TAMPERE UNIVERSITY OF TECHNOLOGY

AKSELI PALÉN

INTERACTION DESIGN SPACE OF GRAPH-BASED USER INTERFACES

Bachelor of Science Thesis

Examiner: Adj. Prof. Ossi Nykänen

Thesis submitted for review
on June 16, 2015

ABSTRACT

Akseli Palén: Interaction Design Space of Graph-Based User Interfaces
Tampere University of Technology
Bachelor of Science Thesis, 37 pages
June 2015
Information Technology
Major: Hypermedia
Examiner: Adj. Prof. Ossi Nykänen

Keywords: graph-based user interfaces, interaction, navigation, manipulation, collaboration, user interfaces

Graph-based user interfaces have been studied since the 1980s but mainly in the context of information visualization. Even though they have been applied to a wide range of applications from social network analysis to audio synthesizers, little is known about their interaction capabilities on a general level. In this study, by drawing from the human-computer interaction research of the last 30 years, we derive a technique and device independent interaction framework consisting of 4 main types and 12 subtypes of interaction on graph-based user interfaces. We discuss how node-link diagrams can be used, for example, to issue instructions, to mediate conversation, to manipulate spatial locations, and to explore hierarchically clustered structures. We target the framework to not outline the best interaction practices or conventions but instead to widen the perspective and offer insight for the future graph-based user interface research and development.

PREFACE

This bachelor of science thesis is made for the Faculty of Computing and Electrical Engineering of Tampere University of Technology.

I would like to give my deep regards to adjunct professor Ossi Nykänen for directing me gently through the thesis process, our deep and enlightening discussions, and for all the advices on a broad set of topics. I also want to thank researchers Jukka Huhtamäki and Anne-Maritta Tervakari and my friends Samil and Elina for additional comments, corrections and inspiring discussions.

In Tampere, Finland, on June 16, 2015

Akseli Palén

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

> Wholly new forms of encyclopedias will appear, ready-made with a mesh of associative trails running through them
>
> – Vannevar Bush, As We May Think, 1945.

> Good hypertext design is based on [...] the use of cues to show the structure of the information space to the user.
>
> – Jacko Sears, Human-Computer Interaction Handbook, 2008

Graph-based user interfaces are types of user interfaces (UI) that employ graph structures as their main *UI metaphor* [1]. In other words, a graph-based UI exploits the resemblance with familiar network structures, such as tree branches, forest paths, and animal skeletons. By mimicking their workings, a graph-based UI allows the interface users to benefit from the existing knowledge on how the interaction with networks should be carried out. Other familiar, even though more human-induced structures include building frameworks, star constellations, fishnets, molecule illustrations, and maps, consisting of cities and connecting roads.

Probably due to the familiarity and also the generality of graph structures, graph-based UIs are applied on a wide range. Applications can be found from mind mapping and network analysis to audio synthesizers and linguistic networks as illustrated in Figure 1.1, in addition to genetic maps, database design, semantic networks, and many others, listed for example in [2]. Starting from the 1980's [3][4] multiple fields have studied graph-based UIs in their own context, notably *knowledge engineering*, *process engineering*, and *information visualization*. However, in spite of the vastness of applications and the array of fields, only little is known about graph-based UIs in general. A media processing related review by Schultz et al. in 2008 [1] is one rare instance where these UIs are handled on a general level. Thus, to begin to fill this gap, in this study we try to approach the subject from a domain-independent perspective.

As the variety implies, graph-based UIs are interacted within many ways and for many purposes. However, the interaction is a wide concept. For example, let us consider two cases: discussing with a person and searching for a flower from a forest. Even though the both require interaction with the environment, they fundamentally differ in how the interaction is carried out. Thus, it is not obvious what is meant

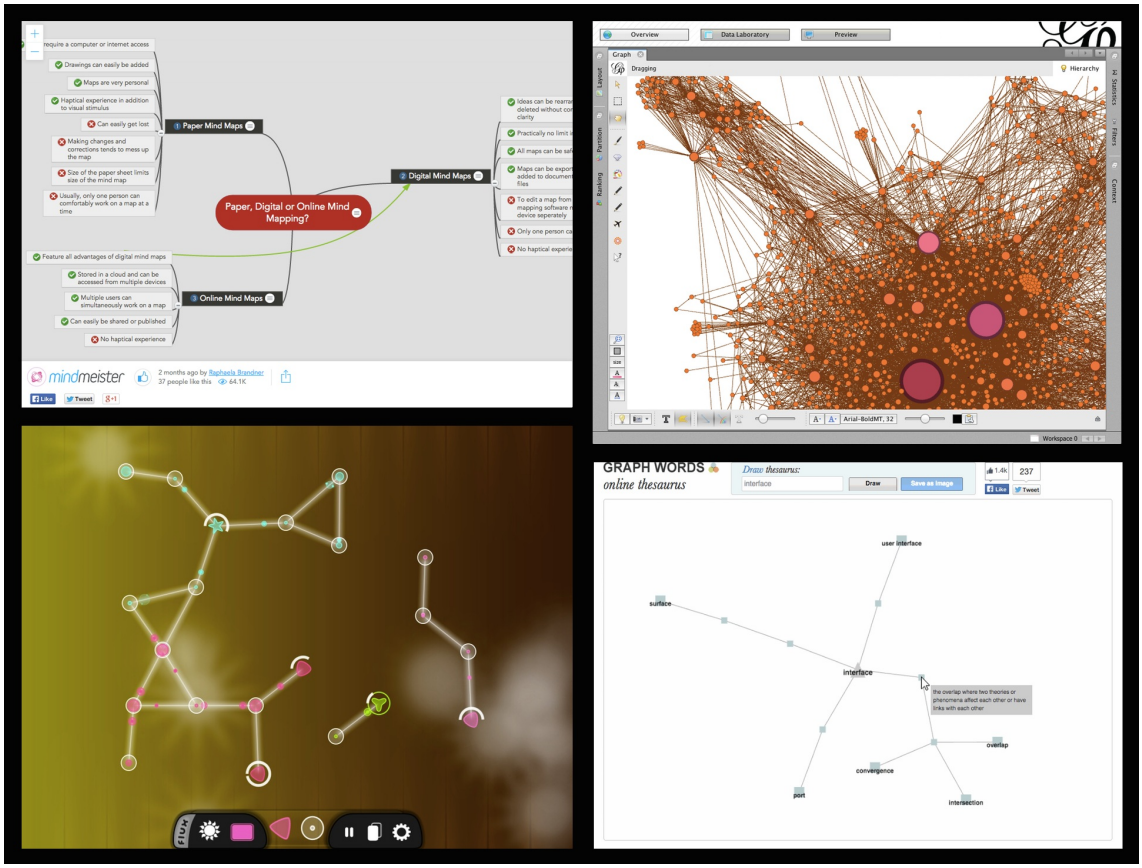

Figure 1.1: Four applications where a graph-based UI plays a central role in their UI. Starting from the top-left corner, an online mind mapping application MindMeister *[5], a graph visualization platform* Gephi *[6], an ambient music composer mobile application* Aura Flux *[7], and an online thesaurus* Graph Words *[8].*

by interaction in graph-based UIs either. Which kind of graph-based interaction is possible? Are there some fundamental limitations? Altogether, where are the borders of the interaction design space of graph-based UIs?

To answer these questions, in this study we review graph-based UIs on a general level and try to get a grasp on what interactional extents graph-based UIs are used or could be used for. By building upon interaction models and frameworks suggested in human-computer interaction literature, and by drawing examples from several publications, we construct a descriptive, interaction-type and user-intention based interaction framework for graph-based UIs. We target this framework to be device and interaction technique independent and thus to be applicable wherever graph-based UIs emerge.

For the task, we first conducted a systematic literature review on about 20 years of publications of two major human-computer interaction journals, Journal of Human-Computer Studies [9] and Proceedings of the ACM Symposium on User Interface Software and Technology [10], and shorter time spans of several others [11–16]. From the findings we picked references and keywords for a more focused search with three science-oriented search engines, Google Scholar [17], ACM Digital Library [18], and IEEExplore [19], allowing us to reach both the oldest and the most important publications on graph-based UIs.

To outline the structure of the thesis, we will first discuss interaction in graph-based UIs in general. We define what is meant by interaction and how it applies to graph-based UIs. Then we approach the graph-based interaction from the point of view of four main interaction types: *instructing*, *conversing*, *manipulating*, and *exploring*, and provide examples and further analysis for each. Finally, we will discuss the validity and extent of the survey and take a glance into the future of the graph-based interaction.

2 INTERACTION

We begin by describing our approach and the context of the study. First, we examine the graph-based interaction in general. Secondly, we deploy an interaction model by Ren et al. [20] to delimit the study, and lastly an interaction framework by Rogers et al. [21] to derive an initial structure for our framework.

2.1 Syntax of interaction

All the familiar structures mentioned in Chapter 1 can be modeled with *graphs*. Because of the generality and that the graph being an abstract, mathematical concept without a form of presentation by its own, it can be called a fundamental structure that can represent everything [2, p. 24][22]. To in turn represent a graph, apart from its mathematical notation, *adjacency matrices* or *node-link diagrams* are typically used. Even though the matrices might outperform the node-link diagrams in many information seeking tasks [23], and also hybrids exists [24] as illustrated in Figure 2.1, the node-link representation similar to Figure 2.2 is more prominent [25, p. 144] and the graph presentation method of the graph-based UI. Therefore, to clarify, an alternative or even more accurate term to refer to the type of UI discussed here would be the *node-link user interface*.

Before a deeper analysis of the interaction on a node-link diagram, we must first define what we exactly mean by the node-link diagram. To give a definition that is

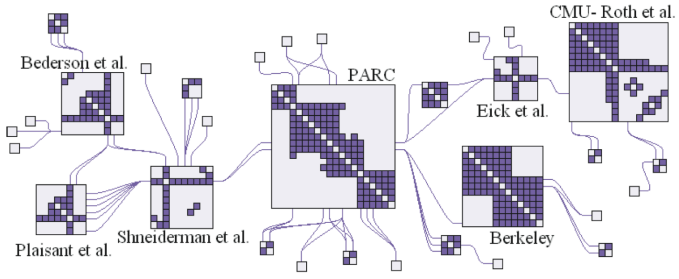

Figure 2.1: InfoViz co-authorship network represented as a NodeTrix *diagram, a hybrid of node-link and matrix-based graph representations. Adopted from [24].*

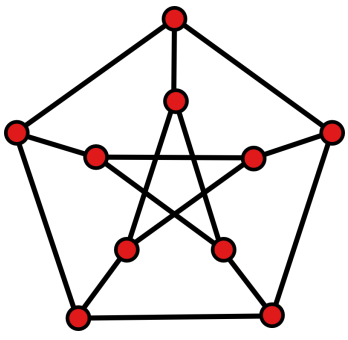

Figure 2.2: A node-link diagram of the Petersen graph.

formal enough for our purposes, we will use graph theoretic terminology but avoid mathematical notation. We define a node-link diagram being a set of *nodes* and *links*. Each node and each link has a *geometric object* in a n-dimensional space and a *physical object* fitted inside the boundary of the geometric object. Additionally, each node is associated with an abstract *vertex* and each link is associated with an abstract *edge*, and thus, the node-link diagram as a whole is associated with an abstract graph. Finally, we define a physical object to be a visual, tactile, auditory, or other type of perceivable object in the physical world. The geometry of an auditory object, for example, could mean a size and shape of the area where the audio can be heard. This definition allows us to describe node-link diagrams in both 2D and 3D, apply graph-theoretic results and algorithms to them, and also it does not limit us to the visual mode of interaction.

