Washington University in St. Louis
Washington University Open Scholarship

All Computer Science and Engineering Research | Computer Science and Engineering

Report Number: WUCS-93-18

1993-01-01

A Comparison Study of The Pen and The Mouse in Editing Graphic Diagrams

Ajay Apte and Takayuki Dan Kimura

We report the results of an experiment comparing the merits of the pen and the mouse as drawing devices. For this study a pen-based graphic diagram editor equipped with a shape recognition algorithm was developed on GO's PenPoint operating system. A commercially available drawing program on NeXT was used for mouse-based editing. Twelve CS students were chosen as subjects and asked to draw four different diagrams of similar complexity: two with a pen and the other two with a mouse. The diagrams are chosen from the categories of dataflow visual language, Petri nets, flowcharts, and state diagrams. The results... **Read complete abstract on page 2.**

Follow this and additional works at: https://openscholarship.wustl.edu/cse_research

 Part of the Computer Engineering Commons, and the Computer Sciences Commons

Recommended Citation

Apte, Ajay and Kimura, Takayuki Dan, "A Comparison Study of The Pen and The Mouse in Editing Graphic Diagrams" Report Number: WUCS-93-18 (1993). *All Computer Science and Engineering Research.*
https://openscholarship.wustl.edu/cse_research/305

Department of Computer Science & Engineering - Washington University in St. Louis
Campus Box 1045 - St. Louis, MO - 63130 - ph: (314) 935-6160.

This technical report is available at Washington University Open Scholarship: https://openscholarship.wustl.edu/cse_research/305

A Comparison Study of The Pen and The Mouse in Editing Graphic Diagrams

Ajay Apte and Takayuki Dan Kimura

Complete Abstract:

We report the results of an experiment comparing the merits of the pen and the mouse as drawing devices. For this study a pen-based graphic diagram editor equipped with a shape recognition algorithm was developed on GO's PenPoint operating system. A commercially available drawing program on NeXT was used for mouse-based editing. Twelve CS students were chosen as subjects and asked to draw four different diagrams of similar complexity: two with a pen and the other two with a mouse. The diagrams are chosen from the categories of dataflow visual language, Petri nets, flowcharts, and state diagrams. The results indicate that drawing by pen is twice as fast as drawing by mouse.

A Comparison Study of The Pen and The Mouse in Editing Graphic Diagrams

Ajay Apte and Takayuki Dan Kimura

WUCS-93-18

March 1993

Laboratory for Pen-Based Silicon Paper Technology
Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis MO 63130-4899
phone (314) 935-6122, fax (314) 935-7302
tdk@wucs1.wustl.edu

A Comparison Study of The Pen and The Mouse in Editing Graphic Diagrams[1]

Ajay Apte and Takayuki Dan Kimura

Laboratory for Pen-Based Silicon Paper Technology (PenLab)
Department of Computer Science
Washington University
St. Louis, MO 63105, USA
(314) 935-6122, tdk@wucs1.wustl.edu

Abstract

We report the results of an experiment comparing the merits of the pen and the mouse as drawing devices. For this study a pen-based graphic diagram editor equipped with a shape recognition algorithm was developed on GO's PenPoint operating system. A commercially available drawing program on NeXT was used for mouse-based editing. Twelve CS students were chosen as subjects and asked to draw four different diagrams of similar complexity: two with a pen and the other two with a mouse. The diagrams are chosen from the categories of dataflow visual language, Petri nets, flowcharts, and state diagrams. The results indicate that drawing by pen is twice as fast as drawing by mouse.

1. Introduction

Various visual languages require drawing and editing graphic diagrams. Convenient and easy drawing is an important factor for the acceptability of visual languages by end users. The existing tools for diagram drawing, such as MacDraw, use the mouse/menu paradigm of user interface. The pen/gesture paradigm offers a possibly improved alternative.

The pen user interface is heralded as more natural and powerful than the mouse, due to pointing, handwriting, gesturing, and drawing [3, 5, 6]. It is widely believed that the pen is mightier than the mouse. There are comparison studies of different input devices such as pen, mouse and keyboard in different application areas, e.g., [10] for spreadsheet applications and [8] for text entry. However, scientific evidence is not yet well-established in the area of visual languages. It is the goal of this study to fill this gap.

