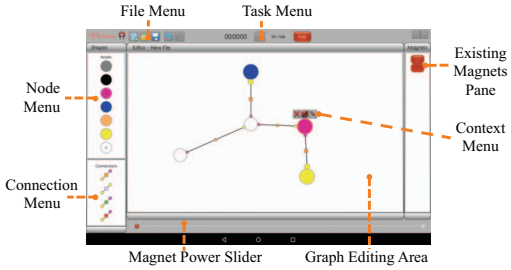


Fig. 3. User interface of the tool used in the experiment.

V. Experiment Tasks

We designed the experiment tasks based on four common operations that are regularly performed when working with RE artifacts [2]: (i) Searching for specific information, (ii) Editing an artifact using information from other parts of the same artifact, (iii) Searching in a hierarchical structure, (iv) Searching and matching information stored in more than one artifact.

In addition, we considered two criteria. First, the tasks need to be challenging enough so that we can observe time and error differences. Second, the tasks have to be doable within the limited time frame of a trial. The latter limits the size of the models used in our experiment. Based on these considerations we designed the following four tasks.

A. Task Search

In this task, the participants are given an ImitGraph that mimics a class diagram, i.e., nodes represent classes and edges

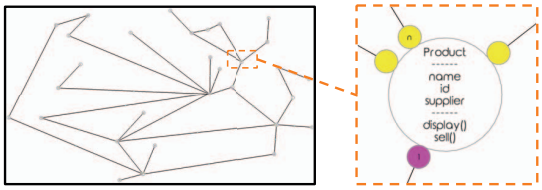


Fig. 4. The ImitGraph used in task Search, representing a class diagram. The magnified part shows a node with the class name, attributes, and functions.

represent associations. Figure 4 shows this graph and a magnified part of it. Each node holds a class specification consisting of a name, a list of attributes and a list of functions. These become visible when zooming into a node. The participants must find all nodes that contain an attribute named "id" (which is present in five out of 30 nodes) and add a function named "getId()" to these nodes.

In this task, we wanted to observe how the participants search the whole model and how efficient their search paths are with regard to missed nodes and repeated visits.

B. Task Recreate

In this task, the participants are given three ImitGraphs in one artifact (Figure 5). The left graph represents a given UML activity diagram. The right graph represents an activity diagram under construction, where the already existing parts have been copy-pasted from the left graph.

The participants must now complete the missing parts of the right graph such that (i) the two graphs are structurally identical and (ii) the names of the re-created nodes are transformed according to the rules specified by the ImitGraph in the middle. The transformation works as follows: when participants create a new node in the right graph, they have to determine the label n of the corresponding node in the original (left) graph. Then they have to determine whether there is a transformation rule for n, i.e., a connection in the middle ImitGraph that has a left node labeled n. If this is the case, the newly created node in the right graph gets the label n' which is given by the right node of the transformation rule. Otherwise, the new node is labeled n.

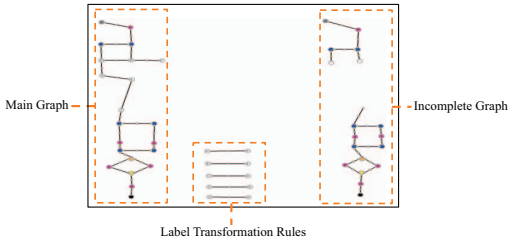


Fig. 5. The artifact that was used in task Recreate with its three ImitGraphs.

In this task, we wanted to observe how efficiently the participants navigate between three known locations of the artifact and to what extent the off-screen information that they need affects their performance.

C. Task Hierarchy

In this task, the participants are given an ImitGraph with an embedded hierarchical structure. This ImitGraph mimics a structure found in i* models, which may contain hierarchical goal decompositions, but also contain other links between the nodes of the model. In this ImitGraph, the edges expressing hierarchical structure are marked with a small green rectangle in the middle of the connection, while the other edges are labeled red or orange. The root node of the hierarchy that the participants had to explore is labeled *Start*.

Figure 6 shows the given graph and the embedded hierarchical structure. Hierarchy edges link the nodes on the first decomposition layer with the root node. Every node on the first decomposition layer may have hierarchy edges to nodes on the second decomposition layer, etc. The hierarchy is not visible in the spatial arrangement of the nodes, which is also typically the case in i* models.

The task of the participants is to traverse the hierarchy, starting from the root node, and delete all occurrences of the attribute *static* from the nodes belonging to the hierarchy, but not from any other nodes.

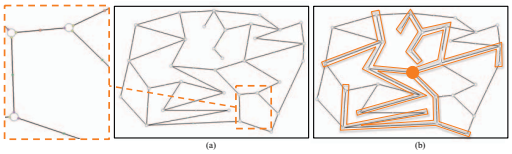


Fig. 6. (a) The ImitGraph given to the participants and an enlarged region. (b) The hierarchical structure is highlighted (not shown to the participants).

