2012 4th International Conference on Computer Engineering and Technology (ICCET 2012)
IPCSIT vol.40 (2012) © (2012) IACSIT Press, Singapore

Dependency Modeling and Data Processing based on AddFlow

Guo Yousong [1 +], Chen Shengjian [1], Xu Jun [1]

[1] Academy of Armored Forces Engineering, Beijing, China

Abstract. A friendly user interface can be good to the end users. Though it probably means more work to do as a developer, especially there are not enough UI components to be used for a special purpose as mentioned in this paper, the graphic modeling environment for dependency analysis. To reduce the related cost of software development, an approach to graphical modeling based on AddFlow is proposed. Some important features of AddFlow are introduced and the algorithm for extracting the dependencies from the graph is given to construct the dependency structure matrix. This approach is proved to be effective through an experimental project.

Keywords: dependency modeling, dependency structure matrix, AddFlow

1. Introduction

In the development of a product, a collection of tasks is performed. These tasks have dependencies on one another, either because of physical objects that must flow from task to task, or because of information that one task requires and which another task provides [1]. Usually a dependency that represents a direct causal relationship is called "First Order Dependency", while a dependency that represents an indirect causal relationship is called "Nth-Order Dependency". A Full Order Dependency will represent all dependencies that are directly or indirectly responsible for a given event [2]. To represent these dependencies, the "Dependency Structure Matrix" (DSM) was invented. Nowadays it has been a widely used powerful technique in project planning, system engineering, hardware analysis and software architecture management. In general, the dependencies should be given at first in order to construct the DSM. And to give a better experience to the end users, a well designed User Interface (UI) should be necessary. The companies who have designed the operation systems such as Microsoft Windows or Apple Mac OS often provide a set of standard UI components for the developers. But that's not enough in most areas mentioned above. Developers may design their own UI components, leading to high cost of development. Or they can use some commercial components or software development kits.

In this paper, an ActiveX control named "AddFlow" is introduced into an experimental project. The AddFlow is a product of Lassalle Technologies. It is designed for most flowchart enabled applications. Compared to other components or SDKs, this ActiveX control is lightweight, feature-rich and simple to use in practical project.

2. Implementation

The project needs a well designed, friendly and interactive user interface for the end users, who will be able to describe and draw the dependency graphs. As a result, a DSM will be constructed based on the dependencies extracted from the graph. The AddFlow provides two ways in building a diagram, the interactive way and the programmatic way. The programmatic way is based on Application Programmatic Interface (API), mainly used in design stage [3], while the interactive way may be a better choice. For the developers, all they need to do is draw and place the icon of this component onto the forms in the IDE of Visual Basic or Visual C++ [Fig. 1]. When the project is compiled and executed, a friendly user interface is

+ Corresponding author. *E-mail address*: gyousong@163.com.

ready for the users to finish their job as wishes. Of course, some attributes should be set up if you want to customise the styles, shapes, colour, fonts and pictures for the final appearance. And only a few of methods and events need to be manipulated in order to access the data behind the graph.

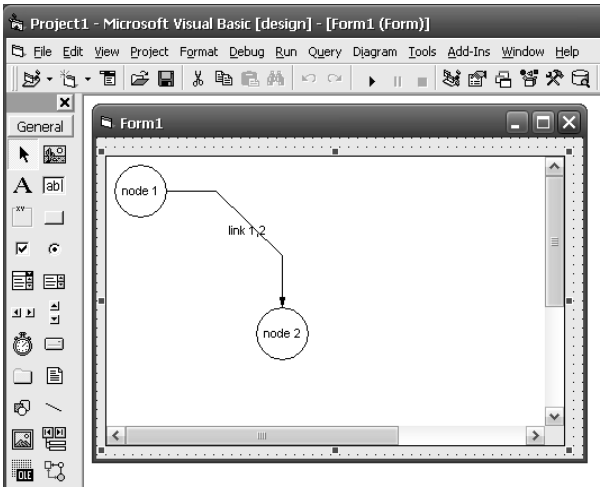

Fig. 1: AddFlow in Visual Basic IDE

An AddFlow diagram is a set of objects linked together. The objects are called “nodes”, connected by objects called “links”. All the nodes and links can be added by programmatic code or by interactive way. Each link object has properties like “Org” and “Dst”, through which the indices of origin and destination nodes will be accessed. All nodes and links can be enumerated, and accessed through the reference of the neighbour nodes or links.