This paper reports the results of an experiment designed to compare the ease of constructing graphic diagrams, with either a pen or a mouse. To disconnect the issue of keyboarding in user interface, we have eliminated all textual components of graphic diagrams. The experiment consists of:

Selection of subjects.

Selection of diagrams to be drawn by each subject.

Selection of editing tools (pen or mouse).

Assignment of tasks; who draws which diagram in what order using which tool.

[1] This research is partially supported by the Kumon Machine Project and by Laboratory for Pen-Based Silicon Paper Technology.

Execution of each task and data collection.

Analysis of collected data.

Twelve CS students with little pen experience were chosen as subjects. Four diagrams of similar complexity were selected, one each from the categories of dataflow visual programs, Petri nets, flowcharts, and state diagrams. We have developed a pen-based diagram editor for GO's PenPoint [1] running on NCR3125 pen computer [9]. The editor can recognize and respond to shapes and gesture commands. Appsoft's Draw version 1.02[4] on NeXT was used as the mouse-based diagram editor. A random task assignment table was constructed. Procedures and instruction sheets for the subjects were prepared. The experiment was executed and monitored by the same person throughout to collect relevant data.

We faced two difficulties at the onset of this work: (1) It is difficult to measure the convenience and ease of drawing, and (2) few pen-based drawing programs are available. As an objective measurement, we have chosen to compare the length of time necessary to construct the same diagram with reasonable precision, using both pen and mouse. As a subjective measurement, we sought comments from each participant comparing the pen and the mouse with respect to the ease of use and general preferences.

To address the second difficulty, we evaluated a commercial product, NoteTaker 1.0.3 of InkWare [6], as a possible candidate for pen-based drawing tool. We found that it was not suitable for this experiment . Its shape recognition is too limited. In NoteTaker every graphic object, e.g., a rectangle or a circle, must be entered by a single stroke for correct recognition. This would impose undue restrictions on user interaction with the editor. Instead, we decided to construct our own editor, called GDE (Graphic Diagram Editor), that can recognize multi-stroke entries of a graphic shape. There is no restriction in GDE on the number of strokes that can be used to construct a single shape, as long as the stroke sequence is entered without long pauses between the strokes.

Another minor problem with NoteTaker is that it does not provide any gesture-based commands in ShapeExpert, its drawing component. The ShapeExpert is controlled by tapping menus and palettes. It does not take advantage of pen's power to select an operator and an operand by a single gesture, thus reducing the traffic between the editing space and menu items. GDE offers mostly gesture-based commands for editing diagrams.

In the next section we describe GDE. In Section 3 we outline the experimental design. In Section 4 we tabulate the results . In Section 5 we evaluate the validity of our experiments and results. Finally, we conclude with future directions.

2. Graphic Diagram Editor

GDE is a pen-based geometric editor with a shape recognition capability. It runs on GO's PenPoint operating system. The editor supports a two-dimensional layout of geometric objects on a transparent background. The geometric objects can be rectangles, circles, ellipses, lines, diamonds and polylines. No text can be entered by GDE. Objects can be created, selected, deleted, moved, resized, copied and pasted. Options are provided to set the line thickness to

either normal or bold. The option 'Line Style' is provided which allows lines to be drawn with or without arrowheads. GDE provides gesture support in editing geometric objects. The gestures that are currently provided include single-tap, flick and circle. These gestures are used to select, move, resize, copy and paste the objects. In short, all the editing operations except delete can be performed without using menu options. An important feature of GDE is its automatic shape recognition. (See Section 2.1.) The recognizer can be turned on or off selectively. GDE also supports loading and saving of diagrams. GDE is coded in PenPoint C, and the source code is approximately 5000 lines and the executable code is 262KB.