This task was designed to observe how participants traverse a hierarchical structure and how well they remember the relative location of nodes that they have already visited.

D. Task Matching

While we designed the first three tasks for comparing the performance of ZoomScroll vs. FlexiView navigation in a single artifact, this task was designed to explore the effects of FlexiView navigation in a single artifact vs. presenting the same information in multiple artifacts (with ZoomScroll navigation). For this, we defined two settings. In the first setting, the participants are given an ImitGraph with three central nodes which are connected to five secondary nodes each (Figure 7). Each central node contains six central codes and each of the secondary nodes contains one secondary code.

In the second setting, the participants are given four files. The first file contains the three central nodes. The second, third and fourth files contain the five secondary nodes that are connected to the first central node, second central node and third central node, respectively. The names of these files indicate to which central node they belong. As usual on tablets, the participants can open and view only one file at a time.

In both settings, the task of the participants is to inspect the linked pairs of central and secondary nodes and identify those that have matching codes. A match occurs when the first character of the code in a secondary node is equal to the last

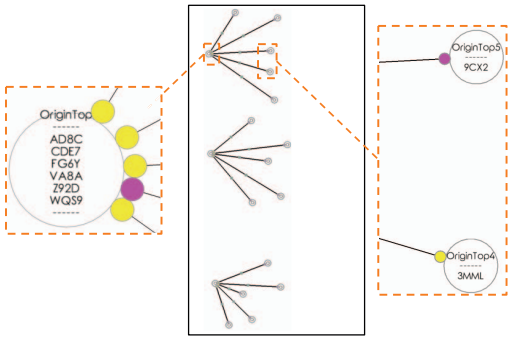


Fig. 7. The artifact used in the first setting of task Matching. A central node and two secondary nodes are magnified, showing the codes they contain.

character of one of the six codes in the corresponding central node. For example, in Figure 7, the code "9CX2" in secondary node *OriginTop5* matches the code "WQS9" in central node *OriginTop*. In this task, the participants were allowed to use a small piece of paper (1.5×5cm) for taking notes.

This task was designed to observe whether presenting related information in a single graphical model with semantic zooming as offered by FlexiView has benefits over presenting the same information in multiple related files.

VI. The Experiment

A. Participants

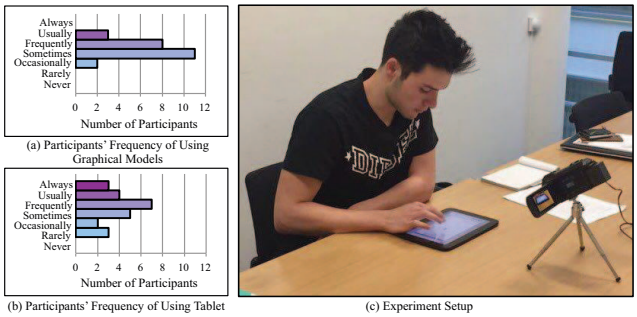
We had two criteria for recruiting the participants: (i) being familiar with graphical requirements models, (ii) knowing how to use a tablet. We particularly targeted students who had recently taken the Introduction to RE course offered at our university. Advanced knowledge in RE was no requirement.

We recruited 24 students (12 advanced undergraduates and 12 graduate students) from the student population in Computer Science at our university. Each participant received an honorarium equivalent to about $ 20 for participating in the experiment. The age of the subjects ranged from 21 to 44; the mean age was 26.70 years. Figures 8a and 8b show the participants' experience in using graphical models and tablets based on their claims. All had normal or corrected to normal vision and did not suffer from any form of color blindness.

B. Experiment Design

After designing the tasks, we performed two pilot studies with a postdoc and a PhD student who were knowledgeable in this field, but not involved in this project directly. We adjusted the task difficulties and time limits based on the pilot results, thus ensuring that all tasks were doable in the given time. We used time limits for motivating the participants to work as fast and productively as they would have to work on real tasks in industry. An on-screen timer showed the remaining time during the tasks. The participants were asked to finish the tasks even after the timer went off.

All experiment trials were carried out on a 10-inch Android tablet. The whole sessions were recorded with a video camera such that only the tablet screen and the hands of the



(a) Participants' Frequency of Using Graphical Models

(b) Participants' Frequency of Using Tablet

(c) Experiment Setup

Fig. 8. Experience of users in using (a) graphical models and (b) tablets, (c) Experiment Setup

participants were recorded. We used these videos only as a complement to the data recorded by our tool: they helped us resolve ambiguities and double-check special cases, e.g., when a search path crossed a node but the participant did not visit it. Figure 8c shows the experiment setup.

Each experiment trial comprised a tutorial session, four tasks, and a final survey. Each trial started with a short introduction of our tool followed by a training session. During the training session, we first made sure that the participants learned how to use the basic functions of the tool (e.g., creating nodes, connecting nodes and moving them). Then, we explained how FlexiView works and guided them to do basic operations (e.g., creating a virtual magnet, changing a magnet's strength and managing the magnets). We also explained how they could zoom and scroll. Next, we shortly explained the concept of ImitGraphs. At the end of the tutorial, we gave the students about five minutes to try out the tool. The mean tutorial time was 15 minutes and 20 seconds.

