A Software Engineering Approach to Multimedia Presentation Designs

Timothy K. Shih, Y. H, Wang, C. H. Kuo
D. R. Jiang, and Jason C. Hung
Dept. of Computer Science and
Information Engineering
Tamkang University
Tamsui, Taiwan 251, R.O.C.
email: TSHIH@CS.TKU.EDU.TW

Wen C. Pai and C. C. Wang
Kuang Wu Institute of
Technology and Commerce
Paitou, Taipei
Taiwan 112
R.O.C.

Abstract

In this paper, we discuss a new authoring system for the stepwise refinement of multimedia presentations. The proposed system is based on data flow and control flow diagrams, with many enhanced notations especially useful for multimedia presentation designs. Presentation designers can use our system to quickly develop a prototype presentation, and refine it step by step toward a final presentation. A multimedia DFD/CFD of multiple levels is to help a presentation designer to analyze the script structure of a presentation. However, it is not powerful enough to define the precise schedule or layout of a presentation. Incorporated with an interactive multimedia Petri net diagramming mechanism, the last level of a presentation window is refined to a Petri net, which describes the temporal behavior of a presentation window. Our interactive multimedia Petri net is a variation of timed Petri net, with the addition of User Transitions and Sync Arc. Since a multimedia presentation is interactive, we introduce the above two objects for participant dependent synchronization. Using our system, a presentation designer is able to analyze his/her presentation script and design the presentation in a systematic manner.

Key words: Multimedia Presentation, Software Engineering, Data Flow Diagram, Structured Analysis, Petri Net

1 Introduction

Structured analysis and design have been used for quite a while in software development. A system engineer analyzes the requirements of a software system and constructs a system specification. Based on this specification, the engineer uses data flow diagrams or control flow diagrams to design a functional specification and the preliminary design of a software system. The design is then refined stepwise toward a structured chart before the implementation proceeds. This Water Fall paradigm [4] has several benefits. Firstly, with the help of a CASE tool, an engineer is able to clearly organize the data flow and the transition control of a system. The engineer does not need to have the detail at the beginning. The initial design of a system can be refined step by step in a multi-level data flow/control flow diagram. This methodology allows the engineer to focus on a level of detail each time without being loose in the complexity of the system. Moreover, a data flow diagram allows the engineer to precisely specify the type of data each part of the system needs in a data dictionary. this dictionary is then used as a base in the design of a program's data structures. After the structured chart is derived from the data flow diagram, the implementation of the system can be constructed more efficiently. These benefits make a data flow/control flow diagram become a useful tool in the process of software development.

The development of a multimedia presentation involves the analysis of a presentation script, the collection of multimedia resources, and the realization of a presentation layout and navigation control flow. In line with the growing needs of multimedia presentations, many authoring systems were developed. However, in the community of multimedia computing, there is no discussion of the need of a good presentation development methodology. We have surveyed a number of presentation design tools. We found that, not many authoring system focus on the concept of stepwise refinement. In this paper, we discuss a new authoring system for this purpose. The proposed system is based on data flow and control flow diagrams, with many enhanced notations especially useful for multimedia presentation designs. The proposed notations and the methodology are implemented in a prototype software run on Microsoft Windows 95/3.1.

This paper is organized as the following. In section 2, we survey a number of well-known authoring packages. To run a multimedia presentation, we have developed a Multimedia Abstract Machine (MAM) which is based

on a timed Petri net model. A short discussion of MAM is given in section 3. The proposed multimedia data flow/control flow diagram is discussed in section 4. Section 5 proposes a new direction of multimedia presentation designs. Unlike other systems, which generates static presentations, our system allows dynamic multimedia presentations. The graphical user interface design of our system is given in section 6. A short conclusion summarizes our contributions is also presented in section 7.

