RESEARCH REPORT ISIS-RR-96-7E

Techniques for DUI Platforms: Developing Graph Drawing Applications on D-ABDUCTOR

Kazuo Misue Kiyoshi Nitta Kozo Sugiyama
Takeshi Koshiba Robert Inder*

June, 1996

Institute for Social Information Science (*ISIS*)
at Numazu

FUJITSU LABORATORIES LTD.

140 Miyamoto, Numazu-shi, Shizuoka 410-03, Japan
Telephone: +81-559-24-7210 Fax: +81-559-24-6180

Techniques for DUI Platforms: Developing Graph Drawing Applications on D-ABDUCTOR

Kazuo Misue Kiyoshi Nitta Kozo Sugiyama
Takeshi Koshiba Robert Inder*

Institute for Social Information Science (*ISIS*)
at Numazu

FUJITSU LABORATORIES LTD.

140 Miyamoto, Numazu-shi, Shizuoka 410-03, Japan

Email: misue@iias.flab.fujitsu.co.jp

Abstract

An important application of automatic graph drawing is providing diagrammatic user interface (DUI). Although a DUI is an important part for the user of a graph drawing application, implementation of application systems with such the DUIs tends to require much cost. Therefore, a DUI platform is required. In this paper, how a DUI platform can be customized or extended to create a variety of DUIs are illustrated. Three generic levels of DUIs are distinguished; viewing level, selecting level, and manipulating level. In each DUI level, examples of graph drawing applications are shown. All examples in this paper have been developed by using D-ABDUCTOR as a DUI platform. This paper also presents how to use D-ABDUCTOR as a DUI platform by illustrating mechanisms of these examples.

Key words: diagrammatic user interface (DUI), DUI platform, D-ABDUCTOR, automatic graph drawing

*University of Edinburgh, Human Communication Research Centre

1 Introduction

It is widely known that automatic graph drawing is of use. One of its signs is that very many graph drawing algorithms have been developed [1]. An important application of automatic graph drawing is providing *diagrammatic user interface* (DUI) [2], which allows human to interact with computers through diagrams, that is, visual representations of structural information.

A DUI is an important part for the user of a graph drawing application, since the DUI has an influence on usability of the application. To offer more usability to the users, DUIs have to provide greater services by exploiting automatic graph drawing effectively. However, implementation of application systems with such the DUIs tends to require more complex mechanisms and more cost. So, a DUI platform, that is, a flexible base on which we can develop graph drawing applications with various DUIs quickly and efficiently is required.

In this paper we show how a DUI platform can be customized or extended to create a variety of DUIs; we illustrate with systems for "Decision Tree Generator," "Network Monitor," "Directory Browser," "Graphical Hypertext," "Graphical Outline Processor," and "Structure Analyzer."

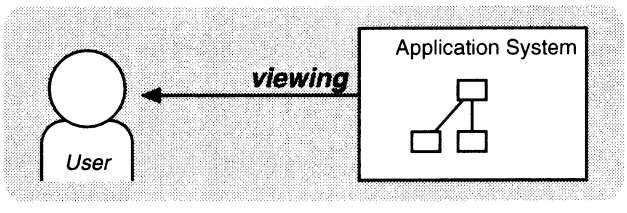
In the following part of this paper, first we explain three generic levels of DUIs, and then list requirements for DUI platforms applicable to all DUI levels. Next, in each DUI level, we show examples of graph drawing applications developed by using D-ABDUCTOR [3,4] as a DUI platform. Finally, based on our experience of developing applications, we discuss what is desired for DUI platforms.

2 DUI Levels

Architecture of application systems depends on styles of DUIs, in other words, services offered by DUIs. The following three generic levels of DUIs can be distinguished.

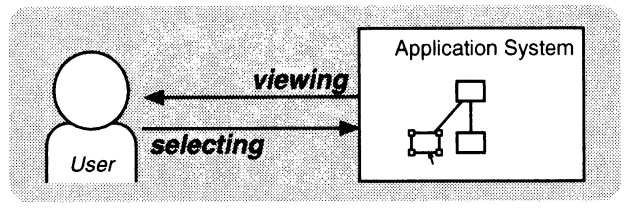
I. Viewing Level

In this level, graphs are drawn only to be viewed. The users are not allowed to input anything through graphs.



