

# Diagram Editing on Interactive Displays Using Multi-Touch and Pen Gestures

Mathias Frisch, Jens Heydekorn, Raimund Dachsel

User Interface & Software Engineering Group  
Otto-von-Guericke-University Magdeburg,  
39106 Magdeburg, Germany  
{mfrisch, jheyde, dachsel}@isg.cs.uni-magdeburg.de

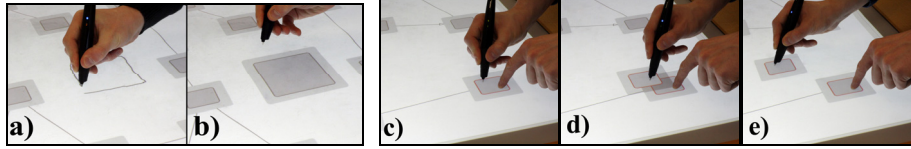
**Abstract.** Creating and editing graphs and node-link diagrams by means of digital tools are crucial activities in domains such as software or business process modeling. However, these tools have several drawbacks with regard to interaction techniques and usability. In order to address these issues, we investigate the promising combination of pen and multi-touch input on interactive displays. In this work, we contribute a gesture set to make the interaction with diagrams more efficient and effective by means of pen and hand gestures. Thereby, two prevalent mental models are supported: *structural editing* and *sketching*. The gesture set is based on the results of a previous pilot study asking users for suggestions to accomplish diagram editing tasks on tabletops. In this paper, we provide a careful analysis of the resulting user-elicited gestures. We propose solutions to resolve ambiguities within this gesture collection and discuss design decisions for a comprehensible diagram editor. We also present the multi-touch and pen gesture set as a fully implemented prototype for diagram editing on interactive surfaces.

**Keywords:** interaction techniques for diagrams, user-centered approach, mental models, multi-touch and pen interaction, digital pen, sketching, tabletops

## 1 Introduction

Current research in the field of creating diagrams is often focused on methods and algorithms to produce and to process diagrams in an automatic way. This comprises, for example, the generation and transformation of diagrams or automatic layout algorithms. However, creating graph and node-link diagrams from scratch and editing them manually in digital editors are very common activities in domains such as software modeling, business process modeling, project management and simulation.

With regard to interaction techniques, there basically exist two approaches to create and edit graphical node-link diagrams. There are digital editors to build diagrams, commonly based on formal notations, by means of *structural editing*. They are usually limited to traditional point and click interaction techniques, e.g. dragging and dropping diagram elements from a toolbar or switching modes by means of buttons or menus. In domains such as software development these editing tools are often perceived as constrictive and inflexible [8], [15].



**Fig. 1.** Creating a node by sketching (a), copying an already existing node by means of a bimanual hold and drag gesture: hold and drag have to start from the interior of the node (b). When the pen is dragged, the copy appears (c) and can be rearranged (d).

This is one reason why freehand *sketching* is often preferred. Beyond that, drawing with pens is a more natural human capability. Therefore, in many situations diagrams are drafted on whiteboards or flip charts [6]. With these traditional setups, diagrams can be produced in an informal and ad hoc way. As a consequence, they often have to be remodeled in digital tools, which is a time consuming process. Digital sketching tools try to solve this problem and additionally offer techniques such as rearranging, grouping or scaling elements on electronic whiteboards or Tablet PCs [8], [15]. Usually, these kinds of applications solely support pen or single touch interaction. Hence, for tasks such as zooming or changing types of elements users still have to navigate through menus or click buttons, which disrupts the editing process.

In contrast to that, we are investigating the combination of multi-touch and pen interaction for the domain of diagram editing on interactive displays. We expect that this approach is able to make the handling of diagrams more efficient and effective. Certain tasks such as rearranging or deleting elements can be accomplished simultaneously by using both hands. Beyond that, it is conceivable to switch modes by means of hand gestures without interrupting the current workflow (see Figure 1 c-e). Moreover, gestures with a physical or metaphorical nature can make the interaction more natural, which is important especially for novices.

Devices with pen and multi-touch support occur in different form factors, reaching from handheld tablets to huge wall size displays. Some of them are commercially available currently [22], and it can be expected that they will be increasingly applied.

In order to start investigating how node-link diagrams can be edited with touch and pen interaction on interactive displays, we conducted a pilot study [14]. The study applied a user-centered design approach. As a result, a collection of user-defined gestures was identified. However, the collection included several ambiguities, because in some cases the participants assigned the same gestures to different tasks.