2 Related Works

Before we design our system, we have surveyed both academic researches and industrial software packages. Sony Corporation developed a hypermedia prototype system (SAL) [3] for multimedia authoring, which is based on a link and node model used in most authoring systems. The Layered Multimedia Data Model (LMDM) [5] allows the reuse of presentation templates which is important for improving the efficiency of multimedia presentation designs. LMDM has a number of strengths including the support of a general model of media synchronization, limited system dependencies, and the generalization of a traditional animation model. The work discussed in [1] proposes an architecture and data model for integrated multimedia presentations. The architecture provides a homogeneous strategy to access, process, and exchange multimedia documents generated by different authoring and presentation systems. Diamond [10] is a multimedia message system built on a distributed architecture for creating, editing, transmitting, and managing multimedia documents. Multimedia documents are stored in folders in a distributed database. An open hypermedia system is discussed in [2]. The system supports heterogeneous multimedia data types and allows new types to be added. A platform independent multimedia presentation composition system is discussed in [11]. In the system, a script based approach is used to model the object-oriented presentations as well as user interactions.

We also looked at the following commercial products related to multimedia authoring or presentation designs:

1. Authorware by Macromedia, Inc.
2. Multimedia Viewer by Microsoft
3. Multimedia Toolbook by Asymetrix Corporation
4. Hypermedia Authoring and Playback System by ITRI (Taiwan)
5. Action! by Macromedia, Inc.
6. Audio Visual Connection by IBM
7. Astound by Gold Disk Inc.

Authorware uses an event control flow diagram allowing the presenter to specify presentation objects and controls, which can be decomposed into several levels in a hierarchy structure. The system also provides a simple script language for calculation and data manipulation. Other systems (i.e., 2, 3, and 6 above) also provide script languages and API (application program interface) functions. Hypermedia System and Action! use a time line table allowing actions or objects be dropped in a particular time slot. Most systems allow users to cut and paste presentation objects or actions via button click and drawing. Multimedia Viewer also provides a set of medium editing tools. Presentation objects produced by these tools can be linked together by a script language supporting functions, data structures, and commands.

None of the above system, however, allows dynamic presentations. That is, a presentation generated by the above systems will stay as the form it was created. Different audience watches the same presentation over and over again. If a presentation can take user interactions and mutate itself, the presentation is more diversified. And this is the main strength of our system. Moreover, not many systems focus on the stepwise refinement of presentation designs. A presentation designer who uses the above systems must have his/her presentation script ready before using the systems. On the other hand, our system helps the designer to design his/her presentation step by step until the final version is created.

3 A Multimedia Abstract Machine

The main theme of our system is a timed Petri Net based multimedia abstract mechine (MAM). MAM is a software architecture and system for multimedia documentation demonstrations and presentations. We realized that most multimedia information processes consist of a sequence of atomic steps, such as opening a sound channel, transfering a block of video data from buffer to the graphic memory page, closing a MIDI sequencer, etc. These atomic steps, from the perspective of multimedia computing, are non-separable items. This concept is relatively similar to assembly language instructions, which are atomic steps of a procedural program. We have the MAM instruction set defined [6]. Some instructions control multimedia hardware devices while others support user interactions. A multimedia presentation designed by using our multimedia DFD/CFD and Petri net design tool has a number of Petri nets and navigation messages. These Petri nets are represented in a MAM assembly program, which is produced by an internal program translator. Therefore, the designed presentation is runnable on the MAM. Figure 1 pictures the overall software environment of the MAM.