Each participant executed the first three tasks two times: one-time using ZoomScroll and the other time using FlexiView. They executed the fourth task, once using the MultiFile+ZoomScroll technique and once using FlexiView. We equally divided the participants into two groups: FlexiView-first and FlexiView-second. The order of the tasks was the same for both groups of participants, but the FlexiView-first participants executed each task first using FlexiView then the other technique; FlexiView-second participants executed each task first using the other technique and then FlexiView. We asked the participants to mark their satisfaction with the ease of use after each execution (i.e., two times for each task).

At the beginning of a new task, we provided the instructions about the task on a sheet of paper and gave the participants time to read and understand them. The supervisor answered their questions about the task. When they felt ready, the supervisor started the on-screen timer for the task and the participant had to finish it before the timer went off. When they were done, the participants pressed the *Finish* button in the menu bar. The participants were asked to finish their task even after the timer went off. After the participants finished all four tasks, we asked them to fill out the final survey. This survey contained eight statements in four groups: learnability, memory usage, performance and overall satisfaction.

The experiment was conducted in January and February 2018. Each trial lasted approximately one hour and 35 minutes. The descriptions of the tasks, the training material, and the satisfaction questionnaires were all in English[1]. We chose a quiet place for running the experiment and made sure that no disturbance occurred during the trials. All experiment trials were supervised by the first author to gather consistent data.

VII. Results and Discussion

In this section, we report our results of analyzing the gathered interaction data for the four tasks. Since we designed our tool to record user interactions in fine details, the resulting raw data set (more than 26 hours of recorded interactions) was so large that we could not analyze it without postprocessing. We developed a Java program to extract higher level data from the raw data, e.g., visited nodes, the speed of scrolling, states, and duration. Then, instead of the raw data, we analyzed the extracted data to find the information that we were looking for, e.g., missed nodes, repeated visits, and searching paths.

We used two types of charts for depicting our results: box plots and bar charts. Box plots provide a visual image of the distribution of the data, while bar charts allow comparing populations of different answers on a Likert scale. Since we repeatedly compare two datasets, one from FlexiView and one from ZoomScroll, we employed a paired two-tailed Student's t-test [21] to find if the differences are statistically significant. Since all t-tests were done with results from 24 participants, the degree of freedom was 23. Also, we set 0.05 as Alpha for all t-tests, i.e., we used p-values of 0.05 to determine if the differences were significant. On the right side of the box plots, mean and standard deviation of each execution and t-test results (t-value and p-value) for each task are presented. We refer to each execution of a task as Task+Technique, e.g., Search+ZoomScroll. We present our results in four categories: efficiency, effectiveness, cognitive load, and satisfaction. The first two categories answer RQ1. The third and fourth categories answer RQ2 and RQ3 respectively.

A. *Efficiency (RQ1)*

Completion time is the main factor that is taken into account for assessing the efficiency. Figure 9 shows the distribution of completion times for all tasks and techniques. In all tasks, the mean of completion time for ZoomScroll executions is higher than for FlexiView executions. In tasks Search, Recreate and Matching the difference is significant ($p < 0.05$). In task Hierarchy, the difference is not significant ($p > 0.05$).

To find out why FlexiView was not significantly better than ZoomScroll in task Hierarchy, we studied the videos of this task and found that with ZoomScroll, this task could be accomplished without much zooming in or out: the participants could zoom in on the root node once and then find their path without further zooming, using the hierarchical links as guides.

The navigation technique is one of the many parameters that affect the completion time. In our study, other parameters

[1]https://re18flexiview.page.link/jdF1

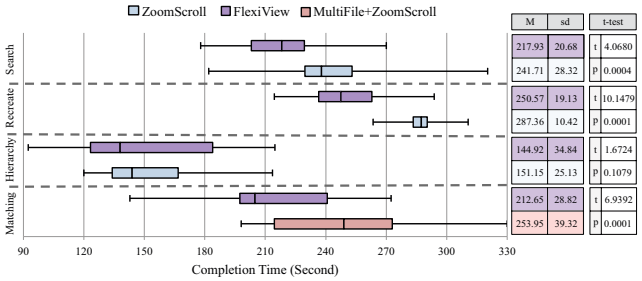


Fig. 9. Completion times for four tasks

(e.g., typing skill on a touch-screen keyboard) act as noise. We managed to reduce the noise for two tasks (Search and Recreate). First, we identified four states into which the participants switched in each task. Second, we classified each point of the timeline into one of the four states. Finally, we aggregated the times that the participants spent in each state.