At the heart of this work, we solve these ambiguities and contribute a successfully implemented gesture set which is based on this collection. The set allows the combination of pen and hand input and supports both approaches mentioned above – *structural editing* and *sketching*. The paper is structured as follows: After a brief summary of the design and the results of the pilot study, we present our methodology how to resolve the conflicts within the collection of gestures. We determine design goals, analyze the elicited gestures and propose options which can be applied to resolve the ambiguities. After that, we present the implementation of the gesture set in detail and give an insight into the architecture of the prototype. Finally, we discuss how the prototype can be extended to a diagram editor for more complex diagram notations and give an outline of future work.

## **2 Related Work**

### **2.1 Digital Diagram Sketching**

Various tools have been realized for sketching on electronic whiteboards or Tablet PCs. In [23], several domain-independent pen interaction techniques are presented. They cover copying, pasting or scaling elements, and the respective prototype is also capable of recognizing node-link diagrams. Beyond that, there are digital sketching tools such as presented in [8], [15], [17] or [5]. They are tailored to the domain of software engineering and convert sketches to diagram notations such as UML. In contrast to the prototype presented in this paper, they solely support pen interaction and do not consider additional modalities. Especially in the domain of software development, several studies have been conducted [8], [6], [9]. They investigate how and for what purposes software designers use whiteboards for diagram sketching. As a result, design principles for digital sketch applications were concluded. However, to our knowledge, the combination of multi-touch and pen interaction has not been studied yet in the domain of diagram sketching.

### **2.2 Multi-Touch Gestures for Tabletops**

Over the past years, various multi-touch enabled interactive displays have been developed. Some of these devices are already commercially available. They are applying different approaches to detect touch events from users, such as computer vision [18], [22] or capacitive technology [10], [25]. These devices have been used to investigate and to propose a number of multi-touch gestures for particular purposes.

Gestures proposed in [26] cover panning, scaling, rotating, and picking up objects. In [30] a set of gestures for multi-user tabletops is presented. It includes gestures for rotating, collecting objects, and for private viewing. Wu et al. [30] describe design principles for gesture design and built a prototype of a publishing application to illustrate the usage of their principles. Other research on gestures can be found in [20], [26] or [4]. Amongst others, they are including whole-hand gestures on interactive surfaces. Concerning the interaction with node-link diagrams, Dwyer et al. [11] conducted a study to investigate how users would layout graphs on tabletops in a manual way. They observed several gestures participants applied to reposition and group nodes. However, pen interaction and the combination of pen and touch were not considered. Besides that, the given tasks were limited to layout purposes and they do not propose a particular gesture set for a respective diagram editor application. Our contribution instead covers diagram editing tasks, such as creating, deleting or copying of nodes and edges.

### **2.3 Combination of Pen and Touch Interaction**

In addition to the aforementioned devices, there are technical systems which explicitly support multi-touch and pen input simultaneously. Flux [19] is a tabletop of

this type and can be tilted to horizontal, vertical and slanted positions. For the prototype presented here we are using a similar technical approach which will be discussed in Section 6. Another vertical-only solution is INTOI [3], including the capability of pen and hand gesture recognition.

Concerning interaction techniques, the combination of pen input and single touch is investigated by Yee [31]. He proposes panning the canvas with the finger while drawing with the pen. Beyond that, the usage of digital pens and multi-touch on tabletops has been studied in Brandl et al. [2]. They suggest general design principles and present interaction techniques for a graphics application. Both works consider Guiard's Kinematic Chain Model [16], which proposes principles for the division of roles between hands: the dominant hand moves within the frame of reference set by the non-dominant hand, the non-dominant hand precedes the dominant hand, and the dominant hand performs more precise actions.

#### **2.4 Research on User-defined Gestures**

All of the gesture sets mentioned above are designed by experts. In contrast to that, there are approaches to elicit gestures from users. For that, Nielsen et al. [24] propose a procedure of four steps. Initially, the system functions should be found, which the gestures will have to communicate. Then participants are asked to perform spontaneous gestures for each of the functions. This is done under video surveillance. As a next step, the video material is evaluated to extract the gesture vocabulary. Finally, the elicited gestures are benchmarked. In their work, Epps et al. [12], Micire et al. [21] and Wobbrock et al. [28] used this approach for their studies. The latter conducted a study to develop a user-defined set of general one-hand and two-hand gestures and presented a respective gesture taxonomy.

The gesture set introduced in this paper is the result of our study with similar design [14]. In contrast to the aforementioned approaches, we studied both, multi-touch and pen interaction for the domain of diagram editing. A short summary of our study is given in the following section.

### **3 Eliciting Gestures for Diagram Editing**

We conducted a pilot study applying a user participatory approach based on the work of Nielsen et al. [24] (see Section 2.4) to investigate how users would edit node-link diagrams on an interactive tabletop display [14]. To our knowledge, this was the first work which applied this approach to the domain of diagram editing. Particular editing tasks were given to the participants, and they were asked to perform spontaneous hand and pen gestures to solve these tasks. From this, we were capable to get an insight into preferred modalities and prevalent mental models. However, when applying such an approach, one should bear in mind that users are not interaction designers. Therefore, our goal was not to come up with a final user-defined gesture set which can be implemented in a straightforward way. Our main purpose was rather

to involve the users in the design process right from the beginning and to get a feeling for preferred interaction procedures in this particular domain.