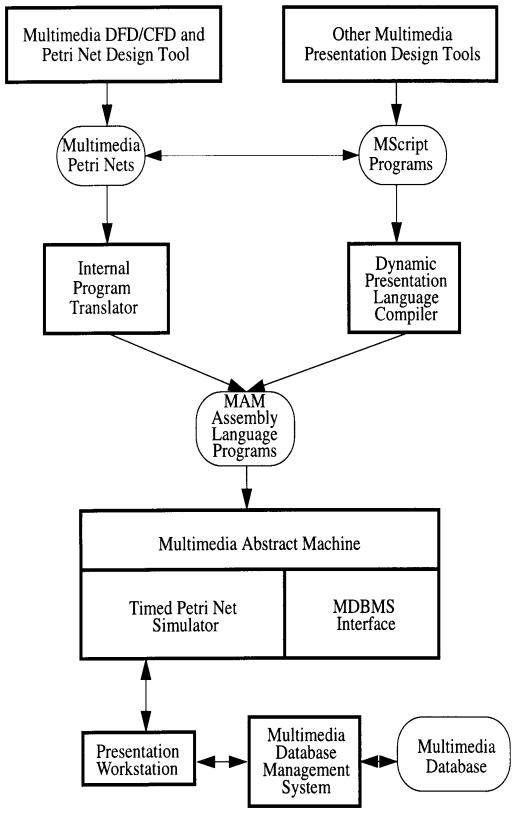

Figure 1: The software environment of our Multimedia Abstract Machine

The development of MAM has multiple purposes. We have developed a multimedia script language. The language serves as an intermediate representations of various formats of multimedia presentations which we have developed [9, 7]. These representations are compiled to the MAM instructions. Thus, it is possible to link multimedia presentations produced by different tools to make an integrated presentation. On the other side of the MAM environment is a multimedia DBMS [8], which we are trying to integrate to the environment. The objective of this MDBMS, from the perspective of MAM, is to assist a presentation to quickly locate the multimedia resources required in the demonstration.

Since MAM is an integrated multimedia programming environment with a relatively large scope, in this paper, we focus on the discussion of the multimedia DFD/CFD and Petri net design tool. Other components in the environment are omitted.

4 Multimedia DFD/CFD

In this section, we propose a revised DFD/CFD mechanism for multimedia presentation designs. The design of a multimedia presentation, utilizing message passing for navigation, has a similar concept to writing a software specification. The proposed multimedia data flow/control flow diagram is based on DFD and CFD, with the extension of some new objects:

- **Multimedia Resource**: similar to the data store in a DFD, a multimedia resource is denoted by a pair of thick parallel lines.
- **State Variable**: besides resources, an interactive presentation may contain state variables store presentation data, such as the audience's name. A state variable is represented by a pair of thin parallel lines. State variables not only keep information, but also control *dynamic presentations* (to be discussed in section 5).
- **Resource Data**: similar to the data link in a DFD, resources are passed to a presentation program by a link with a regular arrow.
- **Navigation Message (NM)**: similar to the control link in a CFD, a navigation message is passed by a link with a light-weighted arrow.
- **Dynamic Mutation**: to support dynamic multimedia presentations, the dynamic mutation link is introduced, which is represented by a curved arrow. There are four types of mutations in a multimedia DFD/CFD:
  - State Variable Change
  - Layout Change
  - Resource Change
  - Navigation Change

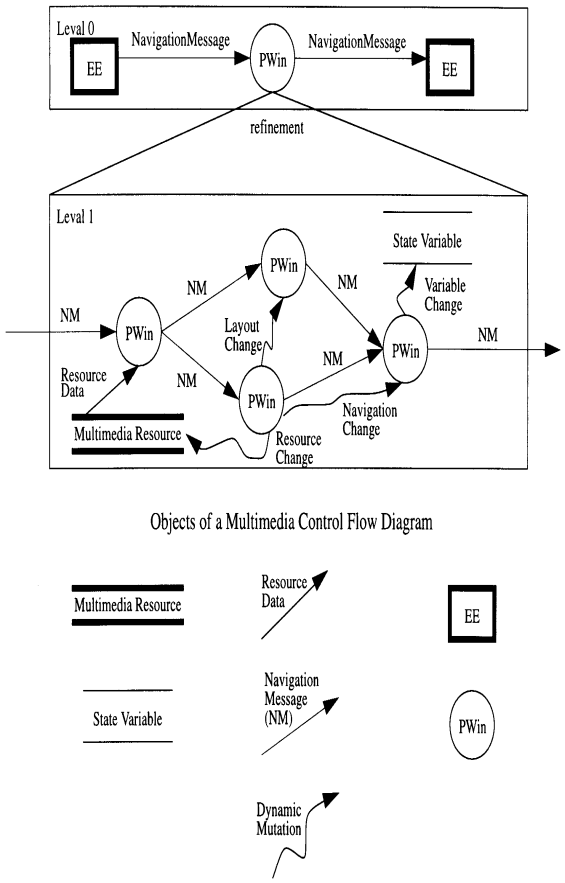

Figure 2: Components of a Multimedia DFD/CFD

These links mutate the content and appearance of a multimedia presentation. When we discuss the stepwise refinement of a presentation, we will define these mutations in detail.