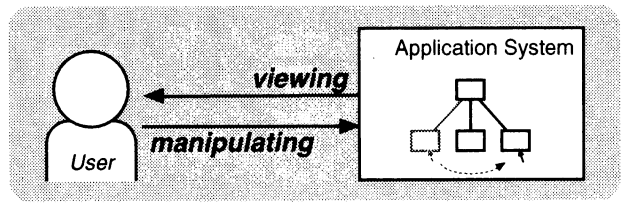
II. Selecting Level

In this level, graphs are drawn to be viewed and their elements work as selectable buttons. The users are allowed to select some elements of graphs.



III. Manipulating Level

In this level, graphs are drawn to be viewed and also works as an object of direct manipulation. Manipulation of the graph changes application data.



These three levels of DUI can be considered to represent a hierarchy in terms of the degree and sophistication of interaction between computers and the users. Although a higher level of DUIs can more sufficiently exploit automatic graph drawing, implementation of a higher level of DUIs tends to require more complex mechanisms; more information must be exchanged between a DUI and an application module. More complex mechanism certainly requires more cost in development.

3 DUI Platforms

A DUI platform is desired to develop graph drawing applications with less cost. DUI platforms have two aspects: (1) a system offering DUIs, it has to provide facilities to handle diagrams, and (2) a platform software, it must be extensible and be able to be combined with other application programs.

3.1 Diagram Handling

A DUI platform should provide diagram handling facilities such as the followings. DUIs in the all levels require the following facilities unless we describe notice especially.

General Diagram Handling: dealing with general diagrams to be applicable various applications.

Layout Creation: producing layout of diagrams and visualizing them by using automatic graph drawing.

Layout Adjustment: modifying visual attributes and / or geometric attributes of diagrams.

Mental-Map Preservation: preserving the user's mental map of diagrams [5], which is apt to be destroyed by layout creation or adjustment, to not decrease efficiency of user's tasks with application systems.

Direct Manipulation: allowing the users to manipulate diagrams directory by using pointing devices like a mouse. This facility is required partially in the selecting-level DUIs and totally in the manipulating-level DUIs.

3.2 Extensibility

A DUI platform needs to be combined with another application module in order to work as a DUI of an application system, and to be customizable to offer a DUI suitable for the application system.

Combination with Application Modules: A DUI platform can be combine with application modules, for example, it can communicate with application processes by using pipes, sockets, and so on, be linked with object programs of application modules, and be embedded in application programs.

Customizable User Interface: The users can define or redefine menus and semantics of mouse operations.

3.3 System Integration as a DUI Platform

Implementation of a DUI platform requires not only such facilities as mentioned above but also to integrate the facilities. It is very difficult how to integrate these many facilities for general purposes. However, it is an important problem in order to exploit automatic graph drawing in actual systems, and so a lot of research effort should be put into the problem in future. In this paper, we illustrate an instance of a DUI platform with D-ABDUCTOR.

D-ABDUCTOR[1]is a generic compound graph visualizer / manipulator which provides facilities for diagram visualization / manipulation in a direct manipulation environment. D-ABDUCTOR provides the following facilities to handle diagrams [3]:

Compound Graphs: As general diagrams, compound graphs that can represent both adjacency relationships and inclusion relationships can be handled.

Automatic Layout: As layout creation facilities, a Sugiyama-style drawing algorithm extended for compound graphs [6] is built-in, and also other external layout programs of, for example, a spring layout algorithm can be used. Some of these layout programs are applicable to the current diagram as layout adjustment facilities.

Diagram Dressing: As another layout adjustment facility, "diagram dressing" (a kind of fisheye view) which changes visual attributes of nodes according to their importance is provided.

[1]D-ABDUCTOR is available for non-profit use by anonymous ftp from: `SunSITE.sut.ac.jp` (133.31.30.7) `/pub/asia-info/japanese-src/packages/abd2.23.tar.gz`

Display Animation: Changes of diagrams are shown by animation which reduces the instantaneous visual change of diagrams so that the user's mental map of diagrams is preserved.

Direct Manipulation: Direct manipulation environment to edit compound graphs is provided. The users can select elements, move and resize nodes, make edges, and make and resolve groups directory by using mouse.