The result of the study was a collection of pen and hand gestures elicited from users. The gesture set presented in this paper (see Table 1) is based on this collection. Furthermore, we made several additional observations concerning participants' remarks and behavior. These results served as a starting point for designing and implementing the gesture set described later in this paper.

### **3.1 Design of the Pilot Study**

The pilot study applied a within-subjects design. Every participant was asked to complete 14 basic editing tasks in a fixed order (see left column of Table 1 for a condensed overview). Besides elementary tasks like creating and deleting elements, we also asked for more specific procedures such as copying a sub-graph (see task 8 in Table 1) and changing a solid edge to a dashed one (see task 9 in Table 1). In order to produce results which are applicable to a variety of visual diagram languages, we used an elementary variant of node-link diagrams. Nodes were represented by simple rectangles and connected by directed or undirected edges.

### **3.2 Participants and Apparatus**

Seventeen participants took part in the study. All of them had a solid background in software engineering. Therefore, they were familiar with visual node-link diagrams such as UML. They were neither expert users of visual diagram editors nor UI experts. The study was conducted on a tabletop display which combines multi-touch with pen interaction (see Section 6, Figure 4). Users' inputs were recorded by video camera, by the vision system of the tabletop and by taking notes during the procedure.

### **3.3 Tasks and Procedure**

The display was horizontally divided into two areas. The lower area displayed a diagram in the original state, and the upper area showed its final state. For each task, the participants were asked to transfer the diagram from the original state to the final state by performing a spontaneous gesture inside the lower area. Thereby, they got no feedback from the system. Participants performed gestures with three different interaction techniques per task: with one hand (whereby they could use all fingers of the hand), with two hands, and with pen (held in their dominant hand). For the latter, it was free to them to combine the pen gesture with all fingers of the non-dominant hand. Furthermore, each subject was asked to start with the variant that he or she considered as most suitable for the respective task. After each task, the participants were asked to fill out a questionnaire concerning the suitability of each interaction technique.

### 3.4 Results

**Gestures.** We analyzed a total number of 658 gestures. A result of this first analysis was that no task was solved by a single or unique gesture among all participants. The absolute number of variations was highest for the *copy* task (33 variations), which can be certainly attributed to its rather abstract nature. The lowest amount of variations was elicited for the *select node* task (13 variations). Overall, we observed that in general one-hand and pen modality was preferred for solving most of the tasks. However, there were also situations where bimanual interaction was preferred, such as zooming and scaling, copying elements or achieving a mode switch by resting the non-dominant hand on the background.

**Mental Models.** We identified two classes of basic mental models the participants relied on while they solved the given tasks: *sketching* and *structural editing*. The *sketching* class is characterized by physical imitation of real-world ad hoc sketching such as drawing a node or edge. Beyond that, gestures with a metaphorical nature fall into this class as well, e.g. deleting an element by wiping (task 5 in Table 1) or creating a dashed edge by performing a “rake” gesture (see task 9 in Table 1). However, the *sketching* class is not stringently associated with the usage of a pen. In contrast, the *structural editing* class is oriented more towards digital diagram editors and applies a higher grade of abstraction to phrase the intention. The associated gestures, e.g. for copying elements, therefore have a more abstract nature.

**Gesture Collection.** We used these results and observations from the study to identify the top candidates for each task and created a respective collection of pen and multi-touch gestures. Where appropriate, we assigned more than just one gesture to a task. For example, we considered both prevalent mental models mentioned above. In order to support both approaches, gestures based on *sketching* as well as gestures based on *structural editing* were assigned to respective tasks.

## 4 Expert Analysis of the Elicited Gestures

In Table 1 our proposed pen and hand gesture set for node-link diagram editing is depicted. It is based on the elicited collection from the pilot study described above. Where possible, it still supports both approaches – *sketching* and *structural editing*. In Table 1 gestures which correspond to the *sketching* approach have a gray background. However, the proposed set comprises several ambiguities and conflicts, as in some cases the same gesture was assigned to different tasks by the participants. This is no surprise, because we did not explicitly ask them to be consistent and to apply a particular gesture just once. Nevertheless, this prevents a straightforward implementation of the gestures. Therefore, preliminary considerations and a prior analysis by experts are necessary. In the following sub-sections we describe our methodology to resolve these ambiguities and to design a system which realizes the gesture set.