With the aid of the definition, we can analyze the interaction on a node-link diagram further and employ the linguistic terminology of Hutchins et al. [26]. We can say that the syntax of the *interface output language* of a graph-based UI equals the definition of the node-link diagram. In other words, the syntax, i.e. the structure of this diagrammatic language, includes the formalism of the graph, enhanced with physical objects and geometric properties. Governed by the syntax, the physical objects emit signals for the users to perceive and conceptualize, and thus form the vocabulary of the interface output language of a graph-based UI.

To consider the *interface input language*, on most graph-based UIs the users can, in addition to perceiving, refer to the physical objects. The users are able to interact directly with the nodes and links, for example by pointing and dragging. Therefore, the interface input language of those graph-based UIs is, if not symmetric, at least similar with the output. By deriving from [26], having this kind of *inter-referential I/O* makes each of those graph-based UIs a *direct manipulation interface*, a renowned concept introduced by Shneiderman in the early 1980's [27]. From this notion, we can include the well-known benefits and drawbacks of direct manipulation [26], even though they are left to the reader to explore.

As nodes and links are now identified as pieces of syntax, carrying semantics and physical form of their contents, do they also have semantics on their own? And more interestingly from the cognitive perspective, why do we interpret the semantics as we do? It has been stated that the explicit links make it easy for a human to understand structure [28, 29]. A reason behind this is that the links exploit the *gestalt law of continuity* [30, p. 61] and therefore allow the human mind to understand that the linked nodes are related. In addition, many *graph layout algorithms*, especially *force-directed algorithms*, arrange the nodes so that the *configuration proximity*, i.e. the distance of two nodes on the diagram, is proportional to their distance in the abstract graph [31]. This more implicit relationship exploits, we propose, the *gestalt law of proximity*, making the mind to understand that two nodes are related if they can be seen nearby each other. Through these two laws, nodes and links efficiently

emit the message of relatedness. In spite of the importance of the cognitive aspect in UI design, we do not push the aspect further but only wanted to establish an initial view on what human properties the graph-based interaction is based on.

As a general matter, we would like to elaborate the terminology used when we refer to parts of a node-link diagram or an abstract graph. As implied in the definition, the counterparts of nodes and links in a graph-theoretic, abstract graph are called vertices and edges. Where nodes and links include a physical representation, commonly a graphical form, the vertices and edges do not. In spite of that in this study we use this separation, we want to emphasize that many authors use the terms interchangeably and do not make a distinction between the mathematical concepts and their physical representations.

2.2 Levels of interaction

On node-link diagrams and in user interfaces in general, the interaction happens on many levels of abstraction. On a low level we could understand the interaction as the events of pressing down a button or releasing it, changing color of a pixel et cetera. In contrast on a high level, we could consider user's long-term goals and how system responses to those. For example, if a couple wants to spend an evening out, they can interact with a restaurant. The restaurant hopefully responses by allowing meals to be ordered and the couple's hunger for food or a shared experience to be answered. The levels are numerous.

For our task to map the interaction design space of graph-based UIs, it is not convenient to consider all the levels. What we need is a suitable level of abstraction that releases us from domain-specific, very high-level tasks but keeps us away from system-specific input methods and technologies like the mouse or the touch screen. With this kind of restriction, we might be able to create a framework that is general enough but still as close to the interface as possible.

In 2013, Ren et al. proposed a three-level interaction model to catch the hierarchy in interaction tasks: *goals*, *behavior*, and *operations* [20]. A task on a goal-level represents a small task that users intend to do and requires users to behave on the behavior-level to reach the goal. This conscious behavior, a set of behavior-level tasks, is transformed to more or less subconscious physical operations that become detected at the interface as raw input primitives, operation-level tasks. Therefore, the operation-level regards interaction as how it happens on the exact surface where the signals from the human are sensed by the system and vice versa.

For a concrete example in a graph-based UI, an intention to connect two nodes is a goal-level task. To reach this goal, some graph-based UIs require the user to behave in drawing a line between the nodes. On the operation-level, this behavior is divided

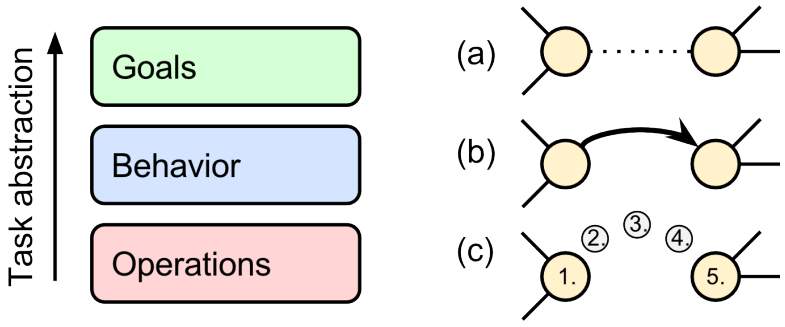

Figure 2.3: The three levels of abstraction in interaction and corresponding examples in the task of connecting two nodes; (a) an intention to create a new connection, (b) a drag between the nodes, and (c) input events recognized by the software. Derived from [20]

into the input events of a press, multiple moves, and a release. See Figure 2.3 for an illustration.

In this study, we focus to the goal level. Therefore, in spite of being important for UI design, we will not discuss behavior-level tasks such as a mouse click or a pinch zoom. Nonetheless, the two behavior-level tasks are commonly utilized in higher, goal-level tasks like selecting a node or zooming in. These higher tasks are not device or input method dependent but still common in graph-based UIs [1]. Therefore they are exactly the tasks the framework should be constructed on.

2.3 Types of interaction

In their book *Interaction Design*, Rogers et al. proposed there to be four main types of interaction: *instructing*, *conversing*, *manipulating*, and *exploring* [21]. For example in a restaurant, the chief cook gives *instructions* to a helper. By these instructions, the helper *manipulates* the ingredients until the meal is ready to be delivered. A waiter delivers the meal by *exploring* the tables to find the right customers and, once found, probably interrupts their *conversation* by handing over the meal. For a graphical illustration of the four types, see Figure 2.4.

For a more abstract description, instructing is about executing commands, selecting actions, or doing gestures to trigger them. Conversing brings in a more equal dialog, an iterative interchange of messages common in natural human communication. However, when we consider manipulating interaction, the world and its objects are not discussed with, but more directly constructed and modified by using the common knowledge of how the world behaves. In exploration, by contrast, this knowledge is not used to change the world but instead to change one's perspective on it to understand what it contains.

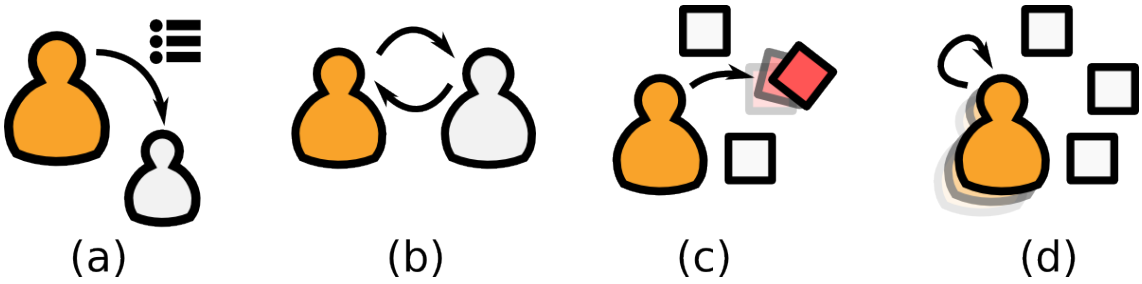

Figure 2.4: Our graphical illustration of the four main types of interaction by Rogers et al. [21]; (a) instructing an agent i.e. human or computer, (b) conversing with an agent, (c) manipulating the surrounding world, and (d) exploring the world by changing one's perspective.

In our pursuit for graph-based interaction tasks, this framework of the four interaction types quarters the search space. Therefore, in addition to that it helps us in determining which kind of tasks to look for, it enforces us to explore these tasks from each angle. Furthermore, the angles immediately raise a non-trivial question. Are all the main types of interaction possible with graph-based UIs? For instance, how can one instruct through a node-link diagram? How could one converse with them?

On top of the four interaction types we build our graph-based interaction framework. In the following chapters, we go through each type separately, present studies and applications to get a grasp on in which types of interaction tasks they are realized, and propose graph-based UI specific interaction subtypes for each. The resulting two-level taxonomy will be our graph-based interaction framework. As instructed, we begin with the graph-based instructing.

3 INSTRUCTING

From the literature, we can identify two types of graph-based instructing:

- *Executing*: a node-link diagram used as a menu of instructions
- *Programming*: a node-link diagram used as an instruction

Menus, consisting of commands, are in some occasions represented as node-link diagrams. In 2008, Bailly et al. introduced a *flower menu* [32], a type of *radial menu* and *marking menu*, where once the menu is activated, the user is shown a set of radially layouted commands that are connected with links to a node at the menu center, as in Figure 3.1. The user is required to make a gesture along one of the links to execute the connected command. Therefore, we suggest this to be a clean example of instructions given through a graph-based UI and a typical case of *executing graph-based instructing*.

We can also think of *graphical database queries* [3, 34]. For example in 2012, Abouzied et al. described *DataPlay* [33], a database query tool that allows users to build and manipulate *graphical query trees* specified by a *graphical query language* and represented as node-link diagrams similar to Figure 3.2. Once a query tree is ready, user can execute it in a manner akin to a traditional database query. What we see here is a combination of manipulating and instructing interaction types; user creates a node-link diagram through manipulation and the node-link diagram, as a whole, represents one single instruction. We see this combined type to be an instance of *programming graph-based instructing*.

Another and slightly different case where a node-link diagram represents a single instruction can be found in the context of *process control*. In 2013, Rietzler et al. introduced *homeBLOX* [35], a home automation system with a graph-based UI, presented in Figure 3.3. The homeBLOX UI was designed to manage a network of sensors, operators, and devices. For example in a homeBLOX compliant home, the sensors of a morning alarm and brightness level could be combined with an *and* operator to turn on a coffee maker. Now, we can notice that the resulting system is one large instruction for how the home should work. Being created by manipulation, this is another example of the combined interaction type. However, in contrast with the DataPlay, the target of instruction is an ongoing, continuous process instead of an operation to be issued and executed in a blink of an eye. Therefore, if we classify

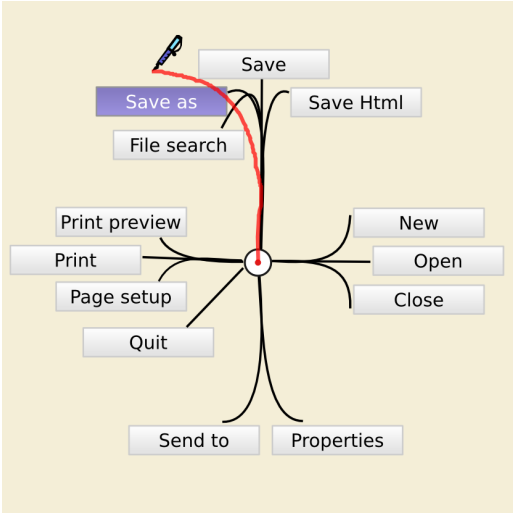

Figure 3.1: A flower menu. Restored from [32]