We can extend D-ABDUCTOR by combining with other application programs and programs for additional functions. To combine D-ABDUCTOR and other programs, the following facilities are available.

Language Simple: The language *Simple* is designed to describe compound graphs and manipulating commands for them. D-ABDUCTOR uses the language to save diagrams, to communicate with D-ABDUCTOR processes on other workstations, and to communicate with other programs. Since D-ABDUCTOR works as an interpreter of the language Simple, it can be controlled by other programs using Simple as a protocol language.

Communication with Other Programs: The current version of D-ABDUCTOR does not wait command from the standard input since it is an event driven system. However, it is convenient that D-ABDUCTOR can read commands from the standard input. A program `abd_tx` [3] reads character strings form the standard input and sends them to a D-ABDUCTOR process on the same display. D-ABDUCTOR provides a facility to record user's working history in the language Simple. The history, that is, a sequence of Simple statements is saved into a specified file or output to the standard output. Therefore, D-ABDUCTOR can communicate with other programs through UNIX pipes or files in the language Simple.

Extensible Menu: The users are allowed to add menu items to menus. When one of the menu items is chosen form a menu, the current diagram data is written into a temporary file, and the command corresponding with the menu item is invoked with the temporal file as an argument. Ordinarily, an invoked command reads the current diagram data, then produces Simple statements to change the diagrams, and finally send back the statements to D-ABDUCTOR by using `abd_tx`. The commands are also allowed to not send back changes to D-ABDUCTOR. So we can define, for example, a special hard-copy command, which does not change the current diagram.

4 Application Examples

We have developed several graph drawing applications by using D-ABDUCTOR as an instance of a DUI platform. In this section, for each DUI level, we show how D-ABDUCTOR can be customized or extended to develop applications with DUIs, and then give a few examples of actual graph drawing applications.

I. Applications with Viewing-Level DUI

Viewing-level DUIs only show diagrams and accept no input. An easy way to make D-ABDUCTOR work as a viewing-level DUI of an application program is to translate output of the application program into the language Simple and pass it to D-ABDUCTOR by using the program `abd_tx`.

Decision Tree Generator

Learning or generating decision trees by computers is an application of machine learning [7]. Computers sometimes generate very large trees, and it is difficult for us to grasp structural features only through text representation of the trees. *Decision Tree Generator* generates decision trees and visualize the trees by using D-ABDUCTOR (see Figure 1). Visualized trees help the user to grasp their structural features.

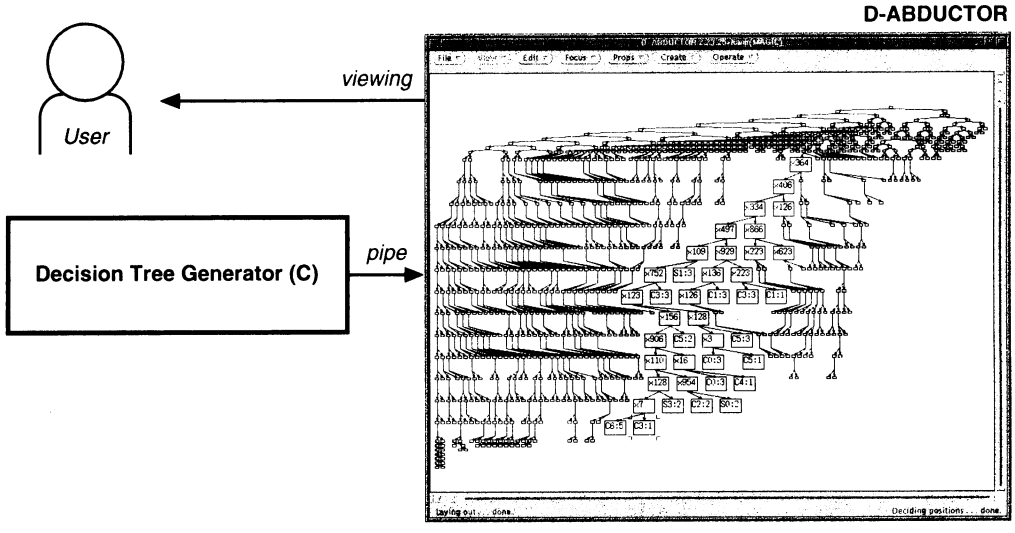


Figure 1: Decision Tree Generator
The sample screen shows a tree really generated by a computer [8]; it has 979 nodes.

