Chapter 16
A Framework for Developing Diagram Applications

Wei Lai
Swinburne University of Technology, Australia

Weidong Huang
CSIRO ICT Centre, Sydney, Australia

ABSTRACT

This chapter presents a framework for developing diagram applications. The diagrams refer to those graphs where nodes vary in shape and size used in real world applications, such as flowcharts, UML diagrams, and E-R diagrams. The framework is based on a model the authors developed for diagrams. The model is robust for diagrams and it can represent a wide variety of applications and support the development of powerful application-specific functions. The framework based on this model supports the development of automatic layout techniques for diagrams and the development of the linkage between the graph structure and applications. Automatic layout for diagrams is demonstrated and two case studies for diagram applications are presented.

1. INTRODUCTION

A graph typically consists of a set of nodes and a set of edges where a node is drawn as a point and an edge is drawn as a line. We are more interested in diagrams which refer to practical graphs. In a practical graph, nodes (e.g. boxes and circles) representing objects take some spaces. They are not abstract points in a classical graph. Diagrams are commonly used to model relations in information systems and software engineering. Examples are data flow diagrams, state transition diagrams, flow charts, PERT charts, organization charts, Petri nets, entity-relationship diagrams, and UML diagrams.

A number of graph drawing algorithms have been developed, these algorithms are listed in (Battista et al 1999). These algorithms produce aesthetically pleasing abstract graph drawings, where nodes are just points in the plane. In prac-

DOI: 10.4018/978-1-4666-4309-3.ch016

Copyright © 2014, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

tice, nodes have many attributes; for example, in a UML diagram, a node has several textual labels in several fields. These attributes need to be represented graphically, for instance, as shapes, sizes, and colors. We call this kind of graph where nodes vary in shape and size a practical graph.

Most diagrams in applications are practical graphs (e.g. UML diagrams). Diagrams are widely used in information visualization. The motivation for information visualization is that relational information, once handled textually, is now commonly displayed and manipulated with diagrams. We call this kind of information visualization using diagram displays and manipulations *diagram applications*. Diagram applications can be divided into two categories:

1. Those which focus on translation of textual information into a diagram, such as translation of a program to a flow chart, and translation of URL relations to a web graph.
2. Those which focus on semantic interpretation of diagrams, such as interpretation of a flow chart to execute a program, code generation from a UML diagram, and using diagrams for data mining.

Both categories of applications need diagram layout interfaces. It is critical to have an efficient tool for creating diagram layout interfaces. Although there are some software tools (such as Visual Basic, Xlib, and Motif) for windows and GUI programming, they are very low-level and not efficient for interactive diagram applications. These tools support building those interface components, such as menus, scrolling bars, dialog boxes, and so on. However, they are not efficient to support building diagram layout interfaces.

Most existing CASE (Computer Aided Software Engineering) tools, such as Rational Rose (7), provide graphical editors for drawing UML diagrams. However, these tools were designed for UML diagram applications only and they cannot be used for various diagram applications. Also, these CASE tools do not efficiently support automatic diagram layout. Automatic layout can release the user from the time-consuming and detail-intensive chore of generating a readable diagram.

This chapter presents a framework for developing diagram applications. It includes the investigation of a robust model for diagrams to solve the problems mentioned above. This model can support automatic layout techniques for diagrams and the development the linkage between the diagram structure and applications.

The critical issue is that what kind of model is suitable to represent diagrams where nodes vary in size and shape and can support the development of diagram layout functions? Is current classical graph model sufficient in playing this role? Can current graph drawing algorithms be applied to diagram layout? In next section, the background is introduced for classical graph model and graph drawing algorithms, and the questions above will be answered.

2. BACKGROUND

Many graph drawing algorithms (Battista et al., 1999) have been developed for abstract graph automatic layout. They are based on the classical graph model, that is, G=(V, E), where V is a set of abstract nodes (points) and E is a set of edges (lines). However, there is little work have been done on automatic layout for practical graphs (i.e. diagrams) used in real world applications, especially for nodes varying in size and shape in practice.

The most difficult problem for diagram interfaces is *layout*-assigning a position for each node and a curve for each edge. The assignment must be chosen to make the resulting picture easy to understand and easy to remember. A good layout can be like a picture-worth a thousand words; a poor layout can confuse or mislead the user. This problem is called the *graph drawing problem*. A number of graph drawing algorithms have been developed. They produce aesthetically pleasing graph layouts (see Figure 1a). These algorithms

8 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/a-framework-for-developing-diagram-applications/78726

Related Content

SBASH Stack Based Allocation of Sheer Window Architecture for Real Time Stream Data Processing
Devesh Kumar Laland Ugrasen Suman (2020). *International Journal of Data Analytics (pp. 1-21).*
www.irma-international.org/article/sbash-stack-based-allocation-of-sheer-window-architecture-for-real-time-stream-data-processing/244166

Using Data Envelopment Analysis to Construct Human Development Index
Paulo Nocera Alves Junior, Enzo Barberio Marianoand Daisy Aparecida do Nascimento Rebelatto (2017). *Emerging Trends in the Development and Application of Composite Indicators (pp. 298-323).*
www.irma-international.org/chapter/using-data-envelopment-analysis-to-construct-human-development-index/165657

Remote Patient Monitoring for Healthcare: A Big Challenge for Big Data
Andrew Stranieriand Venki Balasubramanian (2019). *Managerial Perspectives on Intelligent Big Data Analytics (pp. 163-179).*
www.irma-international.org/chapter/remote-patient-monitoring-for-healthcare/224338

A New Internet Public Opinion Evaluation Model: A Case Study of Public Opinions on COVID-19 in Taiwan
Sheng-Tsung Tu, Louis Y. Y. Lu, Chih-Hung Hsiehand Chia-Yu Wu (2021). *International Journal of Big Data and Analytics in Healthcare (pp. 1-17).*
www.irma-international.org/article/a-new-internet-public-opinion-evaluation-model/287603

Using Intelligent Agents Paradigm in Big Data Security Risks Mitigation
Mihai Horia Zaharia (2019). *Managerial Perspectives on Intelligent Big Data Analytics (pp. 76-97).*
www.irma-international.org/chapter/using-intelligent-agents-paradigm-in-big-data-security-risks-mitigation/224333