We identified the following four states in task Search: (i) *Orientation*: In this state, users decide about the search path and the next node to visit. The characteristics of this state are that the zooming level is not high enough to check the details and the user does not scroll. (ii) *Navigation*: In this state, users know the next node they want to visit and try to reach it. The characteristic of this state is that users change the zoom level and/or scroll. (iii) *Process*: In this state, users check the content of the nodes and decide whether to change it or leave it. The characteristic of this state is that the zooming level is high enough to check the details of the nodes and users do not navigate. (iv) *Edit*: In this state, users open the node editor and change the content of the node.

In the process of classification, for deciding whether the user is scrolling or not, we defined a speed threshold. When the user moves faster than the threshold, we classified it as scrolling; Otherwise, we classified it as not scrolling. Figure 10 shows how much time users spent in each of the four states. Editing and processing times were generally in the same range for both techniques with no statistically significant difference. The main reason that participants completed this task faster in FlexiView executions lies in the differences in orientation and navigation times which are significantly different based on the t-test results.

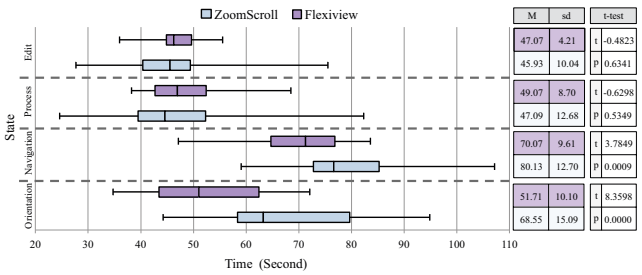


Fig. 10. Search Task: consumed time in each state

In task Recreate, unlike task Search, the participants knew the locations of the information they needed. They needed to constantly move between three sections of the canvas. The participants were in one of the following four states during the whole task: (i) processing the main graph: when they studied the structure and the content of the nodes of the main graph, (ii) processing the rules graph: when the participants checked the label transformation rules, (iii) navigation: the time that they spent to move the focus between the three sections, and (iv) creation: the time that they spent to create the required nodes and connections.

We identified navigation states similar to the previous task. Distinguishing the rest of the states was done based on the location of the focus. Figure 11 shows how much time the participants spent in each state. Among the four states, navigation time has the highest difference and was the main reason why the participants finished this task using FlexiView in a shorter time.

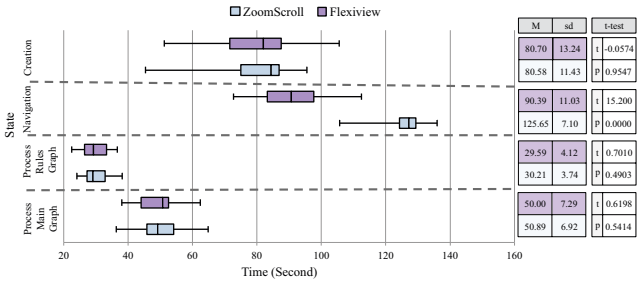


Fig. 11. Recreate task: consumed time in each state

In summary, in tasks Search and Recreate FlexiView affected the efficiency positively. In task Hierarchy, the results of FlexiView were comparable to ZoomScroll. The results of task Matching suggest that showing connected files as a graph and navigating it by FlexiView is more efficient than opening and viewing the files separately.

B. Effectiveness (RQ1)

In tasks Search, Recreate and Hierarchy, the number of errors that participants made in their final results was insignificant for both techniques. However, the participants made errors that did not affect their final results. For example, in task Search, some participants missed visiting some nodes. We call this type of errors navigation errors. Navigation errors could potentially result in errors in the final results. To compare the effectiveness of FlexiView and ZoomScroll, we took the navigation errors into account. Figure 12 shows the distribution of navigation errors for Search+ZoomScroll and Search+FlexiView. We defined three types of navigation errors for task Search: (i) the number of nodes that were missed to be visited, (ii) the number of repeated visits, and (iii) the number of times that the search path intersected itself. A node is counted as visited when the user is not zooming, the scrolling speed is below the threshold that we defined, and the semantic zooming level allows to see the content of the node. We drew a search path based on the sequence of the nodes that each user visited and counted the number of times this path intersected itself. This will be discussed in more detail

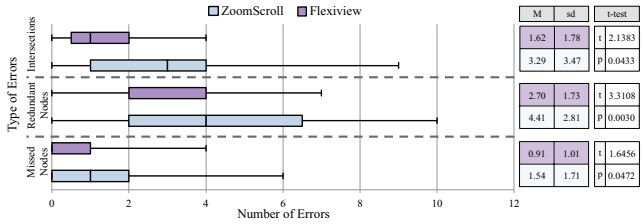


Fig. 12. Search task: the number of errors of different types

in Section VII-C. The mean numbers of errors for FlexiView executions are lower and differences between FlexiView and ZoomScroll are statistically significant ($p < 0.05$).