The program module generating decision trees, written in C, writes tree data and a layout command in the language Simple. The generated tree data in Simple is passed to the standard input of `abd_tx` through a pipe, and D-ABDUCTOR that is passed simple statements by the `abd_tx` draws the tree on the screen. The user does not need to touch D-ABDUCTOR to view the generated tree.

Decision Tree Generator is one of the simplest applications of D-ABDUCTOR. This style of extension is applicable to many other programs that generate graphs or compound graphs.

Network Monitor

Network Monitor is used for monitoring status of LAN in real time. It shows a graph; a node represents a workstation and an edge represents existence of packets between two nodes, i.e., workstations incident to the edge. Visual attributes of elements represent more detail

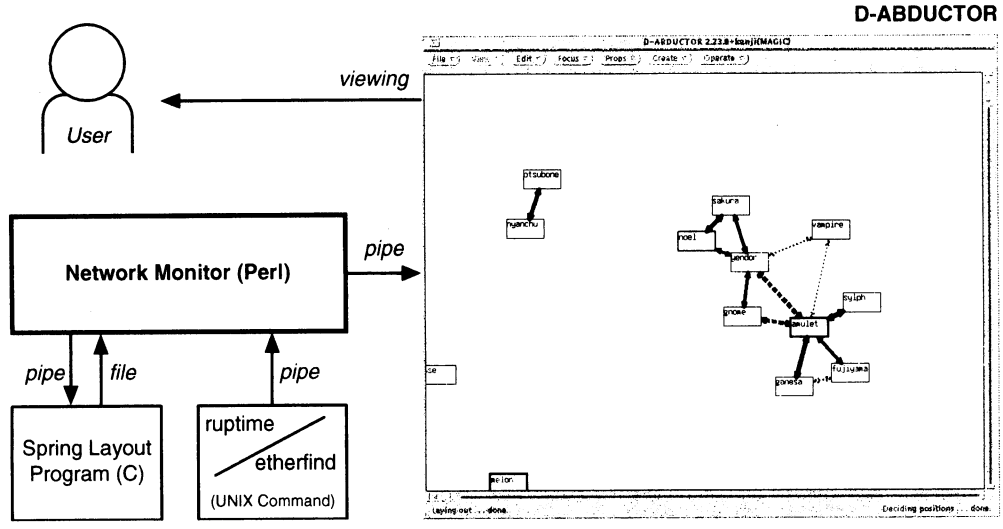


Figure 2: Network Monitor

information. For example, edges with more packets are drawn as wider lines, and nodes (workstations) with higher load average are drawn as wider boundary lines (see Figure 2).

The main module of Network Monitor, written in Perl, gets information about packets on the ethernet by using a UNIX command `etherfind` and load average of each workstation by using `ruptime`. The Perl program receives outputs of these commands through pipes and build a graph data representing current status of LAN. It also invokes external layout program written in C to use a spring layout algorithm [9], which D-ABDUCTOR does not support, then passes Simple statement to move nodes to the standard input of `abd_tx` through a pipe to show new layout of the graph.

Network Monitor uses `etherfind` to construct graphs and an external layout program. By this style of extension, we can employ other commands or programs to construct graphs and to lay them out.

II. Applications with Selecting-Level DUI

Selecting-level DUIs show diagrams and accept selecting information of elements of the diagrams. To make D-ABDUCTOR work as a selecting-level DUI of an application program, the application program have to get log information of D-ABDUCTOR and to pick up select / unselect commands. D-ABDUCTOR can output log information to the standard output, so an application program can know that an element has been selected / unselected by monitoring a pipe.

Directory Browser

Directory Browser is used for management of files and directories. It shows recursive structure of directories by a tree; a node represents a file or a directory and an directed edge represents a directory corresponding where to its tail node (it must be a directory) includes its head node (i.e., a file or a directory). A node for the current directory is drawn in the

largest size, and nodes closer to the current directory are drawn in larger size by using the diagram dressing facility (see Figure 3).