2.1 The Shape Recognizer

The shape recognizer used in GDE is algorithmic rather than adaptive (like neural nets). Some of the benefits of using an algorithm over a neural net recognizer include reduction in the size of the program and quicker response times. An algorithm does not need to be 'taught' like a neural net, thus simplifying user interface design of the editor.

Our shape recognizer does not require users to draw a shape in a certain way. It allows the users to lift the pen and draw the shape with as many strokes as they like and in any order they wish. The algorithm does not use dynamic stroke information to evaluate input. Only static data points (i.e. XY co-ordinates) are used and they may be input in any order. The recognition algorithm is invoked at the end of last stroke of the shape. Successive strokes of the same shape and the terminating stroke are distinguished by time-out. The time-out is proportional to the number of data points collected in the previous stroke so that larger objects have longer time-out.

The recognition algorithm uses several filters. Each filter guesses at the shape gives an error value. If the error is too large, then another filter is used until the error is within the desired range.

The first filter used in this algorithm is based on the observation that for regular polygons the ratio of the square of perimeter(P) to area(A) is constant. For a square, $P^2/A = 16$, independently of the size of the sides. This idea is extended to the shapes with unequal widths and hights such as non-square rectangles. The second filter is the ratio of the input area to the bounding rectangle. A rectangle should have a ratio close to 1, a diamond close to 0.5, and an ellipse somewhere in between.

The final action is rejection of the data or a default shape. All the shapes except polylines are drawn in this way. A polyline is drawn by tapping on the screen at the positions corresponding to the vertices of the polyline. The end of polyline is represented by drawing a small flick gesture starting from the last vertex.

2.2 Shape Editing and Gesture Support

Once the shape is recognized, it is drawn either using normal or bold thickness depending upon the current selection of thickness from the menu. A line or a polyline can be drawn with or without arrowheads. This facility is provided as a 'Line Style' item in the menu bar. In the case of a polyline, the arrowhead is drawn only at the last vertex. A shape drawn on the screen can be

selected by making a single tap gesture anywhere inside that shape for non-line shapes, and, on or near the line for line shapes. A selected shape is represented by control buttons at its corners.

After the shape is selected it can be resized by touching the pen inside any of the control buttons and dragging it in the desired direction. By touching the pen anywhere inside the shape but not in the control buttons, the shape can be moved. The selected shape can be deleted or copied into the buffer by using menu options. The shape that is copied in the buffer can be pasted by tapping the pen on the screen such that the point of contact becomes the geometric centre of that shape. However, the shape stored in the buffer can be pasted only once. Another way to copy the shape is, using a circular gesture. A circular gesture is a drawing of an either circular or elliptical shape at any location on the tablet. Thus by drawing the circle followed by a single tap, one can copy and paste the selected shape. In recognition mode, the circle is recognized as a circular shape, whereas, in selection mode, the circle is recognized as a gesture to copy the selection into the buffer. When the circle is used as a gesture, the size of the gesture does not matter. Similarly, when a 'flick' gesture is used to indicate the end of polyline, the size or orientation of the flick does not matter.

3. The Experimental Design

The experiment consist of asking each subject to draw four diagrams in sequence using pre-assigned drawing tools while the experimenter records start and finish times for each diagram. After the drawing session, the experimenter interviewed the subjects for comments and preferences.

3.1 Subject Selection

We chose twelve graduate and undergraduate computer science majors as our subjects. All the subjects had used a mouse extensively on mouse-based graphic editors, but none had used a pen for this purpose. Only one subject was familiar with a pen-based system. The subjects were not aware of the aim of the experiment. This care was taken so that the result of the experiment would not be influenced by the prejudices of the subject.

3.2 Object Selection

We asked our subjects to draw four diagrams each. We opted for diagrams that are commonly drawn by CS majors, namely, Petri net, flowchart, data flow and state diagram (see Figure 1). All the diagrams were of similar complexity, consisting of approximately ten lines and ten closed shapes each. No text entry was required to construct any of these diagrams. As a precondition, it should not take more than ten minutes to draw any of the four diagrams.