We defined three types of errors for task Hierarchy: (i) the number of nodes that were missed to be visited, (ii) the number of repeated visits, and (iii) the number of visited nodes that were not supposed to be visited. Figure 13 shows the distribution of these errors. FlexiView executions have a lower number of errors and the differences are statistically significant for missed nodes and wrong visits ($p < 0.05$). The p-value of redundant visits is slightly over the 0.05 threshold.

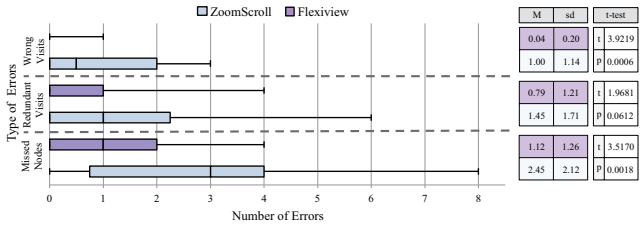


Fig. 13. Hierarchy task: the number of errors of different types

In task Recreate, the participants needed to visit the main graph and the label transformation rules graph multiple times. Some of these visits were necessary and some were not. The necessity of the visits depends on the participants' strategy and how much information they can keep in their mind. Distinguishing them is not trivial and we cannot make any conclusion about the difference of effectiveness in this task.

In task Matching, we counted the mistakes that participants made in the list of matched codes that they handed in. The errors comprise missed matches and wrong matches. Figure 14 shows the distribution of total errors for both techniques in tasks Search, Hierarchy, and Matching. In all these three tasks, the participants made fewer errors using FlexiView and the observed differences are statistically significant ($p < 0.05$).

In ZoomScroll, users have to zoom in when wanting to view details. This enlarges the model canvas, which means that some parts of the model disappear from the screen. In this situation, users most likely lose the overview of the artifact, their current position and the relative position of the nodes. This results in missing some nodes, redundant visits of some other ones and visiting nodes that are not needed to be visited. When using FlexiView, users can check the details while keeping an overview of the information on the screen. The cost of showing details and keeping an overview at the same time is distortion, which is caused by non-proportional scaling. Based

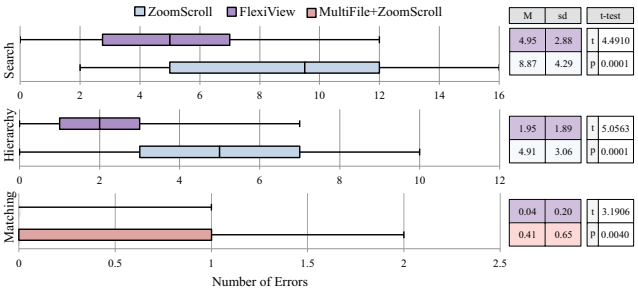


Fig. 14. Total number of errors for tasks Search, Hierarchy and Matching

on our results, despite the introduced distortion, FlexiView reduces the number of navigation errors.

C. Cognitive Load (RQ2)

When zoomed in, if the users need an overview of the artifact (e.g., for planning the search path), they either zoom out to have the whole artifact back on the screen or they rely on their memory. We were interested in knowing the characteristics of the search paths of the participants when using FlexiView and ZoomScroll in task Search. The path that participants traversed provides insight into how well they could plan based on the overview of the artifact they had on the screen and/or in their mind. Figure 15 shows the resulting search paths for two participants (p4 and p17). p4 was in the FlexiView-second group and p17 was in the FlexiView-first group. To compare the search paths we decided to use a simple measure: the number of times each search path intersected itself. Figure 12 shows the distribution of the number of intersections for ZoomScroll and FlexiView. The mean number of errors show that the search paths of FlexiView executions had fewer self-intersections. We can also visually observe that they were more regular and had sweeping patterns. The t-test results show that the difference is significant.

In task Matching, we gave small pieces of paper to the participants so that they could take notes if needed. Writing notes

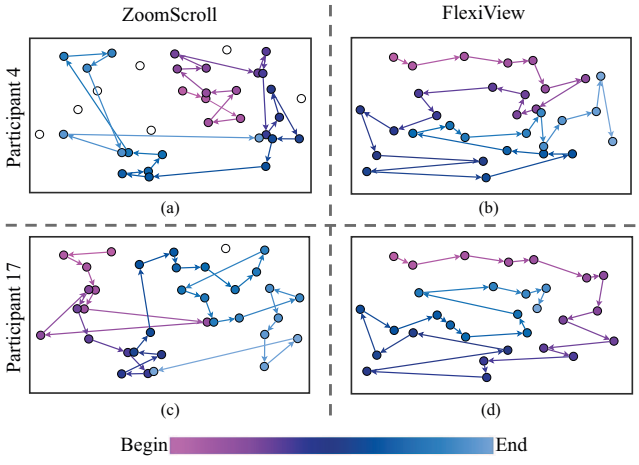


Fig. 15. Samples of search paths: (a) and (c) ZoomScroll, (b) and (d) FlexiView. The color shows the sequence and the white nodes were missed.

on these papers was an indicator of how much they needed to use their memory. We assumed that when the information that people need to keep in mind is above some threshold, they write more notes. We deliberately gave the participants a small piece of note paper only, so that they had to use their memory and could not write down all needed information. Figure 16 shows eight samples of note papers of four participants in MultiFile+ZoomScroll and FlexiView executions. To compare the note papers of the participants, we counted the number of letters they wrote on their MultiFile+ZoomScroll papers ($M = 27.5$, $std = 16.95$, $min = 18$, $max = 72$) and on FlexiView papers ($M = 2$, $std = 3.06$, $min = 0$, $max = 10$). The t-test results shows a significant difference ($t = 7.1618$, $p < 0.0001$).