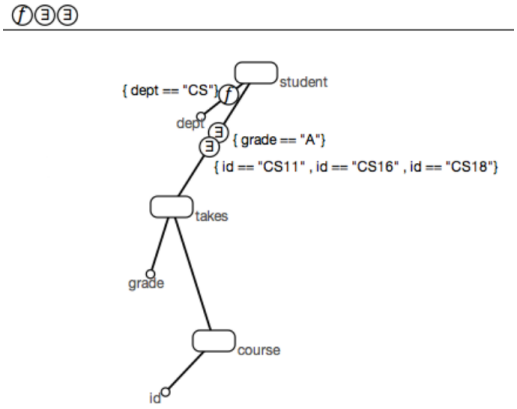

Figure 3.2: A graphical database query builder of DataPlay *query tool. Adapted from [33]*

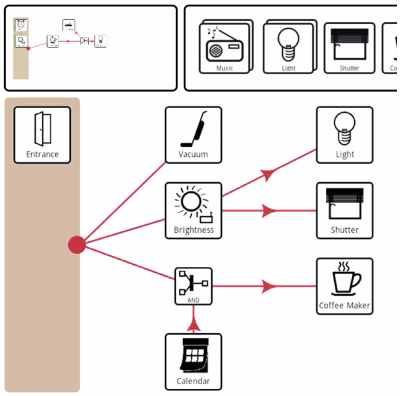

Figure 3.3: A process editor of homeBLOX*, a home automation system. Adapted from [35]*

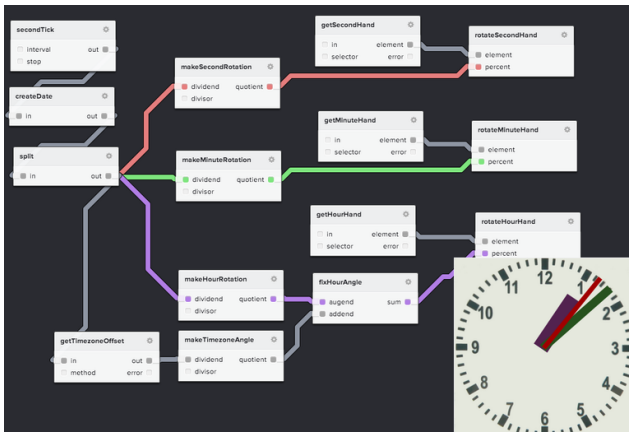

Figure 3.4: An analog clock implemented with a visual programming tool NoFlo.js*. Adopted from [36]*

the use of DataPlay further as *discrete programming graph-based instructing*, the use of homeBLOX can be seen as an instance of *continuous programming graph-based instructing*.

Finally, to justify the terms *executing* and *programming*, we notice the instruction construction in DataPlay and homeBLOX to be a type of *visual programming* [1], and basically similar to the programming in Figure 3.4. In contrast with the flower menu where a user manually *executes* gestures to instruct, here the user *programs* the instruction in a formal visual language. Visual programming is an important application of graph-based UIs and further discussed in [1, 37].

4 CONVERSING

From the literature, we can identify two types of graph-based conversing:

- *Discrete*: a node-link diagram used as a message (Figure 4.1a)
- *Continuous*: a node-link diagram used as a medium (Figure 4.1b)

Many mind mapping tools, such as MindMeister (Figure 1.1), Freemind [38], and Prezi [39], allow sending mind maps to others to be explored, learnt, and modified. In some occasions, as in Prezi, the recipients are able, if allowed, to reply by comment, by another mind map, or by a modified version of the original map. In a workplace meeting, a workflow diagram can be presented as a part the conversation. In all the cases, a node-link diagram is used as a discrete message and acts as a vocabulary item of the conversation, similar to a verbal sentence. Therefore, in spite of being often preceded or followed by nondiagrammatic messages, we label this transmission of node-link diagrams as *discrete graph-based conversing*.

In addition to mind mapping, MindMeister and Prezi can be used to conduct presentations on node-link diagrams similar to the diagram in Figure 4.2. A Prezi presentation consists of a sequence of frames where each frame is a view to the diagram, as illustrated in Figure 4.3. As the diagram is presented to the audience in a frame by frame fashion, the syntax of the node-link diagram, including nodes and links, is used to construct the message of each frame. In spite of the contrast to transmitting a diagram as a whole, we still see this as another example of discrete graph-based conversing.

For the *continuous graph-based conversing*, MindMeister and Prezi allow real-time collaboration on a node-link diagram. Multiple users on separate devices can explore

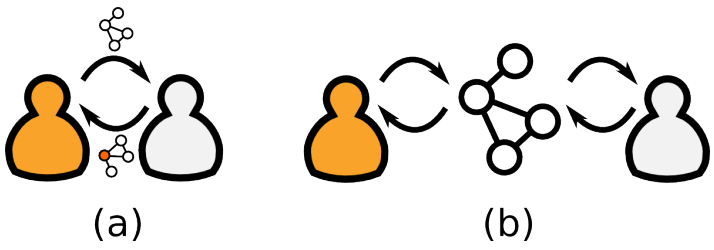

Figure 4.1: The two types of graph-based conversing: (a) diagram as a message and (b) diagram as a medium.

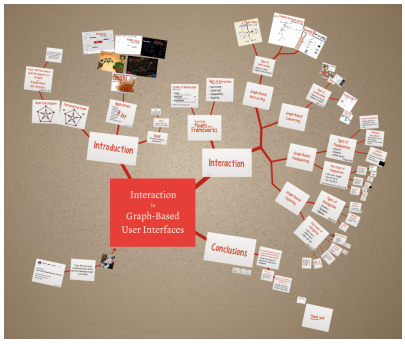

Figure 4.2: A Prezi diagram created for this study.

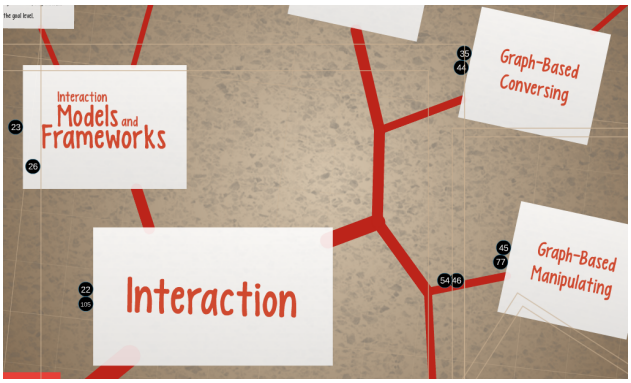

Figure 4.3: The slides of a Prezi presentation are actually frames on a large diagram. The frames are associated with their sequence numbers, presented on the black circles.

and manipulate the diagram concurrently and their moves and modifications are shared instantly, thus creating a sense of shared virtual space. Even though the syntax of conversing still consists of nodes and links, the node-link diagram as a whole has become a *medium* for conversation. The same notion of the diagram becoming a medium has been made by Laufer et al. in their Prezi-introducing paper [39]. Collaboration on node-link diagrams has also been discussed by Tolosa et al. [40].

To provide yet another view to graph-based conversing and to using a node-link diagram as a medium, we have to first discuss *conversation visualization*. In 2008, Gómez et al. visualized several large discussion threads of Slashdot website by using *radial trees* [41], presented in Figure 4.4. These *discourse diagrams* [42] allow brief analysis of massive conversations. For example, possibly important comments can be pointed out from their numerous child nodes or a long-lasting debate can be recognized from a long, nonbranching stem. Altogether, these diagrams help in understanding the topics and the structure of large-scale conversations, and therefore they have also found use in nonscientific purposes [42].

As our topic here is conversation, it is important to note that these discourse diagrams visualize the result of especially conversing type of interaction. The interaction with the plain discourse diagrams is still only exploration and not conversation. Although, this does not always have to be the case. What we now suggest is that, in addition to being navigable and readable visualizations [42], the UIs of these diagrams could also allow joining the discourse. For example, by adding a new leaf node to the diagram a user could reply to the comment represented by the previous node, thus taking part in a *graph-based discussion*, and as visioned in Figure 4.5. We do not take this idea further or claim it a successful approach to communication but only want to show that this kind of interaction could be possible and to provide yet another example of continuous graph-based conversing.

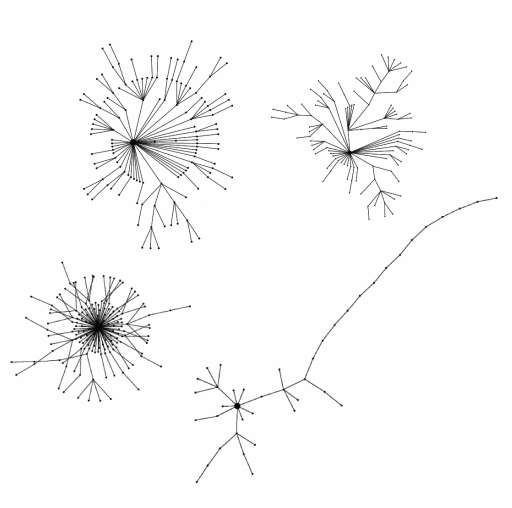

Figure 4.4: Four discussion threads visualized as radial trees. Each node represents a comment. At the bottom-right two users have had a long-lasting debate. Adapted from [41]

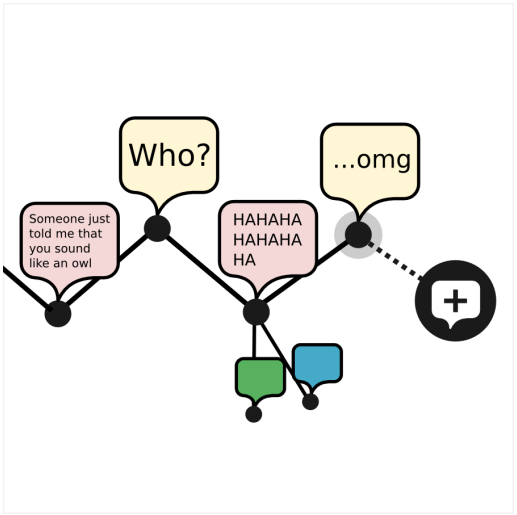

Figure 4.5: An imaginary graph-based chat where a user can join the discourse by adding a bubble node. The discourse progresses from left to the right. The joke is adopted from a image sharing website at smartphowned.com

5 MANIPULATING

From the literature, we can identify four types of graph-based manipulating:

- *Geometric*: direct manipulation of the geometry
- *Topological*: direct manipulation of the topology
- *Algebraic*: manipulation through symbols
- *Temporal*: manipulation through a version history

When a user engages in a manipulating interaction with a node-link diagram, the diagram becomes modified. Whether the manipulation is done directly, in direct manipulation manner, or via instruments, such as formal language or version history, the geometry or the structure of the diagram becomes altered, and thus the state of the world around the user changes. There is two types of changes; the changes can happen on the geometric properties of the nodes or links, or in the topology of the underlying abstract graph. However, we note that the interaction that causes the changes has a separate taxonomy, and that taxonomy is the four-part taxonomy discussed here.

In an abstract graph, vertices and edges do not have spatial locations, sizes, or shapes. However in node-link diagrams, as the definition in Chapter 2 implies, these geometric attributes are the fundamental requirement for representation. Therefore, it is not surprising that in many applications that realize a graph-based UI the spatial arrangement of nodes, also called the *configuration*, is in high importance [2]. We can use the aforementioned Prezi as an example. In a Prezi diagram, the visual beauty of the diagram depends highly on the clear but interesting arrangement of nodes and links. The nodes can be placed and moved freely and arranged in any configuration that the creator sees suitable. A set of other examples are provided by a study by Schultz et al. where they examined 14 graph-based UIs used to control process flows such as audio synthesis and 3D shader design [1]. Interestingly, they found that many of these applications did not provide an automatic graph layout algorithm to arrange the nodes but instead left the freedom or the burden of configuration for the user. The same applies to Prezi; there is no method for automatic rearrangement. We see these nonautomatic and direct spatial modifications of the node locations as instances of *geometric graph-based manipulating*.