3.3 Tools

As a mouse-based editor, we used AppsoftDraw 1.02 on a Next Computer without a keyboard. This is a typical mouse-based graphic editor that uses a tool palette to draw various shapes. It facilitates editing abilities like Selection, Deletion, Copy/Paste. Of all the shapes in the tool palette, we allowed the subjects to use only rectangles, circles, lines and polylines. The keyboard mappings

corresponding to the editing options were disabled.

As a pen-based editor we used GDE described in Section 2. It was installed on an NCR3125 pen computer which runs on a 80286 machine at 20MHz. The features of GDE are also explained in section 2.

3.4 Experimental Setup

To avoid the undesired influence of learning, (i.e. one can draw the same diagram faster a second time), we asked every subject to draw all four diagrams with a different combination of editors. The task assignment for each subject was done based on the design presented in Table 1.

Table 1 - Various combinations of diagrams given to different subjects

	Petri Net	Flowchart	State Diagram	Data Flow
Combination1	P	P	M	M
Combination2	P	M	M	P
Combination3	M	M	P	P
Combination4	M	P	P	M
Combination5	M	P	M	P
Combination6	P	M	P	M

P : Pen-Based Editor M : Mouse-Based Editor

The above table is constructed so that each diagram is drawn an equal number of times using the pen and the mouse. Also all diagrams are drawn the same number of times. The sequence of diagrams is administered in such a way that no two diagrams are drawn on the same editor consecutively. For example, a subject who is assigned to perform the Combination1 may be given four diagrams in the following order; Petri Net, State Diagram, Flowchart, then Data Flow. We did not want our subjects to get used to the editor and bias the result.

3.5 Experiment Execution

Each subject was given a randomly selected combination of four diagrams and two editors based on Table 1. Each subject was given a small demonstration of the two editors. Then he/she was allowed to have a brief hands-on experience with the systems. The subjects were told which shapes can be drawn with the editors, and what editing facilities are provided. They were instructed to use the same editing facilities on both the editors despite the fact that the mouse-based editor had more editing features than the pen-based one.

The subjects were told that the precision of the diagrams was not of utmost importance, however, at the same time it could not be neglected as well. The subjects were not aware of the fact that they were being timed. This care was taken in order for the subjects not to feel pressured while drawing. They were also allowed to ask the experimenter when in doubt. Each subject spent approximately thirty minutes drawing the four diagrams. A brief discussion was held with

each subject after the experiment in which they were asked to give the merits and demerits of the two input devices.

4. Results

The results of our experiment are shown in Table 2, and summarized in Table 3 below.

Table 2- Time required for each subject to draw each diagram

	Petri Net	Flowchart	State Diagram	Data Flow
Subject1	4.2*	2.1*	6.1	6.45
Subject2	4.0*	7.2	8.25	4.15*
Subject3	10.45	9.2	6.50*	7.10*
Subject4	12.4	4.0*	5.20*	9.15
Subject5	9.3	3.20*	9.05	4.0*
Subject6	4.25*	11.25	4.05*	9.1
Subject7	12.4	4.0*	5.20*	10.15
Subject8	5.15*	7.3	8.45	4.55*
Subject9	3.45*	3.15*	7.3	8.15
Subject10	4.15*	8.2	4.50*	9.3
Subject11	8.05	7.4	4.15*	3.55*
Subject12	9.15	2.35*	7.55	3.55*

Note : Pen-based times are marked by *. All times are shown in minutes.

Table 3 - Average time required to draw each diagram using the pen and the mouse

	Petri Net	Flowchart	State Diagram	Data Flow
Time(Pen)	4.2	3.13	5.07	4.42
Time(Mouse)	10.29	8.36	7.58	8.52
Ratio	2.42	2.67	1.55	1.88

Notes: Time(Pen) : Average time taken by six subjects to draw a diagram using the pen.
Time(Mouse) :Average time taken by six subjects to draw a diagram using the mouse.
Ratio : Time(Mouse) / Time(Pen).