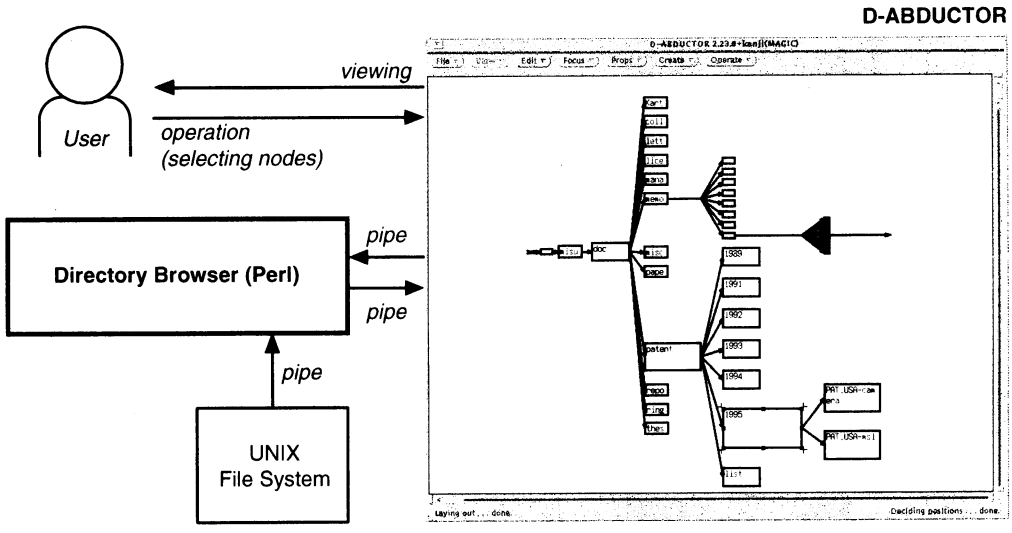


Figure 3: Directory Browser

The main module of Directory Browser, written in Perl, gets information about a directory by communicating with UNIX file system. The Perl program receives directory information through a pipe and build a tree data representing directories the user visited until now. The Perl program is also monitoring log information of D-ABDUCTOR. When the program finds a select command in the log, it collects information about the selected directory, and then draws the contents of the directory.

Directory Browser shows a part of a UNIX file system as a visual tree. By way of this style of extension, we can employ other tree or graph data to be shown as visual representation.

Graphical Hypertext

Graphical Hypertext provides two styles of view for a hypertext. One is an ordinal text view; text is displayed with some anchors for hyper-links. The other is a graph view; a compound graph that shows pages and hyper-links among the pages are displayed, and especially a node for the current page is displayed in larger size (see Figure 4). The user may select anchors to jump according to hyper-links on the text view, while the user may select nodes to jump any pages according to nodes on the graph view. Operations in one view cause immediate change of the other view.

Graphical Hypertext is developed by extending *info* system of Emacs. We first prepared an emacs-Lisp package (`abd.el`), which collects functions to communicate with D-ABDUCTOR. By using this package, the application programmers do not need to take thought of `abd_tx` and log information. The extended version of the info system is `gds-info.el`, which calls functions in the package `abd.el`.

The idea of graphical (dual-view) hypertext should be also applicable to other hypertext systems, for example, the World Wide Web (WWW). For example, if we employ WWW mode of Emacs, we can use the package `abd.el` and thus it should be easy to extend the original program of WWW mode.