- **External Entity (EE)**: similar to one in DFD/CFD, an external entity could represent a user, a hardware device, or another system which pass data/controls to the multimedia presentation. An EE is denoted by a box surrounded by thick lines.
- **Presentation Window (PWin)**: similar to a process in a DFD, a presentation window is denoted by a circle.

A multimedia presentation contains a number of presentation windows. Each presentation window contains a layout definition, a collection of multimedia resources (sharable among presentation windows), some state variables, and nevigation control messages. Figure 2 illustrates the various types of components of a multimedia DFD/CFD.

A presentation window must be refined. Stepwise refinement of a presentation allows the presentation to be specified in different levels of details. The system allows the user to use multimedia DFDs/CFDs, with the help of a drag and drop mechanism, to design a multimedia presentation. The system also provides a connection to a multimedia database [8]. The multimedia database resource browser [8] allows the user to select multimedia resources needed for a presentation.

A multimedia DFD/CFD of multiple levels is to help a presentation designer to analyze the script structure of a presentation. However, it is not powerful enough to define the precise schedule or layout of a presentation. Incorporated with an interactive multimedia Petri net diagramming mechanism, the last level of a presentation window is refined to a Petri net, which describes the temporal behavior of a presentation window. An example showing the last level refinement is shown in figure 3. Our interactive multimedia Petri net is a variation of timed Petri net, with the addition of *User Transitions* and *Sync Arc*. Since a multimedia presentation is interactive, it is natural for us to introduce the above two objects for participant dependent synchronization.

A timed Petri net is a bipartite graph with two types of nodes: the transition nodes and the place nodes. A transition controls synchronization and a place holds a token and a time duration. A transition is fired only after each place adjacent to the transition releases the token. A place holds a multimedia resource to be played for the time duration. Transitions and places are connected by *sync arcs* in our revised Petri net. We add *user transitions* and *user arcs* to the timed Petri net. A user transition receives a navigation message from the user before it is fired. A user transition is directly connected to some transitions. The activation of the user transition interrupts the demonstration of the presentation window and causes the activations of the connected transitions simultaneously. For instance, in figure 3, the

activation of the only one user transition (on the lower-left corner) causes transitions "c" and "d" to be fired together, which makes "Music2", "Map", and "Shopping" resources to be demonstrated. Note that, the resulting presentation is missing the "WallPaper" resource since transition "a" was not fired. In a normal situation, when transition "a" is fired, the presentation proceeds according to a pre-defined schedule. However, the existence of a user transition may cause the rearrangement of a presentation schedule. Using these user transitions and navigation messages, the revised Petri net is able to accept user interactions.

The last level of the refinement between a presentation window and a Petri net needs to maintain two types of balances: the navigation message balance and the multimedia resource/state variable balance. That is, the incoming and outgoing messages of a presentation window have to match those of the refined Petri net. Similarly, the multimedia resources and state variables used have to be the same. The last level of a multimedia presentation refinement has a number of interactive multimedia petri nets. The followings are components of the Petri nets:

- **Transition**: is for synchronization control. Transitions are shown as vertical dark bars.
- **User Transition**: accepts a message and causes the activation of connected transitions. User transitions are shown as vertical light bars.
- **Place**: is for playing a multimedia resource. Places are shown as ellips.
- **User Arc**: connects from a user transition to a transition. User arcs are shown as straight arrowheaded lines.
- **Sync Arc**: connects from a place to a transition, or from a transition to a place. Sync arcs are shown as curved arrowheaded arcs.