In a brief discussion held with each subject after the drawing session, all the subjects except one, felt that drawing graphic diagrams with a pen is not only easier but more natural than drawing with a mouse. Most of the subjects said that they did not have to think about the layout of the shapes in a diagram. They just drew the diagrams as they would draw them on paper with a pencil. Six of the twelve subjects complained about the small screen size of the pen-based

system. Only one subject felt that the mouse was easier to use for precise drawing. He was accustomed to the mouse and could pinpoint any shape on the screen more easily with a mouse than a pen. Samples of diagrams constructed by the subjects using the pens and the mice are given in Figure 2.

5. Evaluation of results and conclusion

The obvious conclusion drawn from the above results is that editing graphic diagrams using the pen is approximately twice as fast as editing using the mouse.

In the examples like the state diagram or Petri net., the subjects particularly preferred the pen due to the gestures provided for Copy/Paste feature. The subjects complained about the fact that, in the mouse-based system, they had to constantly switch back and forth between the tool palette and the drawing context. They also found the pen-based editor easier for resizing and moving the shapes. Half the subjects were annoyed about the small screen size of the pen-based system despite the fact that a scroller was provided in the GDE.

In the design of the experiment, we assured that there would be variation in the combination of diagrams that are given to all the subjects. This was done to eliminate any influence on the results due to the penchant of the subjects towards any particular input device. Also the complexity of the diagrams was such that the subjects did not take more than twelve minutes to draw the diagrams, with a couple of exceptions. We also gave a little break between two diagrams, and when the subject felt fatigued in some cases.

Despite all the above precautions, there are some shortcomings in our experiment. We feel that twelve subjects are not sufficient to quantify the speed of editing. We did not allow text entry in our experiment since we did not want to complicate the issue by involving a keyboard. However, we are aware of the fact that text is an integral part of a graphic editor.

Our experiment focused on speed. We made no efforts to quantify precision or ease of drawing. The precision was visually checked by the experimenter. The ease of using both input devices was determined only qualitatively through a brief discussion after each drawing session. This is not a very accurate measurement of the two above-mentioned factors. We believe that more thorough experimentation needs to be performed in this area.

There are other ways of comparing the pen and the mouse. For example, some subjects stated that the pen is easier to learn than the mouse. We need to design another experiment to validate that claim. Also traditional psychological experiments to measure the response time and the accuracy of selecting randomly displayed pixels by the pen or the mouse would be valuable additions to our efforts to compare the pen and the mouse.

6. Acknowledgements

We wish to thank Van Vo for his ideas about the algorithm for shape recognition and Dale Frye for help in the software development We express our gratitude to Tim Brown for his critical comments about the experiment design and Tom Fuller for editing the manuscripts. Finally, we would also like to thank all the subjects for participating in the experiment.

7. References

1. Carr, R. and Shafer, D. *The Power of PenPoint*. Addison-Wesley, 1991.

2. *Draw User's Guide.* AppSoft, Inc., 1992.

3. Dunne, S. "Towards Interactive Pen Input of Visual Languages," *Proceedings of 1992 IEEE Workshop on Visual Languages*, Seattle, September 1992, pp 243-245.

4. *InkWare NoteTaker User Guide*, InkDevelopment Corporation, 1992.

5. Kimura, T.D. "Silicon Paper and A Visual Interface for Neural Networks," *Proceedings of 1990 IEEE Workshop on Visual Languages*, Chicago, IL, October 1990, pp. 241-246.

6. Kimura, T.D. "Hyperflow: A Visual Programming Language for Pen Computers," *Proceedings of 1992 IEEE Workshop on Visual Languages*, Seattle, September 1992, pp 125-132.

7. Kimura, T.D. "Potentials and Limitations of Pen-Based Computers," Panel session chiarman's position paper," *Proceedings of 21st ACM Annual Computer Science Conference(CSC'93)*, Indianapolis, February 1993, pp. 536-537.

8. Mahach, K.R. "A Comparison of Computer Input Devices: Linus Pen, Mouse, Cursor Keys and Keyboard," *Proceedings of the 33rd Annual Meetings of the Human Factor Society*, 1989, pp. 330-333.

9. *System 3125 User's manual.* NCR Corporation, 1991.