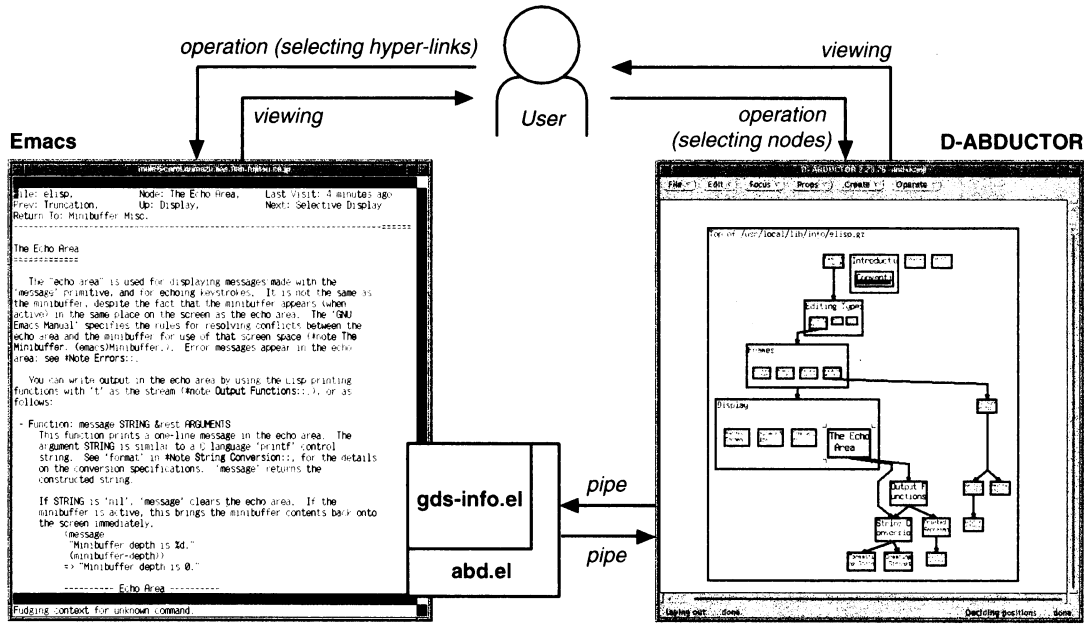


Figure 4: Graphical Hypertext System GDS-info

III. Applications with Manipulating-Level DUI

Manipulating-level DUIs show diagrams and accept manipulation of the diagrams. To make D-ABDUCTOR work as a manipulating-level DUI of an application program, the application program has to get log information of D-ABDUCTOR to know change of diagrams and to change application data.

If an application does not need to have specific application data, the application program can be constructed by collecting variety of commands for diagram manipulation. These commands can be added to menus by the users.

Graphical Outline Processor

Graphical OutLine processor (GOL) [10] provides two styles of view for a document (see Figure 5). One is a text view; outline of a document is displayed with some indentations representing section structure. The other is a graph view; an ordered compound graph [10], which represents section structure, the order of paragraphs, and reference relationships among paragraphs, is displayed. These two views appear simultaneously on a screen. The user can edit text not only by operating the outline text but also by manipulating diagrams in the graph view. Operations in one view change the text and cause immediate change of the other view.

GOL is developed by extending an outline mode of Emacs. Extended version of the outline mode is `gol.el`, which also calls functions in the package `abd.el`.

Structure Analyzer

An example of applications without specific application data is a "structure analyzer." It is used to analyze some relational data by manipulating them from points of combinatorial

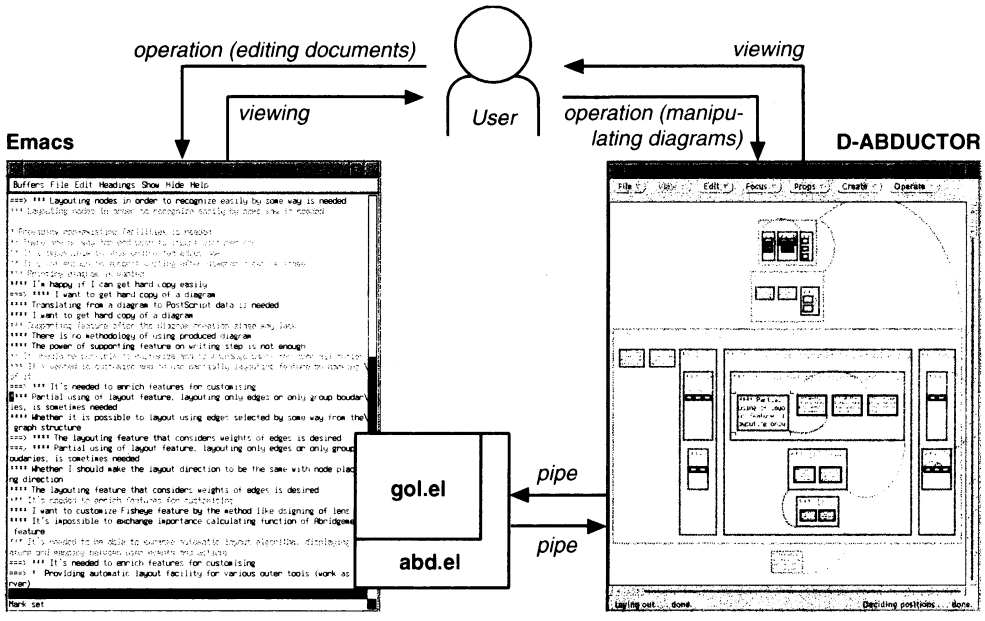


Figure 5: Graphical OutLine processor GOL

view and metrical view.