In addition to the mere spatial locations of nodes and links, Prezi lets users to modify their rotation and scale via handles provided at the corners. Same applies

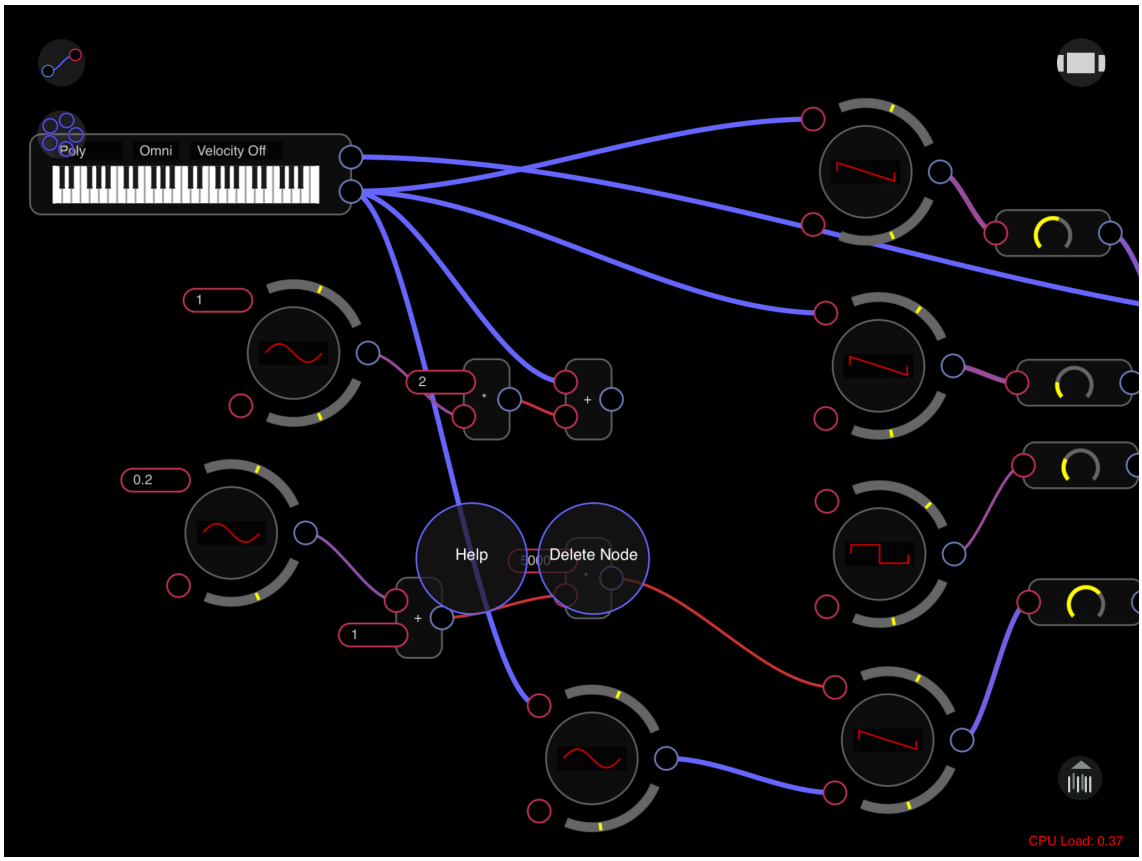

Figure 5.1: The graph-based UI of Audulus, an audio processing application. Adopted from [44]

to a mind mapping and charting application LucidChart [43], where multiple types of diagrammatic components can similarly be transformed via scaling and rotating. When we apply these *geometric transformations*, the average spatial locations or the physical objects of nodes and links do not necessarily change even though their geometric boundary does. Therefore, we also regard issuing these transformations to the boundaries, when done in a direct manipulation manner, as instances of geometric graph-based manipulating.

Topological graph-based manipulating is about directly meddling with the structure of the node-link diagram. In other words, the diagram is manipulated so that the underlying abstract graph changes i.e. vertices or edges become created or removed. In Audulus [44], a software sound synthesizer similar to the audio processing applications analyzed by Schultz et al., the nodes represent sound generators or effects, and links represent cables or streams that govern how audio signals flow between the nodes, as presented in Figure 5.1. In Audulus, the configuration brings clarity for the user but does not affect the sound processing network. Instead, the topology of this network defines how the signals flow and how the system sounds.

If we furthermore examine the UI of Audulus in Figure 5.1, we notice that each node is equipped with its own interface for manipulating its parameters such as frequency

or amplitude. In MindMeister, a user can directly modify a textual content of a node. These content-specific interfaces are not anymore graph-based or even resemble node-link diagrams. Therefore, we now find that the physical objects of nodes as well as links can expose their own UI completely independent of the surrounding graph-based UI. The opposite parent-child relationship is also possible and even typical. As in both MindMeister and Gephi, the graph-based UI is embedded into an outer, traditional WIMP-like UI that consists of non-graph-based UI elements like menus, toolbars, and buttons.

Hence, even though these inner or outer UIs could offer interaction methods for instructing, conversing, manipulating, or exploring, we do not consider their use being graph-based interaction. However, we note two exceptions. First, if an inner or outer UI is a graph-based UI then the interaction is naturally graph-based. Second, if the use of a non-graph-based inner or outer UI affects the node-link diagram, their use takes part in a graph-based interaction, although indirectly. For the latter case, MindMeister and Gephi provide an example. In both applications, the outer UI provides tools for layouting the diagram automatically. Therefore, in this latter case, the interaction is not anymore direct but *instrumental* [45], and we can categorize it further as algebraic or temporal.

In *algebraic graph-based manipulating*, nodes, links, and their properties are referenced and manipulated via their symbolic representations. The symbolic representation can exists in the form of a simple button or even a complex formal textual language. For instance, *SPARQL 1.1* is an RDF query language designed for both querying and manipulating semantic RDF graphs [46]. If a node-link diagram is based on an RDF graph and we update that graph by SPARQL, we classify this interaction being an instance of algebraic graph-based manipulating. In this case, in spite of that the input interface language is textual and not graph-based, the output interface language is still the language of the node-link diagram and thus we see this interaction to be graph-based, even though indirect.

Could a symbolic formal language for manipulation be also graphical? In addition to textual languages, the literature reveals graphical query languages for RDF graphs, for example SPARQL-based RDF-GL [47] and vSPARQL [48]. A simple example of the latter is presented in Figure 5.2. In the both cases, the queries are node-link diagrams and therefore their use includes graph-based interaction. However, neither supports queries that manipulate RDF graphs and therefore their use could only be accounted as graph-based instructing, similar to the use of DataPlay query tool in Chapter 3. On the other hand, the result of a SPARQL query can be an RDF graph [46] that can furthermore be represented as a node-link diagram, as done in Figure 5.3. Would then a graphical modification to the query also cause a change in the diagram? For our taxonomy, does it matter that the symbols in the query do not directly refer to the nodes or links? In this case, both the query and the RDF graph are used as instruments to modify a node-link diagram and therefore,

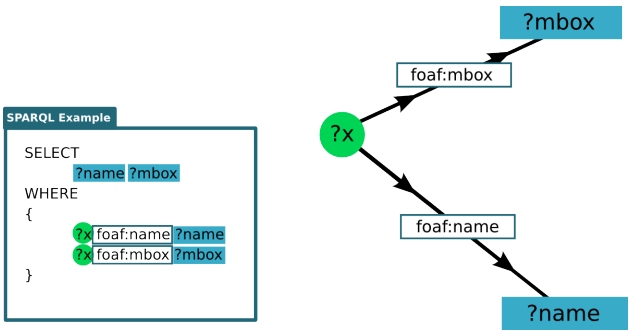

Figure 5.2: A SPARQL query and its visual counterpart in vSPARQL. Adapted from [48]

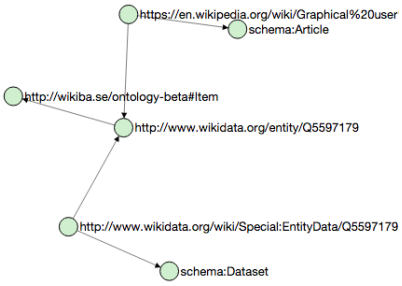

Figure 5.3: An RDF graph of a WikiData *entry [49], visualized with* Visual RDF *tool [50]*

regardless the multiple intermediate levels of symbolism, we still classify this as a case of algebraic graph-based manipulating.

For the fourth subtype, where the previous three subtypes only advanced the current state of a node-link diagram, in *temporal graph-based manipulating* a wider time dimension of the diagram is regarded. Typical examples are the undo/redo function and a version timeline, as available in Prezi and MindMeister. In these cases, the current state of the diagram is reverted to a previous version, or changed back to a previously reverted, future version. Thus we note that the version history is *linear*; there is only one simple timeline.

For a *nonlinear* instance, we must first think of distributed revision control systems such as Git or Mercurial. The both are popular tools in software development as they help developers to maintain different versions of the project and develop new features independently and simultaneously. In a large Git project, the codebase of the project typically does not have one current state but instead is divided into multiple *branches*, for example one branch for each new feature. Later on when a feature of the branch is finished, it can be merged to a release branch for a public release.

As a result of this practice, the version history is not linear but a *directed acyclic graph* [51]. The graph can be visualized as a node-link diagram, as illustrated in Figure 5.4. We can interpret the diagram as a visualization of the temporal manipulations of a source code where the time continuum of the code becomes splitted or multiple continuums merged, thus yielding modifications to the code. We propose that one could have a similar nonlinear version history for a node-link diagram, for example for the visually programmed clock in Figure 3.4. Similar proposition has been made by Chen et al. for 3D meshes and for images in general [52]. However, as we are unable to find a realized case of this kind of *nonlinear temporal graph-based manipulating* from the literature, for now, we can only imagine.

There exists additional views into structure manipulation. For instance in 2012, Hoarau & Conversy derived a set of requirements for manipulating objects through

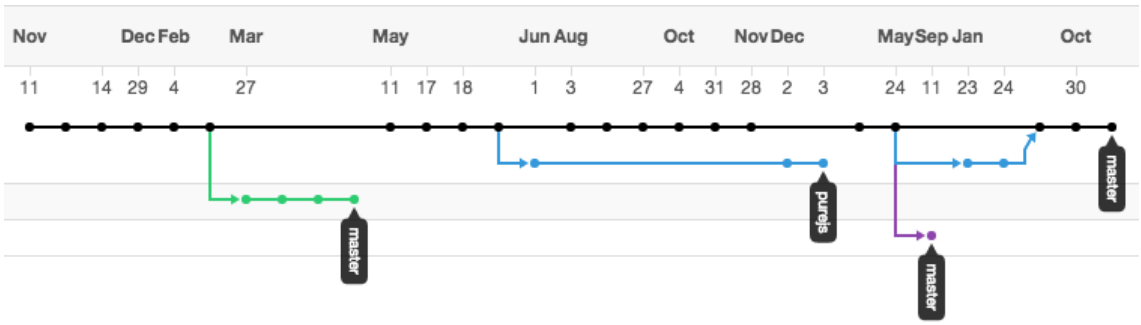

Figure 5.4: A simplified Git revision history of the VisualRDF tool. The tool was employed in the production of Figure 5.3. Adapted from [53]

structures [54]. The requirements encompassed features such as search and selection of structured objects, and exploration and comparison of alternative versions, thus implying the importance of an undo/redo function and a version history. Interestingly, the requirements also noted benefits of informality, discussed a decade earlier by Thimbleby under the title *permissive user interfaces* [55]. Within the context of graph-based UIs, the informality can refer to the allowance of isolated nodes. For example, MindMeister does not allow such disconnected nodes where Aura Flux and Gephi do (see Figure 1.1).