10. Wolf, C. G. "A Comparative Study of Gestural, Keyboard, and Mouse Interfaces," *Behavior & Information Technology*, vol. 11:1, 1992, pp. 13 - 23.

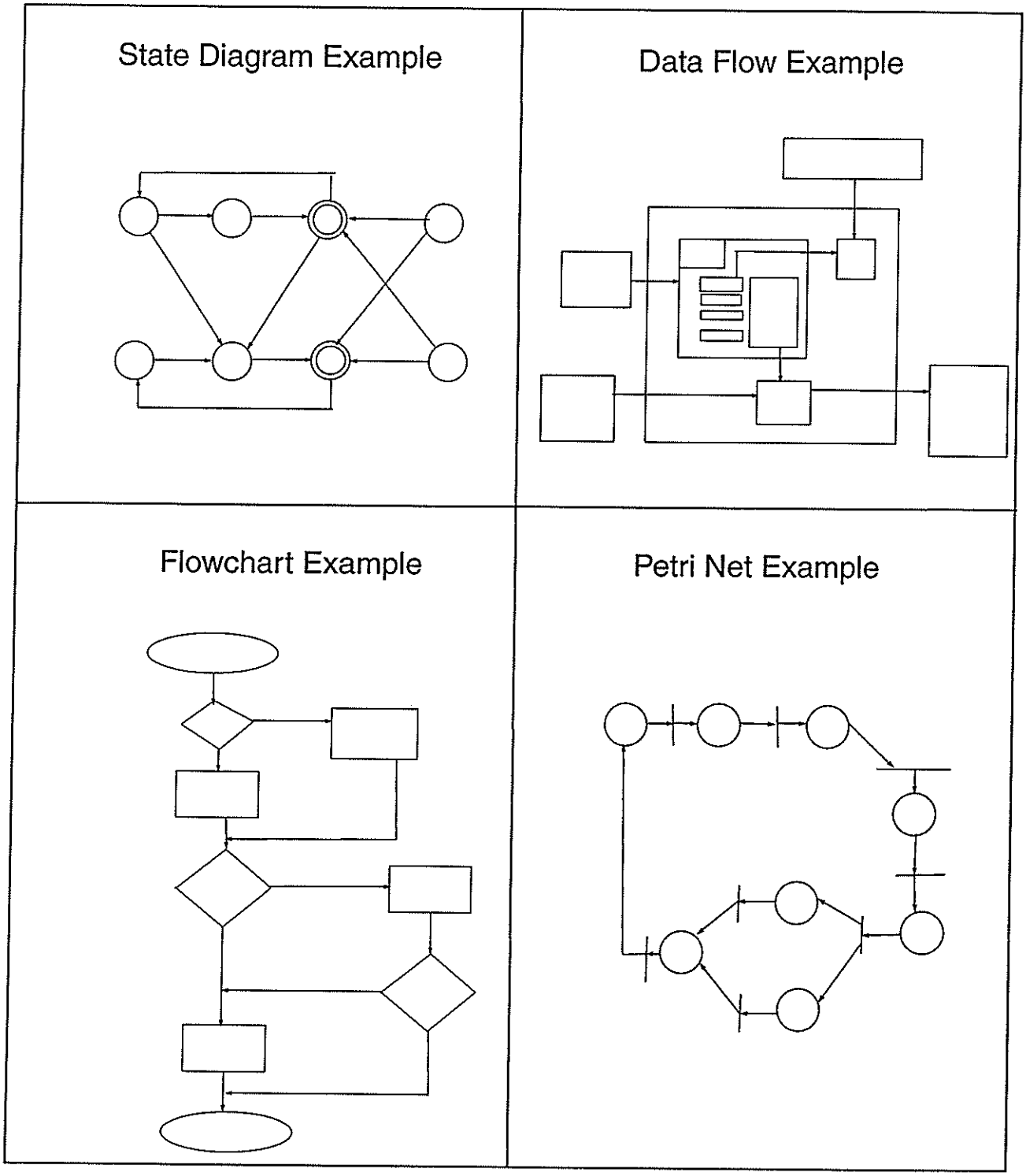