Another observation in task Recreate was that 91.6% of the participants (22 out of 24) used multiple magnets to magnify the main graph and the rules graph at the same time to see more relevant data on the screen. Figure 17 shows a snapshot of such a situation. Although using more magnets causes more distortion, the participants exploited the advantage of having multiple foci.

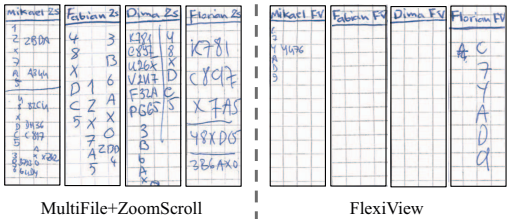


Fig. 16. Samples of note papers used by four participants when using FlexiView and MultiFile+ZoomScroll

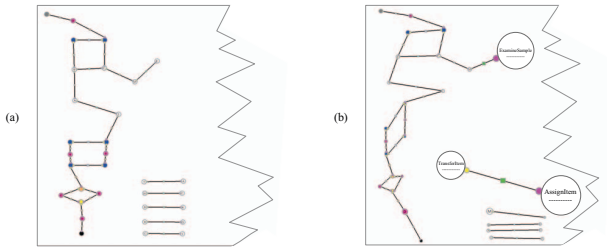


Fig. 17. Two partial snapshots of the artifact used in task Recreate: (a) normal and (b) with three magnets

D. Satisfaction (RQ3)

After each task execution, the participants rated their satisfaction with the ease of use of the navigation technique they had used for that task on a Likert scale. Figure 18 shows the result of these surveys. The first and second column show the results of ZoomScroll and FlexiView executions respectively. The third column shows the difference between ZoomScroll and FlexiView answers. For comparing the answers, we avoided quantification which could exaggerate or understate the difference. Instead, we compared the answers

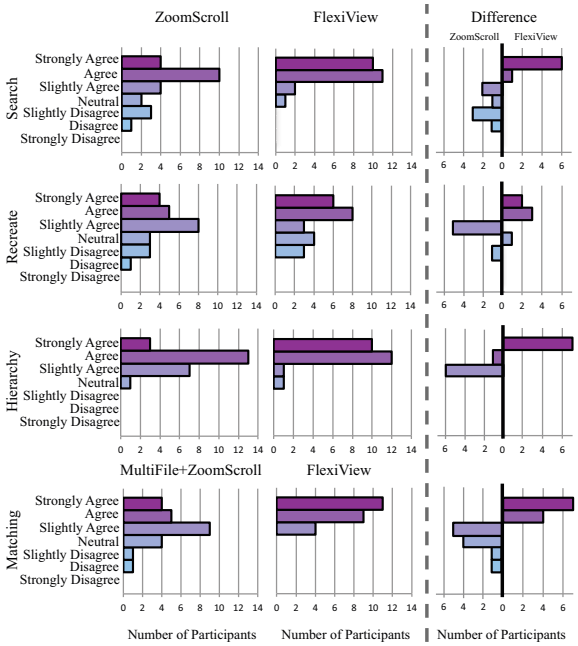


Fig. 18. Satisfaction with ease of use of the used technique after each execution. The third column depicts the difference.

separately by calculating the differences and showing them in a different bar chart.

Satisfaction is a complex feeling which is influenced by different factors. The purpose of the final survey was to collect more clues about this feeling. In this survey, we asked the participants to rate how much they agreed with eight statements in four different topics: learnability, memory usage, performance, and ease of use. Figure 19 presents the ratings for these eight statements, labeled S1-S8.

1) Learnability: In comparison to common navigation techniques used in commercial tools, FlexiView is more complex. However, we observed that all participants learned to use it during a short tutorial ($M = 15.33, min = 12.20, max = 21.75$) and developed their skill during the tasks. The result of the final survey confirms that the participants learned to use FlexiView quickly (S1) and believed that others would also learn it quickly (S2). Based on our observations, the participants were excited to see a physics-based user interface. The physics simulation made the transitions between distorted views of the artifact smooth and predictable. All participants successfully used FlexiView alone for navigating.

2) Memory Usage: We discussed some measurements of user cognitive load, e.g., memory usage. Since exact measurement of memory usage is not possible, we asked the participants how much overview they had while using FlexiView (S3) and how aware they were of their position (S4). The results show that FlexiView was successful in both.