As the requirements of Hoarau & Conversy indicate, the ability to select is vital to many manipulating tasks. For example in MindMeister, one can manipulate the topology of a mind map by cutting, copying, pasting, deleting, and relocating groups of selected nodes. In Prezi, one can via selection rotate and scale nodes and links in groups, thus alleviating their geometric manipulation by reducing the *viscosity* [54] of the UI i.e. the laboriousness of change. However, with the taxonomy of geometric, topological, algebraic, and temporal graph-based manipulating, we cannot unambiguously classify the task of selection. It is used to identify the items for manipulation, to compose groups, or to point out interesting parts of a diagram for discussion. In a way, it is a part of algebraic interaction; the selection could be seen as a symbol for a set of nodes and links. On the other hand, it usually precedes the actual manipulating interaction, whether geometric, topological, algebraic, or temporal. Therefore, in this study, we decide to label it more as a supporting task and thus we do not see the benefit in establishing a dedicated interaction type for it. As our decision, the role of selection stays the same in the case of the graph-based exploring, the topic of the next chapter, Chapter 6.

6 EXPLORING

From the literature, we can identify four types of graph-based exploring:

- *Geometric*: moving by applying geometric transformations
- *Topological*: moving along the structure
- *Algebraic*: altering the search space through symbols
- *Temporal*: moving in navigation or version history

We use the term *exploring* in a broad sense; in addition to exploring, being activity to find something new, we include the activities of *pathfinding*, described as navigating to a predetermined location, and *searching*, described as looking for something specific even though the location is not exactly known. Altogether, regardless of the goal, we refer to interaction where users change their view of the surrounding world.

Before defining the four exploration types, we discuss the concept of view. We define the *view* as a portion of the information space that is perceivable to the user at a given time. Often, the view can be understood as the position of the user in the information space. Sometimes, the view is assembled from pieces across the space and no single location can be determined. In either way, the very reason for exploration is, as we see it, that one cannot perceive everything at once and thus needs to rearrange the space or move in it.

Not all graph-based UIs allow altering the view. For example in the music composer Aura Flux (Figure 1.1), a node-link diagram can be constructed only within the area of the screen and there is no means to alter the view. On the other hand, some graph-based UIs support multiple concurrent views. Not unlike to multiple windows in a WIMP interface, in graph-based UIs, multiple views can be used for graph comparison, creating connections between distant nodes, or moving or copying nodes and links from place to another. Alternatively within a single view, the space can be transformed to give a perception of multiple concurrent locations. These multiple locations are often discussed within the context of *distorted focus+context views* like *fisheye views* [56] and thus called *multiple focal points* or *multiple foci*. For further details, the reader could look into articles by Schaffer et al. [57] and Toyoda & Shibayama [58].

In *geometric graph-based exploring*, users directly apply geometric transformations to their views. In contrast, in geometric manipulation in Chapter 5 these transformations were applied to nodes and links. Such exploration can be realized in the

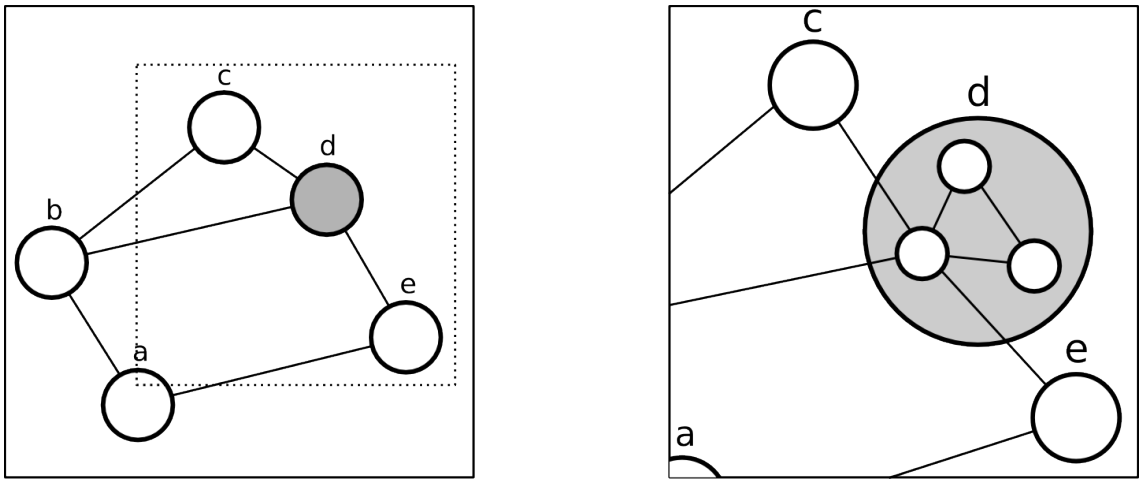

Figure 6.1: Concurrent geometric and semantic zooming in a hierarchically clustered graph; (a) before zooming, (b) after zooming. Derived from [57]

form of *panning* or *zooming* the view, which is typical to a *zoomable user interface* (ZUI) [59]. The ZUI style of exploration is common in graph-based UIs as noted by Schultz et al. [1] and Tominski et al. [60, p. 662] and also appear in MindMeister, Gephi, Prezi, and Audulus. Therefore, even though there are exceptions, like Aura Flux, we can conclude that most graph-based UIs are ZUIs as well.

Where in geometric exploration the transformations are typically not constrained except by the boundaries of the node-link diagram, in the *topological graph-based exploring* the transformations happen only along the structure. For example in both MindMeister and FreeMind, users can topologically explore a mind map by using arrow keys to move a focus point from a node to an adjacent one. The view automatically follows the focus. In contrast, with the aid of a pointer such as the mouse, the both applications allow users to pan and zoom freely. For another example, Tominski et al. mentioned *edge-based navigation* [60]. By taking the user to another end of a link just by pressing on the link, the edge-based navigation makes it easy to move from node to another even when there is a long distance between.

In spite of that in the mentioned examples the topological exploration can be seen as a constrained geometric exploration, this is not always the case. When *semantic zooming* [2] is combined with *hierarchically clustered graphs* as done by Schaffer et al. in 1996 [57], we arrive to situations where traveling along topology shows us new information that could not be revealed by geometric transformations. In the hierarchically clustered graphs of Schaffer et al., a node is broken down into multiple nodes while a user zooms in and merged back while the user zooms out, as illustrated in Figure 6.1. Thus, such exploration changes the topology of the presented node-link diagram, which is impossible with geometric transformations alone.

With the third type, *algebraic graph-based exploring*, we refer to exploration where

users temporarily alter the node-link diagram by joining, transforming, or filtering nodes and links, and does that in an indirect manner, through a symbolic language. As with algebraic manipulation in Chapter 5, the symbolic language does not have to be textual. We adopt the term *algebraic* from Baudel, who used it to label this type of navigation in structured data [61]. We use the term more in a metaphorical than in a strict mathematical sense, allowing us to include all the cases where users apply search queries, filters, or *clutter reduction* techniques to reduce, rearrange, or re-emphasize the nodes and links, thus making the exploration more efficient.

Without recapping the symbolic input methods discussed in Chapter 5, we can discuss the output methods. We found multiple techniques proposed in the literature for altering node-link diagrams to alleviate exploration. For example, node-link diagrams can be filtered by *clustering* [57, 58, 62], *graph splatting* [63], *edge splatting* [64], *ghosting* [2], or *hiding* [2]. In some occasions as in Gephi, not unlike penetrating through a dense jungle with a machete, nodes of a dense diagram can be manually moved to alleviate exploration. In addition to Baudel, filtering of graphs is discussed also by Marshall et al. [65], Tominski et al. [60, p. 666], and Landesberger et al. [66]. All in all, we can see that the issue of how a node-link diagram should be temporarily modified for optimal exploration is widely studied in the field of *graph visualization*.

For the fourth type, *temporal graph-based exploring*, we notice that users can modify their views also in the time dimension. As mentioned in Chapter 5, graph-based UI might include a version history, for example in the form of a version timeline or an undo/redo function. Users are able to traverse the history and thus we can count this as an instance of temporal graph-based exploration. In addition, similar to web browsers, users might be able to traverse their navigation history, providing another instance.

In the cases of version or navigation history, temporal graph-based exploring is implemented to alleviate manipulation or exploration [54]. However, it might also be the primary interaction type of the node-link diagram. For instance, Gómez et al. visualized the discourse diagrams (Chapter 4) also in the time dimension to examine how the discussions evolve, as presented in Figure 6.2. Other examples where the graph evolution is in high importance can be found in the literature of *dynamic graph drawing*, for example [64].

The proposed set of four types of graph-based exploring does not form the only possible taxonomy. Sharing similarities to the manipulation requirements by Hoarau & Conversy discussed in Chapter 5, in 2009 Tominski et al. presented an information visualization specific eight-part interaction framework, including tasks for selecting, abstracting, filtering, and undoing, among others [60]. Multiple other information visualization specific frameworks have been gathered by Ren et al. [20]. Even though the frameworks presented by the both research groups provide insight to common

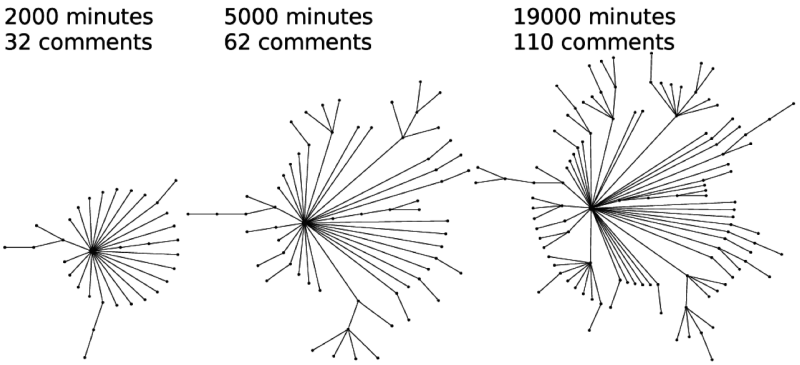

Figure 6.2: An evolution of a discussion thread. Adapted from Gómez et al. [41]

tasks in information visualization systems on all levels of detail, none of them resembles the four-part exploration taxonomy presented here. Among the taxonomies exposed by the literature review, only one shares resemblance with ours, namely the three-part navigation framework by Baudel [61] on which our exploration taxonomy is based.

7 RESULTS

As a result of Chapters 3-6, here we piece together an interaction framework for graph-based UIs, consisting of 4 main types and 12 subtypes of graph-based interaction:

R1 Instructing

- **R1.1 Executing** a node-link diagram used as a menu of instructions
- **R1.2 Programming** a node-link diagram used as an instruction

R2 Conversing

- **R2.1 Discrete** a node-link diagram used as a message
- **R2.2 Continuous** a node-link diagram used as a medium

R3 Manipulating

- **R3.1 Geometric** direct manipulation of the geometry
- **R3.2 Topological** direct manipulation of the topology
- **R3.3 Algebraic** manipulation through symbols
- **R3.4 Temporal** manipulation through a version history

R4 Exploring

- **R4.1 Geometric** moving by applying geometric transformations
- **R4.2 Topological** moving along the structure
- **R4.3 Algebraic** altering the search space through symbols
- **R4.4 Temporal** moving in navigation or version history

In addition to the listed subtypes, we encountered alternatives. In Chapter 3, we identified discrete and continuous cases for graph-based instructing. In Chapter 5, direct manipulation characterized the subtypes R3.1 and R3.2, and in turn, indirect manipulation characterized the subtypes R3.3 and R3.4. However, in all the cases, the selected taxonomies seemed more describing and thus the alternatives were left out from the framework.