## 4.1 Design Goals

Before conducting a deeper analysis of the existing conflicts, we developed general design goals for gestural diagram editing. They are based on the results and the observations from the study to preserve the user-centered approach:

- G1 Preserving support for both mental models – *sketching* and *structural editing* – as they are essential for a satisfying system for most users in this domain.
- G2 Keeping the introduction of new special gestures to a minimum by reusing proposed identical gestures for different tasks [30].
- G3 Providing ad hoc and direct creation of content without time-consuming navigation through options offered by menus (as suggested in [8]).

## 4.2 Resolving Conflicts

In particular, the following conflicts occur within the set of elicited gestures, because users assigned identical gestures to the same task:










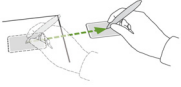


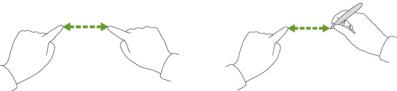

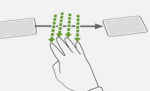
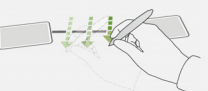
- C1 Creating an edge vs. moving a node: When a node is touched and the finger or pen starts moving, it is not clear if the node should be dragged or an edge should be inserted.
- C2 Selecting nodes vs. creating an edge by tapping: When several nodes are sequentially tapped, it is ambiguous if they should be selected or if edges should be created between them.
- C3 Drawing an edge vs. dragging an edge: Within the *create edge* task it is not obvious if an edge should be dragged to the target node (like a rubber band) or if the edge is meant to be sketched like on a whiteboard.

In some cases during the pilot study, participants realized they were suggesting conflicting gestures and indicated how they would resolve them. We took these suggestions and identified four general options to resolve the aforementioned ambiguities with regard to the design goals G1 - G3. These are general solutions and can also be applied in the design process of other systems.

**Additional gestures.** A trivial solution for all conflicts is the definition of additional gestures. In that way, for each task a specific gesture can be assigned. However, this approach would contradict design goal G2 and would require users' effort for memorizing multiple unique gestures.

**Mode switch.** Similarly to WIMP (Windows, Icons, Menus, Pointer), static menus and variations of buttons could be provided for mode switches. However, they can be difficult to reach on large surfaces, especially if they are placed on traditional places

**Table 1.** The gesture set for diagram editing based on the user-elicited gesture collection [14]. It considers structural editing (white background) and sketching (gray background). All gestures depicted with pen can also be performed with a single finger.

Task	Gesture		
1. Create Node	 single tap	 copy node by holding and dragging	 drawing node
2. Create Edge	 „dragging“ edge	 sequential tapping	 drawing (un)directed edge
3. Select Node(s)	 single tap	 encircle node(s)	
4. Move Node(s)			
5. Delete Node or Edge	 dragging to off-screen	 wiping	
6. Scale Node			
7. Zoom Diagram			
8. Copy Sub-graph	 hold and drag (after select nodes)		
9. Change Edge from Solid to Dashed	 „rake“-gesture	 sequential crossing	



such as the top border of the display. The usage of context menus is more appropriate. However, this is contrary to design goal G3 and an additional gesture for invocation would be necessary. Furthermore, the non-dominant hand can be placed on the background to cause a mode switch while the actual gesture is performed with the dominant hand. However, this is only applicable for one-hand gestures and there must be free and reachable background.

**Distinction of input modalities.** Another way to resolve the conflicts is to distinguish between different input modalities - in our case between touch and pen. This would mean, for example, that drawing elements is solely assigned to the pen, whereas functions such as dragging or scaling elements can only be done with fingers. However, we did not identify a general preference for one modality and would therefore not force users to switch between hand and pen interaction.

**Graphical contexts.** The user interface can offer additional graphical regions which are sensitive to gestures. These regions can represent different contexts. For example, the border of a node is explicitly touchable to create edges, whereby the interior of the node is used for dragging or selecting. This approach can solve the conflicts C1 and C2.

## 5 The Gesture Set

We successfully implemented the proposed gesture set depicted in Table 1. In order to resolve the conflicts identified in Section 4, the set of interactive elements has been extended. We decided to add an interactive border region around every node, which serves as an additional graphical context (e.g. see Figure 2). It can be used to insert edges by tapping or drawing. Every time it is touched with the pen or fingers, the border region changes its color to give a visual feedback. In that way, all gestures can be performed by means of finger or pen and in an ad hoc way – a prior mode switch is not necessary. Details are explained in the following subsections, where we present the gesture set in the way it is implemented.

### 5.1 Creating Diagram Elements

**Creating Nodes.** Nodes can be created in two ways: by sketching their outlines (see Figure 1 a, b) or by tapping with the finger or pen on the background. The latter corresponds to *structural editing*, and a standard-sized node is created at the place where the tap occurs. In order to avoid that nodes are created by unintended touches, we propose that the touches have to be held for a short time delay until the node appears. Beyond that, nodes can be created by copying already existing nodes. This is done through a bimanual gesture. The non-dominant hand holds the node and the copy is dragged from it with a finger of the dominant hand or the pen. Both parts of this gesture - hold and drag - have to start from the interior of the node (see Figure 1 c-e).