Hybrid Idea Processing System HIPS, which is a kind of structure analyzer, can be used to organize / reorganize segments of text data, for example, answers of questionnaires. It provides several automatic layout functions, a shortest-path finding function, a reachability check function, and so on. The sample screen in Figure 6 shows a graph, which has nodes with answers for a questionnaire and edges represent relationships among answers. After being found the shortest path between two interesting answers, the graph has been laid out by using the magnetic-spring algorithm [11, 12]. The path is oriented to toward the right, while most part of the graph appears to be free from any special coordinate systems.

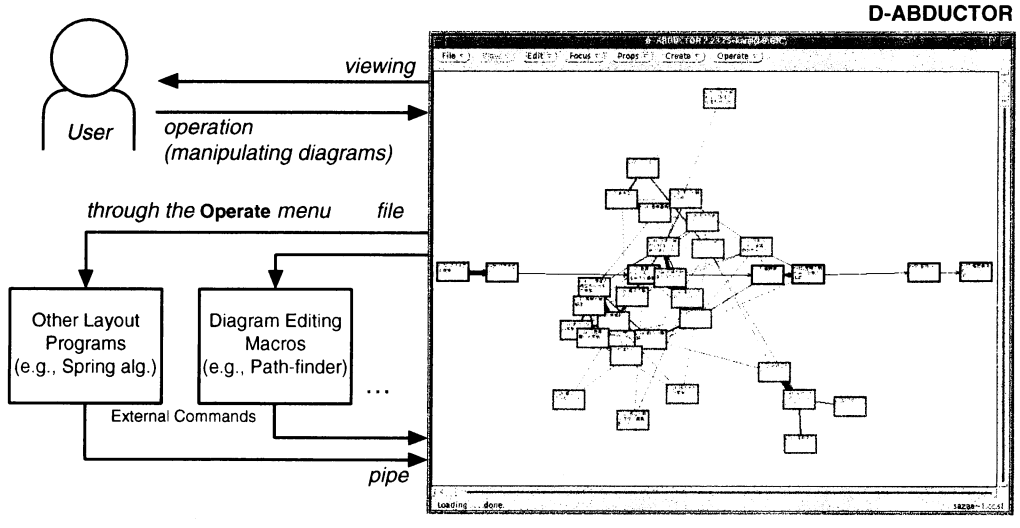


Figure 6: Hybrid Idea Processing System HIPS

Commands such as the magnetic-spring layout function and shortest-path finder are

prepared as external programs written in C, C-Shell, awk, Perl, and so on. Such commands can be issued through a menu. All procedures the users have to do to add commands to menus is to redefine shell environment variables.

5 Conclusions and Future Work

We have had experience of developing several graph drawing applications by using D-ABDUCTOR. In this paper we classified three levels of DUI, and then showed several applications with DUI in each of the three levels to explain how a DUI platform can be customized or extended for the applications.

There are several systems we should mention as related systems. Examples are EDGE [13], daVinci [14] and GraphEd [15]. By using one of these systems, it might be possible to develop the applications we described above. Differences of these systems including D-ABDUCTOR we should remark are ways to customize or to extend them. Features of extensibility of D-ABDUCTOR are summarized as (1) using the language Simple as a communication protocol between a DUI and application programs, and (2) an independent process (program) is invoked for each of application programs and additional functions. These features have the following advantages:

1. The programming language to develop additional functions is not limited to a certain language, so we can employ a suitable language to develop new facilities in the easiest way.

2. Each of application modules and additional functions is managed as a separate program, so it is easy to exchange facilities.

3. Each of application modules and additional functions is executed as a separated program, so it is easy to maintain the whole system.

We hope DUI platforms provide many facilities and generality. If we integrate all expected facilities into a system, the system must grow to a large and complicated program. It is often difficult to obtain the above advantages for large and complicated programs. We should employ some other approaches to provide many and general facilities without enlarging the system. We propose a policy of development of such the systems:

- Make it easy to develop new facilities.
- Make it easy to exchange facilities for modification of the system.
- Make it easy to maintain the system.