In addition to the graph-based interaction types and subtypes, in Chapter 5 we saw that the nodes and links can have their own non-graph-based UIs and that a graph-based UI can be embedded in an outer UI. Interaction on these inner and outer interfaces could include any of the main interaction types. For example, if a user modified a textual content of a node in MindMeister, direct manipulation was

used. The outcomes of the interaction with an inner or outer UI could be completely unrelated to the graph-based UI, and in that case we did not classify the interaction as graph-based. On the contrary, if the interaction with an inner or outer UI affected the diagram, we classified it as indirect graph-based interaction.

In addition to the mixed interaction between non-graph-based and graph-based UI, throughout Chapters 3-6 we confronted tasks that required mixing of types of graph-based interaction. For example, to execute a search query with DataPlay in Chapter 3, the user first engaged in graph-based manipulating, and then issued the manipulated instruction. In Chapter 6, to explore a dense graph, manipulating the configuration was one possible approach. We also confronted apps that allowed multiple types of interaction. Both MindMeister and Prezi used three types of interaction: exploring, manipulating, and conversing. However, as a note for the future, we did not encounter a graph-based UI where all the four main types, including instructing, would have been incorporated.

As a surprising outcome that we did not note earlier, is the asymmetry of the subtypes. Where the subtypes R1.1 and R1.2 are similar to R2.2 and R2.1, they are completely different from the subtypes of R3 and R4, which again are alike. Is there some fundamental flaw in the approach we selected? Why are *topological instructing* or *algebraic conversing* unsuitable subtypes of graph-based interaction? Without ruling out the possibility of the first question, we propose instructing and conversing to be essentially different from manipulating and exploring. As one can detect from Figure 2.4, instructing and conversing are about communicating with an agent and manipulating and exploring are about modifying the state of the world. Additionally, as we saw in Chapter 3, one can either search (explore) for an instruction from a menu or construct (manipulate) an instruction piece by piece. In Chapter 4, the conversation emerged from alternate or concurrent manipulation of a node-link diagram. Therefore it seems that, at least in the case of graph-based UIs, instructing and conversing are always preceded by an amount of manipulating or exploring, thus turning topological instructing or algebraic conversing into odd concepts.

Finally, the framework and its construction showed us that graph-based UIs can be interacted with, and are already interacted with, in every main interaction type. Therefore, the study suggests that they do not have any fundamental limitations from the perspective of interaction in general.

8 DISCUSSION

Here we first discuss the constructed framework and its extent. We note a few oddities and possible flaws and deliberate their reasons. Finally, we reason about the future of graph-based interaction.

8.1 Framework

In this study, we tried to construct a framework that is independent of how the interaction is carried out. For this sake, we focused on the interaction goals and avoided discussion on input gestures or interface devices. Even though the framework was built on references where the focus was mostly on visual desktop applications, we suggest the framework to be applicable in the graph-based UI design regardless of devices (desktop, tablet, phone), input methods (touch, mouse, voice), or even modes of interaction (visual, tactile, auditory; see [67]). As noted by both Dam [68] and Jacob et al. [69, 70], the heterogeneity of devices, input methods, and modes is nothing but increasing, and therefore we see the independence being an valuable property. However, because the visual origins and also the spatiality in our definition of the node-link diagram, limits might occur.

To give an example of a nonvisual graph-based UI and how the framework could be applied, we can think about a physical ridged subway map designed for the blind (Figure 8.1). With the aid of the framework, a designer of such a map can identify the topological exploring interaction being the main interaction type, and by that, consider including the other types of interaction in the map design. For example, the designer could consider the executing instructing interaction and implement it in the form of buttons for user to press to listen some details or to call assistance. After all, the framework allows the designer to explore the interactional possibilities of the node-link diagram.

We recommend that the framework is not to be considered exhaustive or final. Even though we managed to find and review several profound articles on graph-based UIs, we believe many still left unrevealed. There might be interaction types that we did not manage to identify. Furthermore, we feel that we did not study deeply enough the task of selection in spite of its importance in manipulation and exploration. A

Figure 8.1: A photograph of a schematic map of a subway station in Rome for the blind. Adapted by permission from [71]

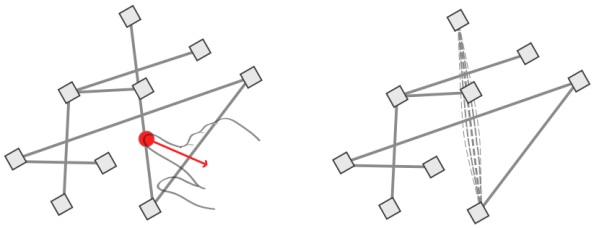

Figure 8.2: TouchStrumming*, an interaction technique presented by Schmidt et al., where a user can pull and release a link to put it into vibration. Adopted from [72]*

closer analysis on what happens when a user selects nodes, links, or groups of them could yield additional interaction types. Also, there might be graph-based interfaces whose interaction design cannot be classified with the framework.

As an example, we have found an interaction task that do not unambiguously fit into the framework. *TouchStrumming*, a guitar-play-like interaction illustrated in Figure 8.2 and introduced by Schmidt et al. in 2010 [72], implies properties of a node-link diagram that we did not consider. Even though the strumming itself is a behavior-level task, it is used to put nodes or links into a vibrating motion. The framework categorizes this as content-specific interaction not related to the graph-based UI. On the other hand, this is manipulation of a geometric graph enhanced with string-like physical properties that occur naturally in spider webs and small tree branches. Therefore from the latter point of view, the vibrations occur in the structure of the node-link diagram itself.

There is one clear point of discrepancy in the framework. The algebraic graph-based exploring, R4.3, requires input to be indirect and symbolic. On the other hand in Chapter 6, we described the type also to involve all interaction where a user modifies the diagram to make sense of it. The latter included the direct manipulation case, exemplified by the analogue of a jungle and a machete. One can notice the inconsistency in the level of directness. A reconsideration could, we suppose, split the current subtype into two: algebraic graph-based exploring that takes into account only the symbolic input, and manipulating graph-based exploring that covers only the cases of pro-exploring manipulation, whether direct or indirect.

To in turn consider the outcomes, we saw that graph-based UIs can be interacted within every main interaction type. Based on that, we suggested there to be no fundamental interactional limitations. This suggestion and the fact that graph-based UIs can embed or be embedded in other types of UIs make us to propose

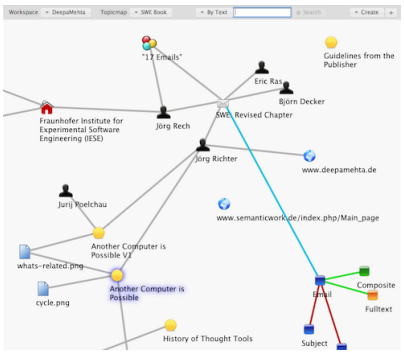

Figure 8.3: DeepaMehta *and its* situation-centered user interface. *Adapted from [73]*

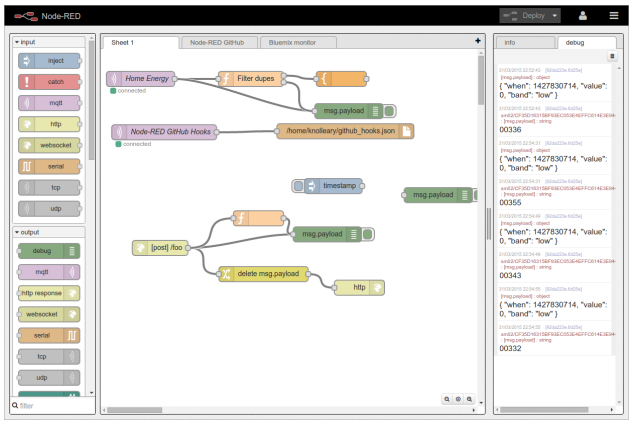

Figure 8.4: Node-RED, *a process control tool for the Internet of Things. Adopted from [75]*

their usage as general-purpose UIs. Furthermore, we even propose a graph-based UI taking the role of the primary UI of an operating system. Even though the proposition is a strong one and might even sound flagrant, there is already a branch of software development devoted to this, or even more radical goal. One of the main visions of the networked semantic desktop project *DeepaMehta* is "the gradual replacement of the application-centered user interface" with their semantic graph-based UI [73], presented in Figure 8.3. For academic reference to DeepaMehta, see [74]. For a lighter example, after the introduction to the UI of the home automation system homeBLOX in Chapter 3, we could effortlessly envision a similar UI to be the primary UI of an embedded control system of an intelligent home.

However, with only the framework and this study, it is hard to say exactly where would graph-based interaction excel and where other types of UIs provide better approaches. The goal of the study was not to be conclusive or final. Instead, we targeted the framework be descriptive, outlining the interaction design space of graph-based UIs. We propose that from the insight given by the framework and the study, in spite of the possible flaws, a graph-based UI designer or researcher is able to see the bigger picture and the richer capabilities of node-link diagrams than what has been offered by previous, mostly graph visualization centered articles.

8.2 Future

The future of graph-based UIs and the graph-based interaction seems anything but clear. A boom of graph visualization techniques in the late 1990s (not the least because the book *Graph Drawing* by Battista et al. [76]) seems to be still ongoing (e.g., see the reviews [66, 77]) and supported by advances in visualization tools, such as D3 [78]. However, relying on our literature review, the tools and the research seems to be mostly geared toward to the needs of information visualization and

exploration. Manipulation is rarely directly discussed, with few exceptions like the paper by Hoarau & Conversy [54].

What comes to the future of graph-based manipulating and also instructing, we predict that if the advances are to come, they are probably seen within the context of the Semantic Web, process control, or visual programming, maybe for the purposes of the Internet of Things. For example, Node-RED [75], presented in Figure 8.4, is a promising graph-based, open-source tool by IBM for connecting devices and services together in a visual manner similar to homeBLOX in Chapter 3 and Audulus in Chapter 5. For conversing interaction, we guess the benefits of collaboration and the increasing technological support for real-time remote interaction, such as mobile internet devices and WebSockets [79], to be probable driving forces.

For a riskier guess, the development of *intelligent user interfaces* might take graph-based conversing to a different level. By applying machine learning techniques the research has already been able to produce learning and adaptive graph-based UIs [58, 80] which predict the interactional decisions of the user. With the aid of social data mining, the interaction sequences can be captured and the decisions predicted [81]. This enhances the one-sided instructing of a machine toward iterative suggestion-decision conversing with an intelligent agent. Additionally, there exists *automated usability analysis* techniques for traditional UIs that turn a traditional UI into a graph structure before the analysis [55, 82–84]. This implies, we hypothesize, that the use of a graph-based UI would make this kind of automatic evaluation, if not an easy, at least an easier task. Finally, as the Semantic Web techniques such as RDF [85] are from the ground up made to be well-formed network-like structures to be understood by computers, we could argue graphs to be a natural breeding ground for applications of machine learning and artificial intelligence, and therefore, for human to intervene, graph-based interaction might be required.

9 CONCLUSIONS

In this study, we constructed an interaction framework for graph-based user interfaces with the goal of outlining their interaction design space. We believe we reached the goal and created a sound and descriptive interaction framework that provides a wider look on the graph-based UI than the previous studies.

Even though the framework is not conclusive or exhaustive as more interaction types might emerge, we hope it to help and guide further graph-based UI research and development by bringing together the capabilities, findings, and instances of graph-based UI interaction from the last 30 years of highly heterogeneous and continuously evolving field of human-computer interaction.