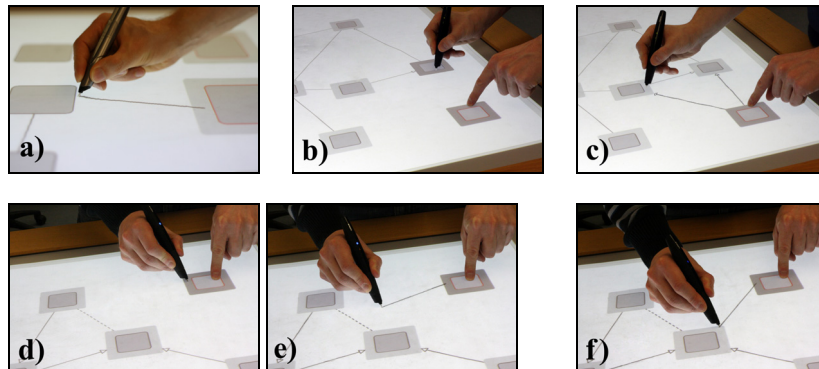
**Creating Edges.** We implemented three solutions to create an edge between two nodes: sketching, dragging and tapping.

*Sketching.* Edges can be sketched by starting a drag gesture from the interactive border of a node (see Figure 2 a). Thereby, drawing a simple line results in an undirected edge and drawing a line with arrow head results in a directed edge.

*Dragging.* The second method is to drag edges in a rubber band style, like in structural editors (see Figure 2 d-f). For that, the same bimanual gesture as for the copy task is applied: holding the node with one finger and dragging the edge with another one. In contrast to the copy task, the dragging gesture has to start from the interactive border region of the node. This modifies the proposed gestures of Table 1 just slightly, but solves C1 by means of the additional graphical context.

*Tapping.* Edges can also be created by tapping. In particular, there are two techniques: *sequential tapping* and *holding & tapping*. With *sequential tapping* an edge is created between nodes when their border regions are tapped sequentially. By means of *holding & tapping* the border region of a node can be held with a finger of the non-dominant hand. Tapping borders of other nodes with a finger of the dominant hand or the pen results in an edge (see Figure 2 b-c). If many edges are going from the same node, this bimanual gesture can serve to create them in a fast way.

In general, these tapping techniques are beneficial for connecting nodes on rather large interactive surfaces, such as tabletops or wall-sized displays, as dragging edges between nodes located far away from each other can be cumbersome. Of course, the nodes have to be in reach of both arms.



**Fig. 2.** Creation of an edge by sketching, sketching has to start on the interactive border region (a), creation of an edge by *holding & tapping*: the border of a node is held (b), tapping the border of other nodes creates edges (c), dragging edge by *holding & dragging*: dragging has to start from the border region (d), edge can be dragged like a rubber band (e, f).

## 5.2 Selecting and Moving Nodes

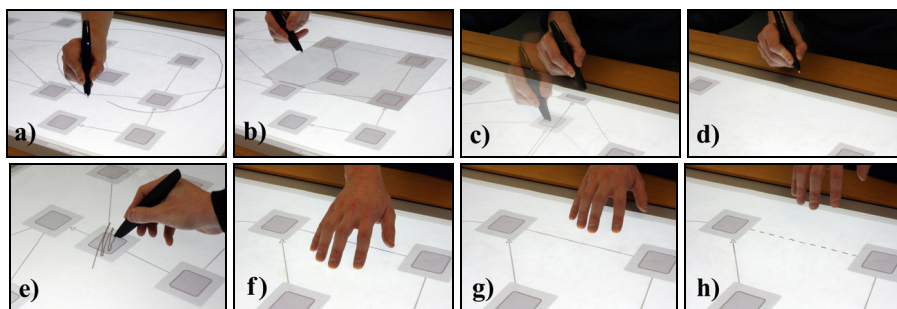
Nodes can be selected by tapping their interior or by encircling them. The latter rather corresponds to the sketching approach. If more than two nodes are selected they are aggregated to a sub-graph (see Figure 3 a, b). The aggregation can be undone by shortly tapping the background. Moving single nodes and sub-graphs is possible by touching their interior and dragging them to the appropriate position by means of finger or pen. It is also possible to copy sub-graphs by applying the aforementioned copy-gesture for nodes (*holding & dragging*). As a result, all nodes within the sub-graph and edges between nodes of the sub-graph are copied. Edges going from and to the sub-graph are not present in the copied sub-graph.