The current version of D-ABDUCTOR will be extended and developed in line with the above policy. With the new system, we will be able to add or exchange most functions by using plug-in module mechanisms. So it should become very easy to customize the system by exchanging functions such as automatic graph layout, fisheye display, communication with application programs, and so on.

The development code of the next system is "Pizza System." The Pizza System consists of a "Plain Pizza," which is a slim platform of DUI, and "Toppings," which are function modules exchangeable according to the user's preference. The users choose some Toppings for their applications and can easily make application systems, "Mixed Pizza," only by putting the Toppings on a Plain Pizza.

Acknowledgment

The authors would like to thank Professor Peter Eades of the University Newcastle for his useful comments.

References

[1] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. Algorithms for automatic graph drawing: An annotated bibliography. Technical report, Department of Computer Science, Brown University, 1993. avaiable via anonymous ftp from wilma.cs.brown.edu (128.148.33.6), files /bup/gdbiblio.tex.Z and /pub/gdbiblio.ps.Z.

[2] Tao Lin. *A General Schema for Diagrammatic User Interfaces.* PhD thesis, Department of Computer Science, The University of Newcastle, 1993.

[3] Kazuo Misue. D-ABDUCTOR 2.0 user manual. Research Report IIAS-RR-93-9E, FUJITSU LABORATORIES, IIAS, 1993.

[4] Kozo Sugiyama and Kazuo Misue. A generic compound graph visualizer / manipulator: D-ABDUCTOR. In *Graph Drawing, Symposium on Graph Drawing, GD '95, Passau, Germany, September 20 - 22, 1995*, pages 500–503, 1996. Lecture Notes in Computer Science 1027, Springer Verlag.

[5] Kazuo Misue, Peter Eades, Wei Lai, and Kozo Sugiyama. Layout adjustment and the mental map. *Journal of Visual Languages and Computing*, 6(2):183–210, 1995.

[6] Kozo Sugiyama and Kazuo Misue. Visualization of structural information: Automatic drawing of compound digraphs. *IEEE Trans. SMC*, 21(4):876–892, 1991.

[7] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[8] Takeshi Koshiba. Decision tree learning system with switching evaluator. In *Proceedings of the 11th Biennial Conference of the Canadian Society for Computational Studies of Intelligence (AI '96), Lecture Notes in Artificial Intelligence 1081, (Ed.) Gordon McCalla, Springer*, pages pp. 349–361, 1996.

[9] Peter Eades. A heuristics for graph drawing. *Congressus Numerantium*, 42:146–160, 1984.

[10] Kiyoshi Nitta, Robert Inder, Kazuo Misue, and Kozo Sugiyama. GOL: Graphical outline processor — simultaneously using a text view and a graph view. In *Asia Pacific Computer Human Interaction (APCHI'96)*, June 25-28 1996. to appear.

[11] Kozo Sugiyama and Kazuo Misue. A simple and unified method for drawing graphs: Magnetic-spring algorithm. In *Graph Drawing, DIMACS International Workshop, GD '94, Princeton, New Jersey, USA, October 1994, Proceedings*, pages 364–375, 1995. Lecture Notes in Computer Science 894, Springer Verlag.

[12] Kozo Sugiyama and Kazuo Misue. Graph drawing by the magnetic spring model. *Journal of Visual Languages and Computing*, 6(3):217–231, 1995.

[13] Frances Newbery Paulisch and Walter F. Tichy. EDGE: An extendible graph editor. *Software - Practice and Experience*, 20(S1):S1/63–S1/88, 1990.

[14] M. Froehlich and M. Werner. Demonstration of the interactive graph visualization system davinci. In *Graph Drawing, DIMACS International Workshop, GD '94, Princeton, New Jersey, USA, October 1994, Proceedings*, pages 266–269, 1995. Lecture Notes in Computer Science 894, Springer Verlag.

[15] Michael Himsolt. GraphEd: A graphical platform for the implementation of graph algorithm. In *Graph Drawing, DIMACS International Workshop, GD '94, Princeton, New Jersey, USA, October 1994, Proceedings*, pages 182–193, 1995. Lecture Notes in Computer Science 894, Springer Verlag.