REFERENCES

[1] Christopher Schultz et al. "An Anatomy of Graph-Based User Interfaces for Media Processing". In: *Audio Engineering Society Convention 124*. Audio Engineering Society, 2008-05, pp. 1–6. URL: http://www.aes.org/e-lib/browse.cfm?elib=14625.

[2] Ivan Herman, Guy Melançon, and M. Scott Marshall. "Graph Visualization and Navigation in Information Visualization: A Survey". In: *IEEE Transactions on Visualization and Computer Graphics* 6.1 (2000-01), pp. 24–43. ISSN: 1077-2626. DOI: 10.1109/2945.841119. URL: http://dx.doi.org/10.1109/2945.841119.

[3] Harry K. T. Wong and Ivy Kuo. "GUIDE: Graphical User Interface for Database Exploration". In: *Proceedings of the 8th International Conference on Very Large Data Bases*. VLDB '82. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1982, pp. 22–32. ISBN: 0-934613-14-1. URL: http://dl.acm.org/citation.cfm?id=645910.673453.

[4] Lawrence A. Rowe et al. *A Browser for Directed Graphs*. Tech. rep. Berkeley, CA, USA, 1986.

[5] MeisterLabs. *MindMeister Features*. 2015. URL: https://www.mindmeister.com/features (visited on 2015-05-30).

[6] Gephi.org. *Gephi - The Open Graph Viz Platform*. 2015. URL: https://gephi.github.io/ (visited on 2015-05-30).

[7] Hige Promotions Ltd. *Aura Flux*. 2009. URL: http://www.higefive.com/apps/flux/ (visited on 2015-05-30).

[8] Graphwords.com. *Free Online Thesaurus*. 2015. URL: http://graphwords.com/ (visited on 2015-05-30).

[9] *International Journal of Human-Computer Studies*. years 1994–2015, volumes 40–78. Duluth, MN, USA: Academic Press, Inc.

[10] *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*. years 2001–2014. New York, NY, USA: ACM.

[11] *ACM Transactions on Computer-Human Interaction (TOCHI)*. years 1994–2004, volumes 1–11. New York, NY, USA: ACM.

[12] *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*. years 1999–2002. New York, NY, USA: ACM.

[13] *IEEE Transactions on Visualization and Computer Graphics*. years 1995–2000. Piscataway, NJ, USA: IEEE Educational Activities Department.

[14] *ACM Transactions on Graphics (TOG)*. years 2003–2004. New York, NY, USA: ACM.

[15] *Proceedings of the International Conference on Human-computer Interaction with Mobile Devices & Services (MobileHCI)*. years 2013–2014. New York, NY, USA: ACM.

[16] *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*. years 2000–2002. New York, NY, USA: ACM.

[17] Google. *Google Scholar*. 2015. URL: https://scholar.google.com/.

[18] ACM. *ACM Digital Library*. 2015. URL: http://dl.acm.org/.

[19] IEEE. *IEEE Xplore Digital Library*. 2015. URL: http://ieeexplore.ieee.org/.

[20] Lei Ren et al. "Multilevel Interaction Model for Hierarchical Tasks in Information Visualization". In: *Proceedings of the 6th International Symposium on Visual Information Communication and Interaction*. VINCI '13. Tianjin, China: ACM, 2013, pp. 11–16. ISBN: 978-1-4503-1988-1. DOI: 10.1145/2493102.2493104. URL: http://doi.acm.org/10.1145/2493102.2493104.

[21] Yvonne Rogers, Helen Sharp, and Jenny Preece. *Interaction Design – Beyond Human-Computer Interaction, 3rd Edition*. Wiley, 2011, pp. I–XV, 1–585. ISBN: 978-0-470-66576-3.

[22] m.c. schraefel and David Karger. "The Pathetic Fallacy of RDF". In: *International Workshop on the Semantic Web and User Interaction (SWUI) 2006*. 2006. URL: http://swui.semanticweb.org/swui06/papers/Karger/Pathetic_Fallacy.html (visited on 2015-05-30).

[23] Mohammad Ghoniem, Jean-Daniel Fekete, and Philippe Castagliola. "On the Readability of Graphs Using Node-link and Matrix-based Representations: A Controlled Experiment and Statistical Analysis". In: *Information Visualization* 4.2 (2005-07), pp. 114–135. ISSN: 1473-8716. DOI: 10.1057/palgrave.ivs.9500092. URL: http://dx.doi.org/10.1057/palgrave.ivs.9500092.

[24] Nathalie Henry, Jean-Daniel Fekete, and Michael J. McGuffin. "NodeTrix: a Hybrid Visualization of Social Networks". In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (2007-11), pp. 1302–1309. ISSN: 1077-2626. DOI: 10.1109/TVCG.2007.70582.

[25] Amalia Kallergi and Fons J. Verbeek. "Video Games for Collection Exploration: Games for and out of Data Repositories". In: *Proceedings of the 14th International Academic MindTrek Conference: Envisioning Future Media Environments*. MindTrek '10. Tampere, Finland: ACM, 2010, pp. 143–146. ISBN: 978-1-4503-0011-7. DOI: 10.1145/1930488.1930518. URL: http://doi.acm.org/10.1145/1930488.1930518.

[26] Edwin L. Hutchins, James D. Hollan, and Donald A. Norman. "Direct Manipulation Interfaces". In: *Human-Computer Interaction* 1.4 (1985-12), pp. 311–338. ISSN: 0737-0024. DOI: 10.1207/s15327051hci0104_2. URL: http://dx.doi.org/10.1207/s15327051hci0104_2.

[27] Ben Shneiderman. "Direct Manipulation: A Step Beyond Programming Languages". In: *Computer* 16.8 (1983-08), pp. 57–69. ISSN: 0018-9162. DOI: 10.1109/MC.1983.1654471.

[28] Vijay Kumar and Richard Furuta. "Visualization of Relationships". In: *Proceedings of the Tenth ACM Conference on Hypertext and Hypermedia : Returning to Our Diverse Roots: Returning to Our Diverse Roots*. HYPERTEXT '99. Darmstadt, Germany: ACM, 1999, pp. 137–138. ISBN: 1-58113-064-3. DOI: 10.1145/294469.294505. URL: http://doi.acm.org/10.1145/294469.294505.

[29] Jouni Huotari, Kalle Lyytinen, and Marketta Niemelä. "Improving Graphical Information System Model Use with Elision and Connecting Lines". In: *ACM Trans. Comput.-Hum. Interact.* 11.1 (2004-03), pp. 26–58. ISSN: 1073-0516. DOI: 10.1145/972648.972650. URL: http://doi.acm.org/10.1145/972648.972650.

[30] Ozgur Turetken and Ramesh Sharda. "Visualization of Web Spaces: State of the Art and Future Directions". In: *SIGMIS Database* 38.3 (2007-07), pp. 51–81. ISSN: 0095-0033. DOI: 10.1145/1278253.1278260. URL: http://doi.acm.org/10.1145/1278253.1278260.

[31] Jonathan D. Cohen. "Drawing Graphs to Convey Proximity: An Incremental Arrangement Method". In: *ACM Trans. Comput.-Hum. Interact.* 4.3 (1997-09), pp. 197–229. ISSN: 1073-0516. DOI: 10.1145/264645.264657. URL: http://doi.acm.org/10.1145/264645.264657.

[32] Gilles Bailly, Eric Lecolinet, and Laurence Nigay. "Flower Menus: A New Type of Marking Menu with Large Menu Breadth, Within Groups and Efficient Expert Mode Memorization". In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '08. Napoli, Italy: ACM, 2008, pp. 15–22. ISBN: 978-1-60558-141-5. DOI: 10.1145/1385569.1385575. URL: http://doi.acm.org/10.1145/1385569.1385575.

[33] Azza Abouzied, Joseph Hellerstein, and Avi Silberschatz. "DataPlay: Interactive Tweaking and Example-driven Correction of Graphical Database Queries". In: *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. UIST '12. Cambridge, Massachusetts, USA: ACM, 2012, pp. 207–218. ISBN: 978-1-4503-1580-7. DOI: 10.1145/2380116.2380144. URL: http://doi.acm.org/10.1145/2380116.2380144.

[34] Nancy H. McDonald and Michael Stonebraker. "CUPID – The Friendly Query Language". In: *ACM Pacific*. 1975, pp. 127–131.

[35] Michael Rietzler et al. "homeBLOX: Introducing Process-driven Home Automation". In: *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*. UbiComp '13 Adjunct. Zurich, Switzerland: ACM, 2013, pp. 801–808. ISBN: 978-1-4503-2215-7. DOI: 10.1145/2494091.2497321. URL: http://doi.acm.org/10.1145/2494091.2497321.

[36] NoFlo.org. *NoFlo Examples*. 2015. URL: http://noflojs.org/example/ (visited on 2015-05-31).

[37] Kirsten N. Whitley, Laura R. Novick, and Doug Fisher. "Evidence in favor of visual representation for the dataflow paradigm: An experiment testing LabVIEW's comprehensibility". In: *International Journal of Human-Computer Studies* 64.4 (2006), pp. 281–303. ISSN: 1071-5819. DOI: http://dx.doi.org/10.1016/j.ijhcs.2005.06.005. URL: http://www.sciencedirect.com/science/article/pii/S1071581905001163.

[38] FreeMind. *FreeMind – free mind mapping software*. 2014-04-12. URL: http://freemind.sourceforge.net/ (visited on 2015-05-30).

[39] Laszlo Laufer, Peter Halacsy, and Adam Somlai-Fischer. "Prezi Meeting: Collaboration in a Zoomable Canvas Based Environment". In: *CHI '11 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '11. Vancouver, BC, Canada: ACM, 2011, pp. 749–752. ISBN: 978-1-4503-0268-5. DOI: 10.1145/1979742.1979673. URL: http://doi.acm.org/10.1145/1979742.1979673.

[40] José Barranquero Tolosa et al. "Interactive web environment for collaborative and extensible diagram based learning". In: *Computers in Human Behavior* 26.2 (2010), pp. 210–217. ISSN: 0747-5632. DOI: http://dx.doi.org/10.1016/j.chb.2009.10.003.

[41] Vicenç Gómez, Andreas Kaltenbrunner, and Vicente López. "Statistical Analysis of the Social Network and Discussion Threads in Slashdot". In: *Proceedings of the 17th International Conference on World Wide Web*. WWW '08. Beijing, China: ACM, 2008, pp. 645–654. ISBN: 978-1-60558-085-2. DOI: 10.1145/1367497.1367585. URL: http://doi.acm.org/10.1145/1367497.1367585.

[42] Warren Sack. "Discourse Diagrams: Interface Design for Very Large-Scale Conversations". In: *33rd Annual Hawaii International Conference on System Sciences (HICSS-33), 4–7 January, 2000, Maui, Hawaii, USA*. 2000. DOI: 10.1109/HICSS.2000.926717. URL: http://dx.doi.org/10.1109/HICSS.2000.926717.

[43] Lucid Software Inc. *Lucidchart – Flow Chart Maker & Online Diagram Software*. 2015. URL: https://www.lucidchart.com/ (visited on 2015-06-15).

[44] Subatomic Software. *Audulus – Design Sound from First Principles*. 2015. URL: http://audulus.com/ (visited on 2015-05-30).

[45] Michel Beaudouin-Lafon. "Instrumental Interaction: An Interaction Model for Designing post-WIMP User Interfaces". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '00. The Hague, The Netherlands: ACM, 2000, pp. 446–453. ISBN: 1-58113-216-6. DOI: 10.1145/332040.332473. URL: http://doi.acm.org/10.1145/332040.332473.