## 5.3 Deleting Diagram Elements

All kinds of diagram elements can be deleted by performing a wipe gesture, like on whiteboards (see Figure 3 e). Thereby, the gesture has to start on the background and all elements intersecting its bounding rectangle are deleted when the finger or pen is lifted. For deleting single nodes or sub-graphs it is also possible to drag them to off screen; an approach which we observed several times during the pilot study. In our current implementation nodes can be dragged “from the screen” in all four directions. When nodes are dragged out of the visible part of the workspace, the nodes (and all their associated edges) are deleted when the finger or pen is lifted (see Figure 3 c, d).

## 5.4 Changing the Type of an Edge

We implemented two different gestures of metaphorical nature to change solid edges to dashed ones. The task can be performed with a multi-touch “rake” gesture. Thereby, three or four fingers are crossing the edge in parallel. After lifting the fingers, the appearance of the edge is changed (see Figure 3 f-h). Besides that, it is also possible to change the edge by crossing it three times sequentially. In this way



**Fig. 3.** Creating a sub-graph by encircling (a, b), deleting a node by dragging to off screen (c, d) and by wiping (e), changing a solid edge to a dashed one by “rake” gesture (f-h).

users are also able to perform the task with one finger or pen. In order to change a dashed edge back to a solid one, the “rake” gesture can be performed again or an edge can be created on top of the existing one. This gesture is an example for a shortcut gesture which is applied to perform a domain-specific diagram editing task. For more suggestions concerning that type of gesture, please see Section 7.

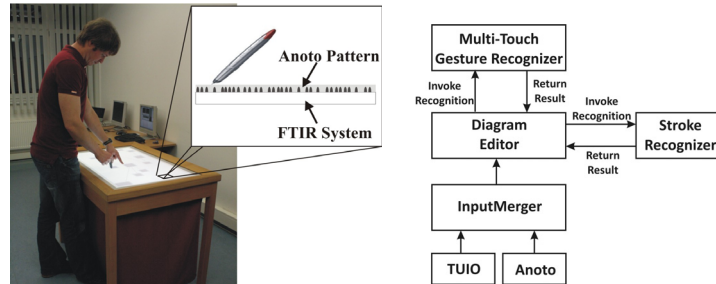
## 5.5 Scaling, Zooming and Panning

A pinch gesture with two fingers [25], or finger and pen respectively, is applied for scaling and zooming operations. For zooming the whole diagram, the gesture has to be performed on the background. For scaling it has to be performed on the respective node or sub-graph. Panning a diagram was not part of the tasks of the pilot study. For that, we added an additional gesture. Performing a multi-touch drag gesture with five fingers activates panning, similar to [2]. This gesture does not have to be performed on the background; panning is possible even if some elements are touched. This approach can be beneficial, especially in large diagrams with a huge amount of elements located very close to each other.

## 6 Technical Implementation

For our prototype we are using a multi-touch enabled tabletop system based on Frustrated Total Internal Reflection (FTIR) [18]. The display has a size of 102 x 77 cm and a resolution of 1280 x 800 pixels. For recognizing touch input we are using the Community Core Vision Toolkit [7]. It sends events by means of the TUIO protocol [27] which delivers basic information for each touch, such as position and ID. Beyond that, our hardware is able to distinguish between touch and pen input. To achieve that, the Anoto technology [1] is applied for pen interaction which is similar to [19]. Digital pens read the Anoto pattern that is printed onto the surface of the display and stream the pen coordinates to our software via Bluetooth (see Figure 4 left). The FTIR approach registers touch events within the IR spectrum. As the digital pens emit an IR signal, pens also produce touch events in addition to the Anoto messages. Therefore, a pen is recognized by the system as finger *and* pen. In order to avoid this, a preceding software component is necessary (*InputMerger*, see Figure 4 right) before the events are sent to the editor application. The *InputMerger* ensures that during pen interaction the touch events arising from the pen are omitted and only the respective Anoto events are transmitted to the application. Beyond that, it converts incoming events (TUIO and Anoto) to a uniform data format.

We implemented our own gesture recognizer. Incoming events are clustered by timestamp, distance and graphical context. After recognizing basic gestures, such as dragging, holding or tapping, these gestures are combined, e.g. to *hold and drag* gestures. When a drag gesture is performed with one finger or pen, the respective stroke data is sent to a sketch recognizer. However, this happens just if the gesture is performed on the background or if it is started on the interactive border region of a node as these activities can result in a sketched shape (see Figure 4 right).



**Fig.4.** Tabletop setup with Anoto pattern on surface (left), Software Architecture (right): TUIO and Anoto events are converted to a consistent data format by *InputMerger*. The application invokes gesture and stroke recognition.