Figure 1: Object diagrams for the experiment

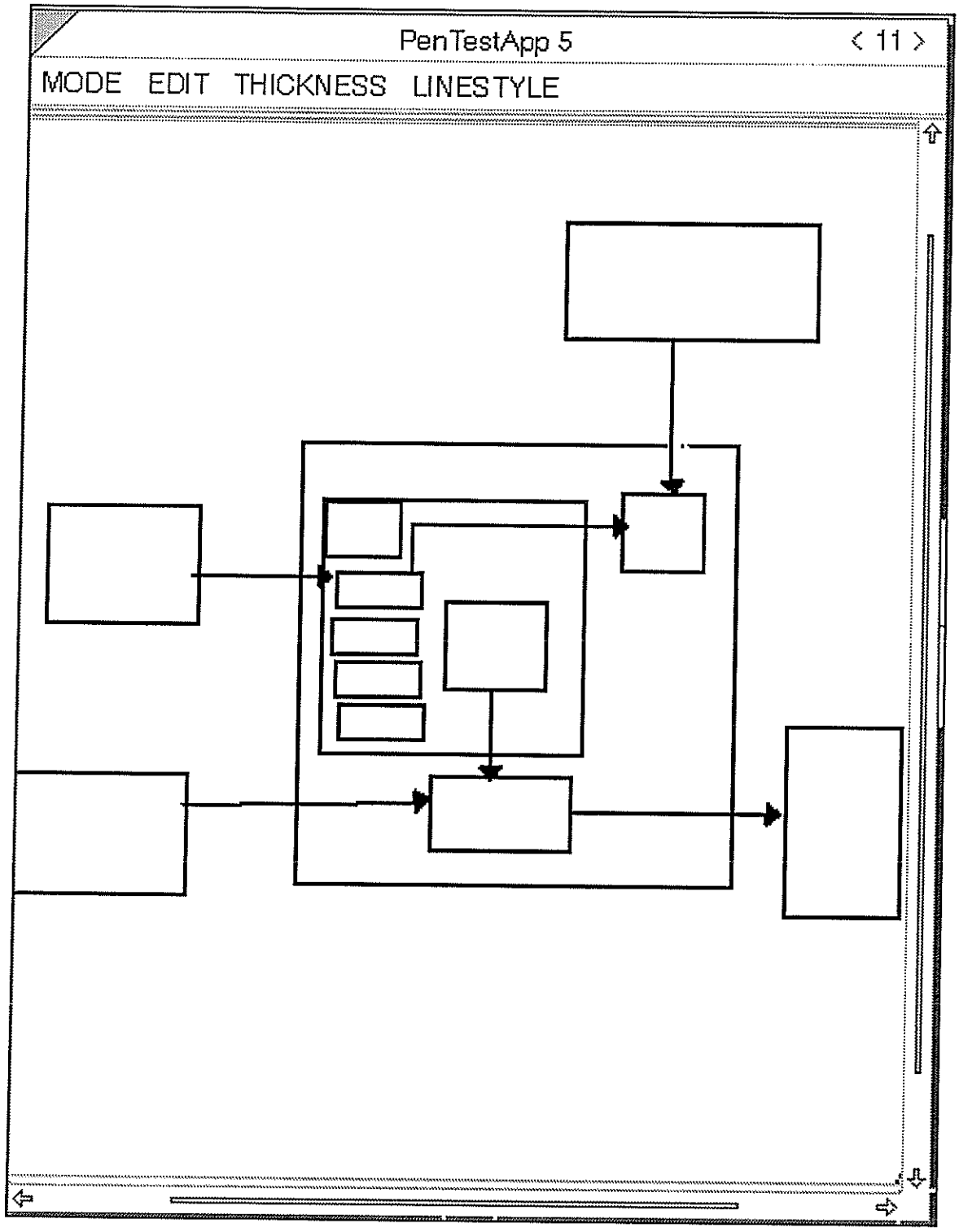

Figure 2a : Sample data flow diagram drawn by Subject8 using GDE.