[46] W3C. *SPARQL 1.1 Overview – W3C Recommendation 21 March 2013*. 2013. URL: http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/.

[47] Frederik Hogenboom et al. "RDF-GL: A SPARQL-Based Graphical Query Language for RDF". English. In: *Emergent Web Intelligence: Advanced Information Retrieval*. Ed. by Richard Chbeir et al. Advanced Information and Knowledge Processing. Springer London, 2010, pp. 87–116. ISBN: 978-1-84996-073-1. DOI: 10.1007/978-1-84996-074-8_4. URL: http://dx.doi.org/10.1007/978-1-84996-074-8_4.

[48] Alistair Russell et al. "NITELIGHT: A Graphical Tool for Semantic Query Construction". In: *Semantic Web User Interaction Workshop (SWUI 2008)*. Event Dates: 5th April 2008. 2008-04. URL: http://eprints.soton.ac.uk/264975/.

[49] WikiData.org. *Q5597179 – graphical user interface element*. 2015-01-09. URL: https://www.wikidata.org/wiki/Q5597179 (visited on 2015-06-10).

[50] Alvaro Graves. *Visual RDF*. 2015. URL: http://graves.cl/visualRDF/ (visited on 2015-06-10).

[51] Bryan O'Sullivan. "Making Sense of Revision-control Systems". In: *Commun. ACM* 52.9 (2009-09), pp. 56–62. ISSN: 0001-0782. DOI: 10.1145/1562164.1562183. URL: http://doi.acm.org/10.1145/1562164.1562183.

[52] Hsiang-Ting Chen, Li-Yi Wei, and Chun-Fa Chang. "Nonlinear Revision Control for Images". In: *ACM Trans. Graph.* 30.4 (2011-07), 105:1–105:10. ISSN: 0730-0301. DOI: 10.1145/2010324.1965000. URL: http://doi.acm.org/10.1145/2010324.1965000.

[53] GitHub Inc. *Network Graph – alangrafu/visualRDF*. 2015. URL: https://github.com/alangrafu/visualRDF/network (visited on 2015-06-10).

[54] Raphaël Hoarau and Stéphane Conversy. "Augmenting the Scope of Interactions with Implicit and Explicit Graphical Structures". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '12. Austin, Texas, USA: ACM, 2012, pp. 1937–1946. ISBN: 978-1-4503-1015-4. DOI: 10.1145/2207676.2208337. URL: http://doi.acm.org/10.1145/2207676.2208337.

[55] Harold W. Thimbleby, Paul A. Cairns, and Matt Jones. "Usability analysis with Markov models". In: *ACM Trans. Comput.-Hum. Interact.* 8.2 (2001), pp. 99–132. DOI: 10.1145/376929.376941. URL: http://doi.acm.org/10.1145/376929.376941.

[56] George W. Furnas. "Generalized Fisheye Views". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '86. Boston, Massachusetts, USA: ACM, 1986, pp. 16–23. ISBN: 0-89791-180-6. DOI: 10.1145/22627.22342. URL: http://doi.acm.org/10.1145/22627.22342.

[57] Doug Schaffer et al. "Navigating Hierarchically Clustered Networks Through Fisheye and Full-zoom Methods". In: *ACM Trans. Comput.-Hum. Interact.* 3.2 (1996-06), pp. 162–188. ISSN: 1073-0516. DOI: 10.1145/230562.230577. URL: http://doi.acm.org/10.1145/230562.230577.

[58] Masashi Toyoda and Etsuya Shibayama. "Hyper Mochi Sheet: A Predictive Focusing Interface for Navigating and Editing Nested Networks Through a Multi-Focus Distortion-Oriented View". In: *Proceeding of the CHI '99 Conference on Human Factors in Computing Systems: The CHI is the Limit, Pittsburgh, PA, USA, May 15–20, 1999*. 1999, pp. 504–511. DOI: 10.1145/302979.303145. URL: http://doi.acm.org/10.1145/302979.303145.

[59] Benjamin B. Bederson. "The promise of zoomable user interfaces". In: *Behaviour & IT* 30.6 (2011), pp. 853–866. DOI: 10.1080/0144929X.2011.586724. URL: http://dx.doi.org/10.1080/0144929X.2011.586724.

[60] Christian Tominski, James Abello, and Heidrun Schumann. "CGV - An interactive graph visualization system". In: *Computers & Graphics* 33.6 (2009), pp. 660–678. DOI: 10.1016/j.cag.2009.06.002. URL: http://dx.doi.org/10.1016/j.cag.2009.06.002.

[61] Thomas Baudel. "From information visualization to direct manipulation: extending a generic visualization framework for the interactive editing of large datasets". In: *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology, Montreux, Switzerland, October 15-18, 2006*. 2006, pp. 67–76. DOI: 10.1145/1166253.1166265. URL: http://doi.acm.org/10.1145/1166253.1166265.

[62] Satu Elisa Schaeffer. "Graph clustering". In: *Computer Science Review* 1.1 (2007), pp. 27–64. DOI: 10.1016/j.cosrev.2007.05.001. URL: http://dx.doi.org/10.1016/j.cosrev.2007.05.001.

[63] Robert van Liere and Wim C. de Leeuw. "GraphSplatting: Visualizing Graphs as Continuous Fields". In: *IEEE Trans. Vis. Comput. Graph.* 9.2 (2003), pp. 206–212. DOI: 10.1109/TVCG.2003.1196007. URL: http://doi.ieeecomputersociety.org/10.1109/TVCG.2003.1196007.

[64] Michael Burch and Daniel Weiskopf. "A Flip-Book of Edge-Splatted Small Multiples for Visualizing Dynamic Graphs". In: *The 7th International Symposium on Visual Information Communication and Interaction, VINCI '14, Sydney, NSW, Australia, August 5–8, 2014*. 2014, p. 29. DOI: 10.1145/2636240.2636839. URL: http://doi.acm.org/10.1145/2636240.2636839.

[65] M. S. Marshall, I. Herman, and G. Melançon. "An object-oriented design for graph visualization". In: *Software: Practice and Experience* 31.8 (2001), pp. 739–756. ISSN: 1097-024X. DOI: 10.1002/spe.385. URL: http://dx.doi.org/10.1002/spe.385.

[66] Tatiana von Landesberger et al. "Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges". In: *Comput. Graph. Forum* 30.6 (2011), pp. 1719–1749. DOI: 10.1111/j.1467-8659.2011.01898.x. URL: http://dx.doi.org/10.1111/j.1467-8659.2011.01898.x.

[67] Mark T. Maybury and Wolf Wahlster. "Readings in Intelligent User Interfaces". In: ed. by Mark T. Maybury and Wolfgang Wahlster. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998. Chap. Intelligent User Interfaces: An Introduction, pp. 1–13. ISBN: 1-55860-444-8. URL: http://dl.acm.org/citation.cfm?id=286013.286437.

[68] Andries van Dam. "Beyond WIMP". In: *IEEE Computer Graphics and Applications* 20.1 (2000), pp. 50–51. DOI: 10.1109/38.814559. URL: http://doi.ieeecomputersociety.org/10.1109/38.814559.

[69] Robert J. K. Jacob, Leonidas Deligiannidis, and Stephen Morrison. "A Software Model and Specification Language for Non-WIMP User Interfaces". In: *ACM Trans. Comput.-Hum. Interact.* 6.1 (1999), pp. 1–46. DOI: 10.1145/310641.310642. URL: http://doi.acm.org/10.1145/310641.310642.

[70] Robert J. K. Jacob et al. "Reality-based interaction: a framework for post-WIMP interfaces". In: *Proceedings of the 2008 Conference on Human Factors in Computing Systems, CHI 2008, 2008, Florence, Italy, April 5-10, 2008*. 2008, pp. 201–210. DOI: 10.1145/1357054.1357089. URL: http://doi.acm.org/10.1145/1357054.1357089.

[71] Reif Larsen. *Schematic map of Rome subway station for blind people*. 2013-07-04. URL: http://reiflarsen.tumblr.com/post/54580770784/schematic-map-of-rome-subway-station-for-blind (visited on 2015-05-29).

[72] Sebastian Schmidt et al. "A set of multi-touch graph interaction techniques". In: *ACM International Conference on Interactive Tabletops and Surfaces, ITS 2010, Saarbrücken, Germany, November 7-10, 2010*. 2010, pp. 113–116. DOI: 10.1145/1936652.1936673. URL: http://doi.acm.org/10.1145/1936652.1936673.

[73] Deepamehta.de. *Welcome to DeepaMehta*. 2012-05-22. URL: http://www.deepamehta.de/en (visited on 2015-05-29).

[74] Jörg Richter, Max Völkel, and Heiko Haller. "DeepaMehta - A Semantic Desktop". In: *Proceedings of the ISWC 2005 Workshop on The Semantic Desktop - Next Generation Information Management & Collaboration Infrastructure. Galway, Ireland, November 6, 2005*. 2005. URL: http://ceur-ws.org/Vol-175/30_dm_poster.pdf.

[75] IBM Emerging Technology. *Node-RED*. 2015. URL: http://nodered.org/ (visited on 2015-06-11).

[76] Giuseppe Di Battista et al. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999. ISBN: 0-13-301615-3.

[77] Shixia Liu et al. "A survey on information visualization: recent advances and challenges". In: *The Visual Computer* 30.12 (2014), pp. 1373–1393. DOI: 10.1007/s00371-013-0892-3. URL: http://dx.doi.org/10.1007/s00371-013-0892-3.

[78] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. "D^3 Data-Driven Documents". In: *IEEE Trans. Vis. Comput. Graph.* 17.12 (2011), pp. 2301–2309. DOI: 10.1109/TVCG.2011.185. URL: http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.185.

[79] I. Fette and A. Melnikov. *The WebSocket Protocol*. RFC 6455. http://www.rfc-editor.org/rfc/rfc6455.txt. RFC Editor, 2011-12. URL: http://www.rfc-editor.org/rfc/rfc6455.txt.

[80] Alan Wexelblat and Pattie Maes. "Footprints: History-Rich Tools for Information Foraging". In: *Proceeding of the CHI '99 Conference on Human Factors in Computing Systems: The CHI is the Limit, Pittsburgh, PA, USA, May 15-20, 1999*. 1999, pp. 270–277. DOI: 10.1145/302979.303060. URL: http://doi.acm.org/10.1145/302979.303060.

[81] Brian Amento et al. "Experiments in social data mining: The TopicShop system". In: *ACM Trans. Comput.-Hum. Interact.* 10.1 (2003), pp. 54–85. DOI: 10.1145/606658.606661. URL: http://doi.acm.org/10.1145/606658.606661.

[82] Mark O. Riedl and Robert St. Amant. "Toward automated exploration of interactive systems". In: *IUI*. 2002, pp. 135–142. DOI: 10.1145/502716.502738. URL: http://doi.acm.org/10.1145/502716.502738.

[83] Harold Thimbleby. "Action graphs and user performance analysis". In: *Int. J. Hum.-Comput. Stud.* 71.3 (2013), pp. 276–302. DOI: 10.1016/j.ijhcs.2012.10.014. URL: http://dx.doi.org/10.1016/j.ijhcs.2012.10.014.

[84] Melody Y. Ivory and Marti A Hearst. "The State of the Art in Automating Usability Evaluation of User Interfaces". In: *ACM Comput. Surv.* 33.4 (2001-12), pp. 470–516. ISSN: 0360-0300. DOI: 10.1145/503112.503114. URL: http://doi.acm.org/10.1145/503112.503114.

[85] W3C. *RDF 1.1 Primer — Working Group Note 24 June 2014*. 2014. URL: http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/.