## 7 Discussion

The presented gesture set and prototype covers the most basic functionalities for diagram editing by means of user-defined gestures. It can serve as a basis for further research in this area. In this section, we discuss ways of extending the prototype to a full-value editor for more complex diagram notations.

**Shortcut gestures:** We propose to add additional shortcut gestures. On the one hand, this can be done for general tasks such as undo and redo or cut and paste (e.g. as proposed by [29]). On the other hand, gestures for more domain-specific diagram editing tasks can be added. As a first example we implemented the “rake” gesture to change a solid edge to a dashed one (see 5.4). Further functionalities that can be supported by single pen and hand gestures are, for example the creation of directed edges going back and forth between two nodes or the creation of diagram layouts. For the latter, gestures for generating automatic layouts of whole diagrams or parts of the graph can be applied. We also suggest determining layout constraints manually by means of hand gestures. Early explorations on this topic can be found in [11].

**Contextual assistance:** We propose contextual assistance (e.g. adapted menus) for complex notations consisting of a huge amount of different types of nodes and edges. Of course, contextual assistance can be applied to automatically suggest valid diagram elements. For example, this can be beneficial in situations where elements shall be created, but some of them would break syntactic rules. Beyond that, we propose to apply contextual assistance to gestures. When a gesture is started, the system could give suggestions in which way the gesture can be continued and possibly preview the respective outcomes (similar to [13]).

**Menus:** For editing complex diagram languages with many different types of nodes and edges, menus are certainly necessary. However, as stated above, invoking menus and selecting items usually disrupts the current workflow. Traditional context menus cope with this issue. They are appearing in situ and their items are adapted to the current context. To support a smooth workflow, and with our design goal G3 in mind (see 4.1), we additionally propose that context menus should explicitly support interaction by means of pen & multi-touch input. Therefore, they have to be carefully designed and adapted to these modalities.

## 8 Conclusion & Future Work

We contributed a user-defined and expert-refined gesture set for diagram editing on interactive surfaces which serves as a basis for the implementation of a diagram editor. It is the first one in this domain and consists of uni- and bimanual gestures combining touch and pen input. A collection of user-elicited gestures provided the starting point for the research presented here. Involving users right from the beginning is beneficial, not only to elicit preferred gestures for given tasks, but also to observe prevalent mental models and behaviors. Although users are not designers, these observations can give valuable hints how design challenges can be solved.

In order to realize the set, we identified basic design goals and analyzed existing conflicts within the collection of gestures. We discussed options how these ambiguities can be solved, whereby we seized on users' suggestions. Thereafter, we described the implemented gesture set in detail and briefly presented the architecture of the prototypical editor application.

For future work we will evaluate the currently implemented gesture set and extend it for example by expert gestures. Furthermore, features such as contextual assistance and menu techniques for more complex diagram types shall be added (see Section 7). Beyond that, additional scenarios need to be investigated and carefully studied.

**Acknowledgements.** We thank Sebastian Kleinau for his great support. This work was funded by the "Stifterverband für die Deutsche Wissenschaft" from funds of the Claussen-Simon-Endowment and the German Ministry of Education and Science (BMBF) within the ViERforES project (no. 01 IM 08003).

## References

1. Anoto Group AB, <http://www.anoto.com/>
2. Brandl, P., Forlines, C., Wigdor, D., Haller, M., and Shen, C. Combining and measuring the benefits of bimanual pen and direct-touch interaction on horizontal interfaces. In: Proc. of AVI '08, pp. 154-161, ACM, New York (2008).
3. Brandl, P.; Haller, M.; Hurnaus, M.; Lugmayr, V.; Oberngruber, J.; Oster, C.; Schafleitner, C. & Billinghamurst, M.: An Adaptable Rear-Projection Screen Using Digital Pens And Hand Gestures, In: Proc. of ICAT 2007, pp. 49-54, IEEE Computer Society (2007)
4. Cao, X., Wilson, A.D., Balakrishnan, R., Hinckley, K., Hudson S. E.: ShapeTouch: Leveraging Contact Shape on Interactive Surfaces. In: Proc. of TABLETOP '08, pp. 139-146 (2008).
5. Chen, Q., Grundy, J., and Hosking, J.: An e-whiteboard application to support early design-stage sketching of UML diagrams. In: Proc. of HCC 2003, pp. 219-226, IEEE (2003)
6. Cherubini, M., Venolia, G., DeLine, R., and Ko, A. J.: Let's go to the whiteboard: how and why software developers use drawings. In: Proc. of CHI 2007, pp. 557-566, ACM (2007)
7. Community Core Vision, NUI Group, <http://ccv.nuigroup.com/>
8. Damm, C. H., Hansen, K. M., and Thomsen, M. 2000. Tool support for cooperative object-oriented design: gesture based modeling on an electronic whiteboard. In Proc. of CHI '00, pp. 518-525, ACM, New York (2000).