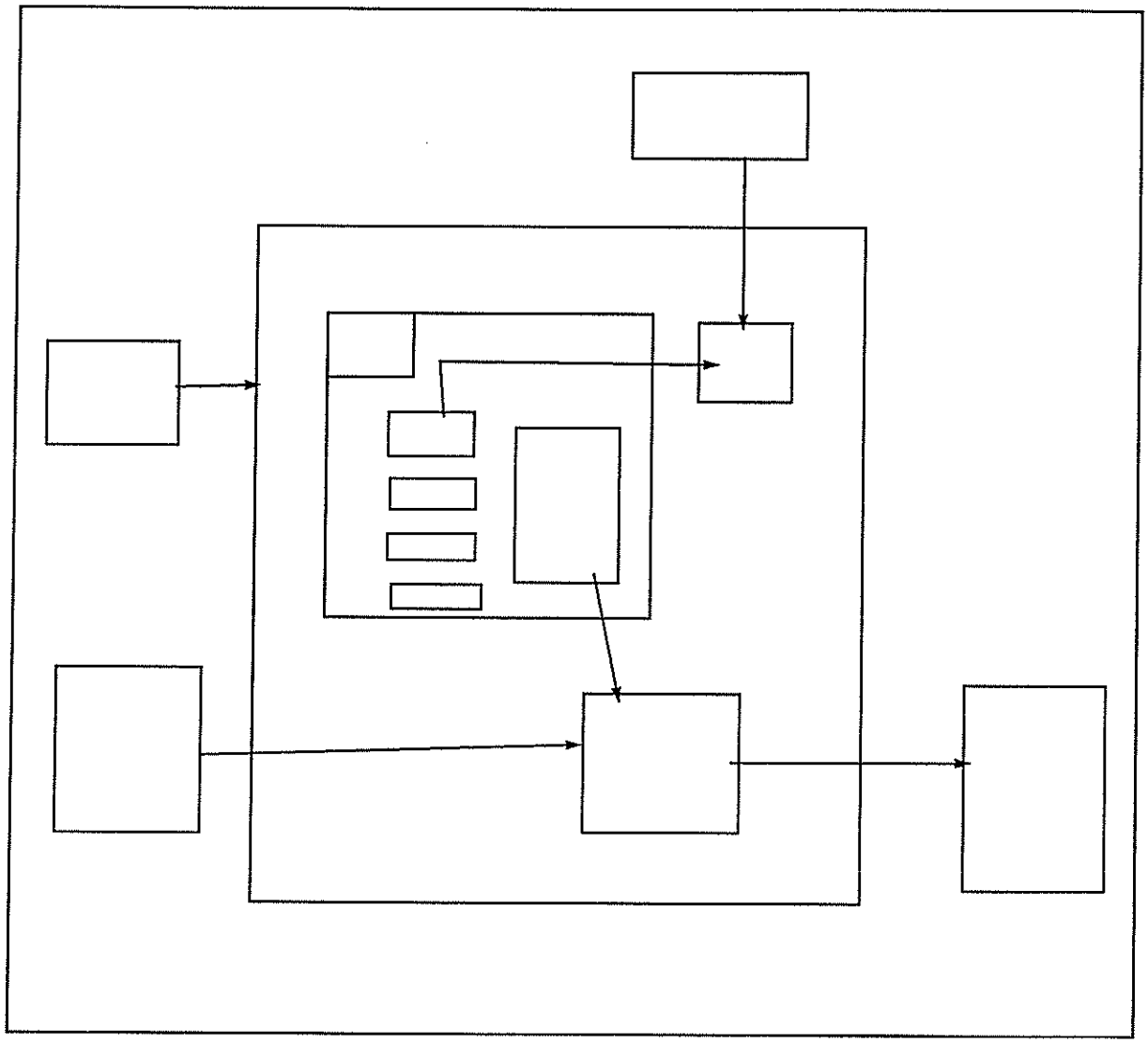

Figure 2b: Sample data flow diagram drawn by Subject4 using AppsoftDraw.