In the multimedia presentation design system, we allow six type of resources: sound, MIDI, video, animation, picture, and text resources. Some of these resource types are static (i.e., picture and text) while others are dynamic. Dynamic multimedia resources come with a resource duration. Within the time period, repeated steps are performed to load data, such as video frames, from disk to the video memory page. As illustrated in figure 4, we defined four subnets for the six type of resources. These subnets consists of atomic execution steps, such as preparing a sound card, or writing data to sound buffer. Each of these atomic step is corresponding to an assembly instruction of the MAM. Some of the subnets (e.g., the video subnet) use transitions for synchronization control. All subnets of dynamic resources have an iterative counter (IC) calculating the duration of the resource demonstration.

We have discussed the global view of our diagramming mechanism. However, a multimedia presentation

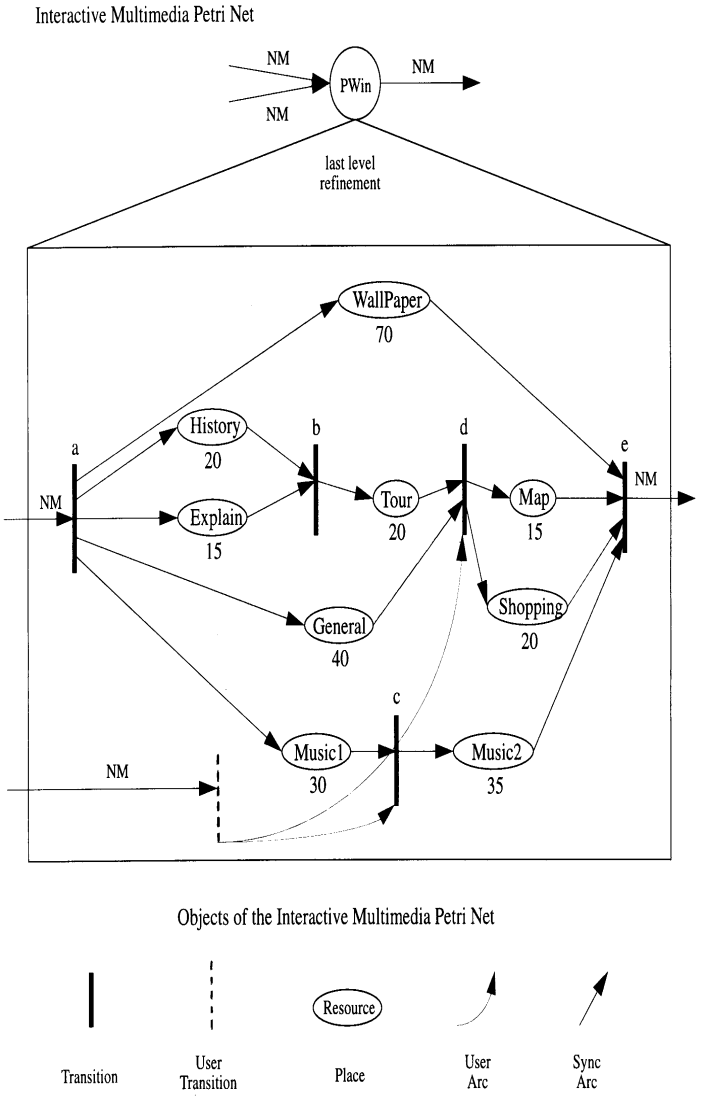

Figure 3: An Interactive Multimedia Petri Net

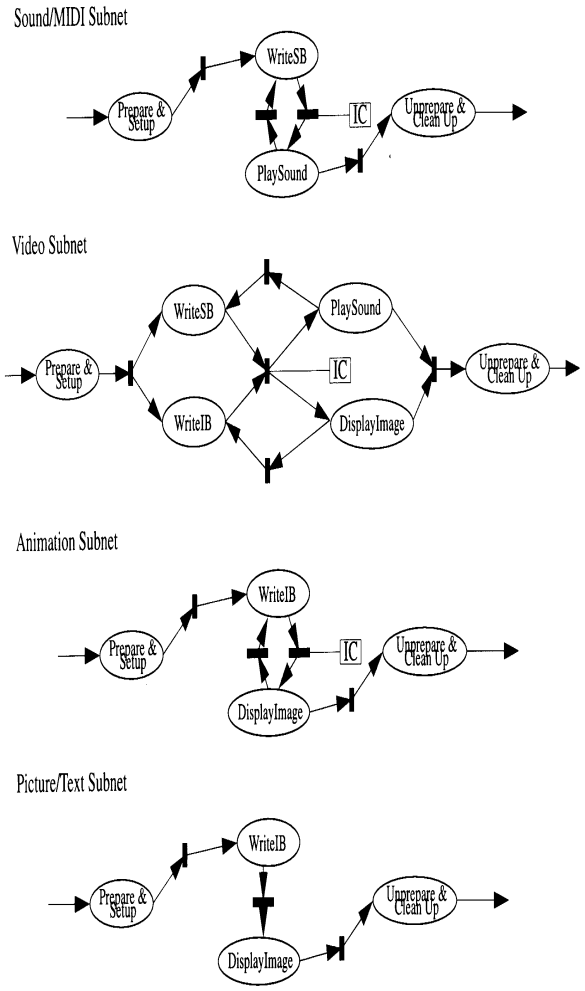

Figure 4: Subnets for Different Multimedia Resources

in our system is dynamic and mutable. In the following section, we propose other diagramming techniques to achieve our goal – allowing dynamic multimedia presentations.

5 Dynamic Presentations

Usually, when a multimedia presentation is designed, the usage of resources, the layout, and the navigation sequences are all fixed until the presentation is re-designed again. We want to improve this approach by allowing dynamic replacement of resources, layouts, and controls of the presentation. This makes the presentation generator have the ability of computing the presentation representation at the run-time. Possible dynamic events are:

- asserting/retracting of information
- changing resources
- changing layouts
- changing navigation controls

Figure 5 illustrates a mutable Petri net. On the top of the revised timed Petri net (see figure 3), we further add the following components:

- **Selection**: selects one of the outgoing navigation messages. A selection could be a push button or a key of the presentation window. A selection holds an internal representation of which outgoing navigation message will be sent upon the activation of the selection. A selection is represented as a circle labeled with an "S" in the Petri net diagram.
- **Assignment**: assigns a value to a state variable. An assignment is connected to a state variable. When an assignment received a navigation message (with a parameter holding a value), the assignment set the state variable to the value. An assignment is denoted by a box labeled with the "Var ¡= Val" sign.
- **Condition**: decides whether to proceed with a change. A condition is a pair of state variable and value. The value is checked against the value of the state variable. If they match each other, the associated outgoing change is fired. A condition is represented by a box with the "Var ?= val" sign.

The Petri nets of a presentation may contain a number of selections. Usually, a selection is associated with only a navigation message. However, if the selection is tight to a condition, the selection may have more than one outgoing messages. Which message to send out is decided by the condition's value. We allow an assignment statement to change the value of a state variable. State variables are used in a presentation to hold internal states. These variables will be saved on the disk when the presentation terminates (upon the user's decision), and will be loaded when the presentation starts