9. Dekel, U.: Supporting distributed software design meetings: what can we learn from co-located meetings? In: Proc. of HSSE 2005, pp. 1-7, ACM, New York (2005)
10. Dietz, P. and Leigh, D. DiamondTouch: a multi-user touch technology. In: Proc. of UIST 2001, pp. 219-226, ACM, New York (2001)
11. Dwyer, T., Lee, B., Fisher, D., Quinn, K. I., Isenberg, P., Robertson, G., and North, C.: A Comparison of User-Generated and Automatic Graph Layouts. In: IEEE Trans. on Visualization and Computer Graphics 15, 6, pp.961-968 (2009)
12. Epps, J.; Lichman, S. & Wu, M.: A study of hand shape use in tabletop gesture interaction, In: CHI 2006 Ext. Abstracts, pp. 748-753, ACM, New York (2006)
13. Freeman, D., Benko, H., Morris, M.R., and Wigdor, D.: ShadowGuides: Visualizations for In-Situ Learning of Multi-Touch and Whole-Hand Gestures. In: Proc. ACM ITS '09, pp. 183-190, ACM, New York (2009)
14. Frisch, M., Heydekorn, J., Dachsel, R.: Investigating Multi-Touch and Pen Gestures for Diagram Editing on Interactive Surfaces. In: Proc. of ACM IST '09, pp. 167-174, ACM, New York (2009)
15. Grundy, J. and Hosking, J.: Supporting Generic Sketching-Based Input of Diagrams in a Domain-Specific Visual Language Meta-Tool. In Pro. of ICSE '07, pp. 282-291 (2007)
16. Guiard, Y., Asymmetric Division of Labor in Human Skilled Bimanual Action: The Kinematic Chain as a Model, The Journal of Motor Behavior, 19,4, pp. 486-517 (1987).
17. Hammond, T. and Davis, R.: Tahuti: a geometrical sketch recognition system for UML class diagrams. In: ACM SIGGRAPH 2006 Courses, ACM, New York (2006)
18. Han, J. Y.: Low-cost multi-touch sensing through frustrated total internal reflection. In Proc. of UIST '05, pp. 115-118, ACM, New York (2005)
19. Leitner, J.; Powell, J.; Brandl, P.; Seifried, T.; Haller, M.; Dorray, B. & To, P.: Flux: a tilting multi-touch and pen based surface, In: CHI '09 Ext. Abstracts., pp. 3211-3216, ACM New York (2009)
20. Malik, S., Ranjan, A., and Balakrishnan, R.: Interacting with large displays from a distance with vision-tracked multi-finger gestural input. In: Proc. UIST '05, pp. 43-52, ACM (2005)
21. Micire, M., Desai, M., Courtemanche, A., Tsui, K., Yanco, H.: Analysis of Natural Gestures for Controlling Robot Teams on Multi-touch Tabletop Surfaces. In: Proc. ACM IST '09, pp. ACM, New York (2009)
22. Microsoft Surface, <http://www.microsoft.com/surface/>
23. Moran, T. P., Chiu, P., and van Melle, W.: Pen-based interaction techniques for organizing material on an electronic whiteboard. In: Proc. UIST '97, pp.45-54, ACM, New York (1997)
24. Nielsen, M., Störing, M., Moeslund, T.B. and Granum, E.: A procedure for developing intuitive and ergonomic gesture interfaces for HCI. In: Int'l Gesture Workshop, LNCS vol. 2915, pp. 409-420, Springer Heidelberg (2004)
25. Rekimoto, J. 2002.: SmartSkin: an infrastructure for freehand manipulation on interactive surfaces. In Proc. CHI '02, pp. 113-120, ACM, New York (2002)
26. Ringel, M.; Ryall, K.; Shen, C.; Forlines, C. & Vernier, F.: Release, relocate, reorient, resize: fluid techniques for document sharing on multi-user interactive tables, In: Ext. Abstracts CHI 2004, pp.1441-1444, ACM, New York (2004)
27. TUIO Protocol, <http://www.tuio.org/>
28. Wobbrock, J. O., Morris, M. R., and Wilson, A. D.: User-defined gestures for surface computing. In: Proc. of CHI '09, pp. 1083-1092, ACM, New York (2009)
29. Wu, M. & Balakrishnan, R.: Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In Proc. of UIST'03, pp. 193-202, ACM, New York (2003).
30. Wu, M.; Shen, C.; Ryall, K.; Forlines, C. & Balakrishnan, R.: Gesture Registration, Relaxation, and Reuse for Multi-Point Direct-Touch Surfaces, In: Proc. TABLETOP '06, pp. 185-192, IEEE (2006)
31. Yee, K.: Two-handed interaction on a tablet display. In: Ext. Abs. CHI '04, pp.1493-149, ACM, New York (2004)