In some cases the extra data needs to be associated with a node or a link. This can be accomplished by using six special properties including “UserData”, “Tag”, “Key”, “TagVariant”, “ToolTip” and “Marked”. These properties can be used to store different type of data for any reason, giving the developers great conveniences.

3. The Algorithm

Here an experimental project has been build for dependency modeling. The users can easily draw and connect the different objects with a mouse for describing the dependencies between them [Fig. 2]. Once the user has finished the modeling process, the dependencies can be extracted from the graph. Since the “Org” property is updated automatically during the drawing procedure, the First Order Dependency can be acquired directly by enumerating all the links. But it is a little more complicated to get the Nth Order or even Full Order Dependency. In order to construct the DSM, we have to design a valid algorithm to obtain these dependencies.

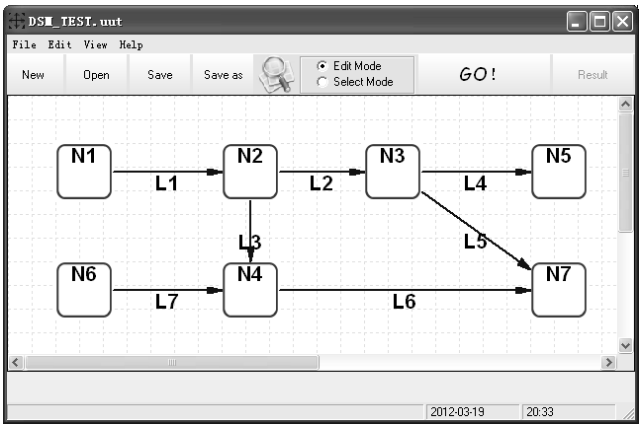

Fig. 2: An Experimental Project based on AddFlow

For one single node, any relationship from any other node should be extracted and recorded. This kind of relationship is like some genetic material. We can have a "genetic" array binding to each node and start enumerating all the nodes in the graph. Once a relation is found, a mark is saved into the genetic array. Keep looping check for relations between any two nodes until no more new relations can be found. Thus a flag variable is needed to indicate when the enumerating should be stopped. [Fig. 3] shows the detailed algorithm. The resulting dependencies will be represented as the indices saved in the genic_array of each node. [Fig. 4] shows the algorithm to construct the DSM based on the dependencies we've got. The resulting DSM [Fig. 5] will be saved in a two-dimensional array named "dsm". The value "1" of dsm[i][j] means the (i)th node will influence the (j)th node, or some information can flows from the (i)th node to the (j)th node.

```
BEGIN OF ALGORITHM
REPEAT UNTIL (termination_condition IS TRUE) DO
   termination_condition = TRUE
   FOR EACH node IN all_nodes DO
      FOR EACH other_node IN all_nodes DO
         IF node IS INFLUENCED BY other_node AND index of
other_node DOES NOT EXIST IN genic_array of node
            SAVE index of other_node INTO genic_array of node;
            termination_condition = FALSE;
         FI
      OD
  OD
OD
END OF ALGORITHM
```

Fig. 3: Algorithm for Extracting Dependencies from the Graph

```
BEGIN OF ALGORITHM
FOR EACH node IN all_nodes DO
   FOR EACH other_node IN all_nodes DO
      IF index of other_node EXISTS IN genic_array of node
         dsm[index of other_node][index of node] = 1;
      ELSE
         dsm[index of other_node][index of node] = 0;
      FI
   OD
OD
END OF ALGORITHM
```

Fig. 4: Algorithm for Constructing DSM based on Extracted Dependencies

$$DSM = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Fig. 5: The Final DSM

4. Conclusion

Using AddFlow in the developments of dependency modeling software can give the developers great help in designing friendly UIs effectively and quickly. By specially designed algorithms, the dependencies can be extracted from the graph, and the DSM can be constructed based on the dependencies. This approach may be a good choice for all the flow graph based applications.

5. References

[1] N. Sangal, E. Jordan, V. Sinha, D. Jackson. Using Dependency Models to Manage Complex Software Architecture. *The International Conference on Object Oriented Programming, Systems, Languages and Applications (OOPSLA)*, 2005.

[2] Basic Dependency Modeling Terminology. *http://www.testability.com/Reference/Glossaries.aspx?Glossary=DependencyModeling*

[3] AddFlow Documents. *http://www.lassalle.com/*