3) Performance: Although we measured the performance of the participants based on the collected data, we still were interested to know how the participants felt about their performance. They agreed that they were more productive using

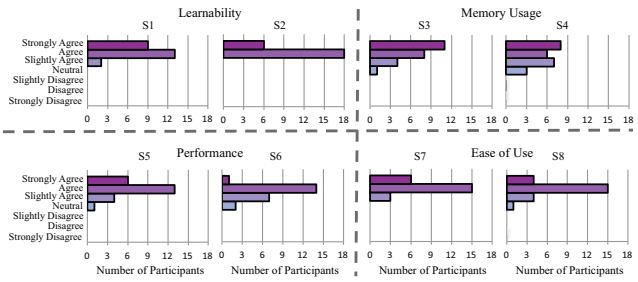


Fig. 19. Final survey results: S1: I learned to use FlexiView quickly. S2: Other people will learn to use FlexiView quickly. S3: Users have the overview of the artifact more often by using FlexiView. S4: Users are more aware of their position inside the artifact by using FlexiView. S5: Users become more productive by using FlexiView. S6: Users make fewer errors by using FlexiView. S7: I recommend FlexiView to be implemented in other tools. S8: I am satisfied with the ease of using FlexiView.

FlexiView (S5) and made fewer errors (S6).

4) Ease of Use: Finally, we wanted to know how satisfied the participants were with the ease of using the new navigation technique that they were trying for the first time. We asked if they were satisfied with the ease of use of FlexiView (S8) and if they recommend it to be implemented in RE modeling tools (S7). The high level of user satisfaction was an aggregated result of all the other aforementioned factors.

VIII. Threats to Validity

In this section, we discuss the usual four categories of threats to validity [22]. Measurements affect the *Conclusion Validity*. Although we recorded the interactions very precisely, the information extraction was based on thresholds that we defined (e.g., the speed threshold which distinguishes between visiting a node from hovering over it). We addressed this in two ways. First, we did the analysis with different thresholds and the extracted data was not sensitive to these thresholds. Second, we double-checked the special cases (e.g., missed nodes on the search path) with the recorded videos. In addition, in order to have equal motivation for finishing the tasks as soon as possible, we put a count-down timer on top of the screen. To have unbiased statements in the surveys, we avoided making direct comparisons between two navigation techniques. However, FlexiView being the subject of some of the questions may cause bias.

For *Internal Validity* we used different artifacts for ZoomScroll and FlexiView executions to remove the learning effect. The artifacts were basically similar with slightly different layouts and node contents. We made sure that the critical properties such as the number of nodes, number of answers, branching factor and the ratio of relevant/irrelevant information were the same. Additionally, we did not imply any expectation and used comparative questions only after all tasks were done to avoid any bias towards FlexiView. Furthermore, we compared the results of the FlexiView-first and FlexiView-second groups and did not find a significant difference.

The difference between ZoomScroll and FlexiView is a threat to *Construct Validity*. Zooming and scrolling have been matured during decades of use. The users of software tools have a great deal of experience in using them. FlexiView is new, more complex and immature. Since most of the results of this study are in favor of FlexiView, overcoming this threat (e.g., by improving the implementation and longer tutorial sessions) would only change the intensity of the results.

We recruited 24 students for our experiment which is a threat to *External Validity*. We addressed this in two ways. First, we used ImitGraphs instead of real requirements models to decrease the influence of previous knowledge on the results. Second, the tasks comprise basic common operations that are typically done on graphical requirements models. However, we do not claim that this study is a proof of how much FlexiView improves the effectiveness, efficiency and user satisfaction of requirements modeling tools. Nevertheless, this study shows a great potential for such a UI technique for improving requirements modeling tools. Using ImitGraphs introduces another threat. Although the behavior of the users when using ImitGraphs is similar to their behavior when using graphical models [23], it is still possible that the results of this experiment will be different when using RE graphical models instead of ImitGraphs.

IX. Conclusion and Future Work

In this study, we compared the effect of two navigation techniques on the performance of working with RE modeling tools: ZoomScroll, a traditional, widely used technique, and FlexiView, a new focus+context technique. The comparison was done using the precise interaction data recorded by our tool during the experiment designed for this study. For the experiment, we recruited students with some RE knowledge to perform basic operations that are usually done on RE models, once using ZoomScroll and another time using FlexiView.

Using FlexiView, the participants finished three tasks out of four in a shorter time and made fewer navigational errors in all tasks. They had a better overview of the artifacts when using FlexiView which resulted in better searching paths which in turn resulted in shorter completion time and fewer errors. We observed that the participants were excited about using the physical metaphors of FlexiView. They learned it quickly and expressed their high satisfaction with its ease of use. In conclusion, we discovered a great potential in a navigation technique such as FlexiView for improving the effectiveness, efficiency and user satisfaction of RE modeling tools.

This work is just a beginning of a path which ultimately will result in navigation techniques that are highly tailored to RE modeling needs. The next steps will be experimenting with various navigation techniques using our platform and comparing them with more complex baselines, focusing on the challenges of RE modeling tools individually, experimenting with more specific tasks including larger models, experimenting on devices with different screen sizes and incorporating modern human-computer interaction input methods.

Acknowledgement

We would like to thank all the participants for their time and valuable contributions that made this research possible.

REFERENCES

[1] O. Liskin, "How artifacts support and impede requirements communication," in *Proceedings of the 21st International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*, 2015, pp. 132–147.

[2] P. Ghazi and M. Glinz, "An exploratory study on user interaction challenges when handling interconnected requirements artifacts of various sizes," in *Proceeding of the 24th IEEE International Requirements Engineering Conference (RE' 16)*, 2016, pp. 76–85.

[3] J. M. C. De Gea, J. Nicolás, J. L. F. Alemán, A. Toval, C. Ebert, and A. Vizcaíno, "Requirements engineering tools: Capabilities, survey and assessment," *Information and Software Technology*, vol. 54, no. 10, pp. 1142–1157, 2012.

[4] E. Frøkjær, M. Hertzum, and K. Hornbæk, "Measuring usability: Are effectiveness, efficiency, and satisfaction really correlated?" in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '00)*. ACM, 2000, pp. 345–352.

[5] Intl. Std. Org. (ISO), *ISO 9241-11:2018: Ergonomics of human-system interaction – Part 11: Usability: Definitions and concepts*, 2018.

[6] P. Ghazi and M. Glinz, "Challenges of working with artifacts in requirements engineering and software engineering," *Requirements Engineering*, vol. 22, no. 3, pp. 359–385, 2017.

[7] G. W. Furnas, "Generalized fisheye views," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '86)*. ACM, 1986, pp. 16–23.

[8] A. Cockburn, A. Karlson, and B. B. Bederson, "A review of overview+detail, zooming, and focus+context interfaces," *ACM Computing Surveys (CSUR)*, vol. 41, no. 1, pp. 1–31, 2009.

[9] T. Reinhard, S. Meier, R. Stoiber, C. Cramer, and M. Glinz, "Tool support for the navigation in graphical models," in *Proceeding of the 30th International Conference on Software Engineering (ICSE '08)*. ACM, 2008, pp. 823–826.

[10] P. Ghazi, N. Seyff, and M. Glinz, "FlexiView: A magnet-based approach for visualizing requirements artifacts," in *Proceedings of the 21st International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*, 2015, pp. 262–269.

[11] H. Kagdi and J. I. Maletic, "Onion graphs for focus+context views of UML class diagrams," in *Proceeding of the 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2007)*. IEEE, 2007, pp. 80–87.

[12] J. M. Rzeszotarski and A. Kittur, "Kinetica: Naturalistic multi-touch data visualization," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, 2014, pp. 897–906.

[13] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Softw. Pract. Exper.*, vol. 21, no. 11, pp. 1129–1164, 1991.

[14] B. B. Bederson and J. D. Hollan, "Pad++: A zooming graphical interface for exploring alternate interface physics," in *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology (UIST)*, 1994, pp. 17–26.

[15] J. Cleland-Huang and R. Habrat, "Visual support in automated tracing," in *Second International Workshop on Requirements Engineering Visualization (REV 2007)*. IEEE, 2007.

[16] P. Ghazi and M. Glinz, "ImitGraphs: Towards faster usability tests of graphical model manipulation techniques," in *Proceedings of the 9th IEEE/ACM International Workshop on Modelling in Software Engineering (MiSE@ICSE 2017)*, 2017, pp. 61–67.

[17] N. A. M. Maiden, N. Seyff, P. Grünbacher, O. Otojare, and K. Mitteregger, "Determining stakeholder needs in the workplace: How mobile technologies can help," *IEEE Software*, vol. 24, no. 2, pp. 46–52, 2007.

[18] D. Wüest, N. Seyff, and M. Glinz, "Sketching and notation creation with FlexiSketch Team: Evaluating a new means for collaborative requirements elicitation," in *Proceeding of the 23rd IEEE International Requirements Engineering Conference (RE '15)*. IEEE, 2015, pp. 186–195.

[19] A. Oehlke, *Learning Libgdx Game Development*. Packt Publishing, 2013.

[20] I. Parberry, *Introduction to Game Physics with Box2D*, 1st ed. CRC Press, Inc., 2013.

[21] T. Tullis and W. Albert, *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics*, 2nd ed. USA: Morgan Kaufmann Publishers Inc., 2013.

[22] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, 2000.

[23] D. Lay, *Designing and Conducting Controlled Experiments on an Experimental Requirements Modeling Tool for Evaluating ImitGraphs*. Bachelor Thesis, Department of Informatics, University of Zurich, 2018.