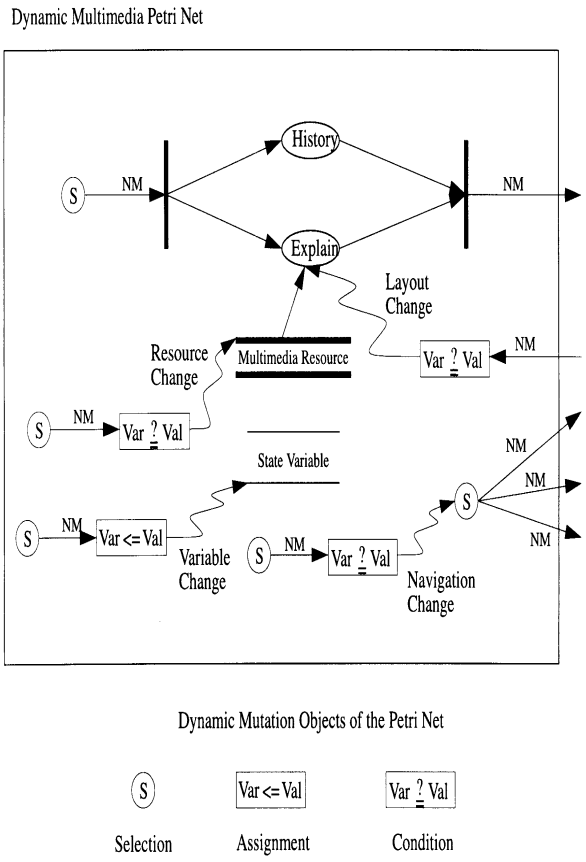

Figure 5: A Dynamic Mutable Multimedia Petri Net

again. A condition decides whether to proceed with a change or the propagation of a navigation message. A condition may change the execution flow of a Petri net.

The above are semantics of the newly introduced components of our Petri net. Since each Petri net place is associated with a multimedia resource, in order to allow the designer to select and reuse a multimedia resource, we have developed a multimedia object database. A resource contains a number of attributes. In the following subsection, we discuss these attributes. In the section of graphical user interface, we will show a multimedia resource browser.

5.1 The Multimedia Resource Attributes

To make a good multimedia presentation, one has to use a set of multimedia resources. Multimedia resources are recorded or captured via camera, tape recorder, or video camera, converted to their digital formats, and saved on the disk. These resource files can be reused in different presentations. A resource is associated with a number of attributes. We consider the following attributes for multimedia resources:

- **name**: a unique name of the resource.
- **keyword**: one or more keywords are used as the description of a multimedia resource. For instance, name of the city is a keyword of the bitmapped picture of Paris.
- **usage**: how the resource may be used (e.g., background, navigation, or focus).
- **medium**: what multimedia device is needed to carry out this resource (e.g., hardware supports for sound, video, MPEG, or picture resources).
- **model**: how the resource is presented (e.g., table, map, chart, or spoken language).
- **temporal endurance**: how long does the resource last in a presentation (e.g., 20 seconds or permanent).
- **synchronization tolerance**: how does a participant feel about the synchronization delay of a resource. For instance, a user usually expects an immediate response after pushing a button for the next page of text. But, the user might be able to tolerate a video playback being delayed for two seconds.
- **detectability**: the intensity of the resource attracting a listener (e.g., high, medium, or low).
- **startup delay**: the time between sending a message and the presentation of the corresponding resource, especially when the resource is on a remote computer connected via network.
- **hardware limitation**: what kind of hardware is essential for carrying out the resource (e.g., MPC level 1, level 2, level 3, or other limitations).
- **version**: the version of this resource file.

- **date/time**: the date and time this resource file was created.
- **resolution**: the resolution of this resource file, specified by $X \times Y$ (or 0×0) screen units.
- **start/end time**: for non-permanent resources, the starting cycle and the ending cycle of the piece of video, sound, or other resource that can be used, especially used as a presentation resource. A cycle can be a second, one-tenth of a second, or a frame number of a video/animation.
- **resource descriptor**: a logical descriptor to a physical resource data segment on the disk.

Since each resource has a number of attributes, it would be cumbersome to require each query searching for a resource to contain all of these attributes. Thus, we propose an intelligent mechanism to simplify a query. The system contains several inference rules. Each rule describes an if-then relation between two attributes. For example, the following are some of the rules used in our system:

```
If usage = focus then
  detectability = high
If model = illustration then
  medium = picture
If medium = picture then
  temporal_endurance = permanent
If medium = MPEG then
  hardware_limitation = MPEG_card
If model = map then
  medium = picture
If ... etc.
```

Some un-specified attributes can be deduced from others. Thus, a user does not need to specify all attributes of a resource when he/she is using a query to search for the resource.

Recently, Data Mining has became a hot research topic in the community of database systems. The existence of mutual dependence among the above multimedia attributes infers us that it is possible to analyze these dependence and use Data Mining techniques to improve our system. For instance, we found that many presentation designers use a bit mapped picture with a low detectability for background usage. A MIDI medium resource is usually used as a background music. We are constructing an interactive database subsystem to collect the ways that presentation designers use our database. Therefore, the subsystem may later on suggests the user to use a good resource.

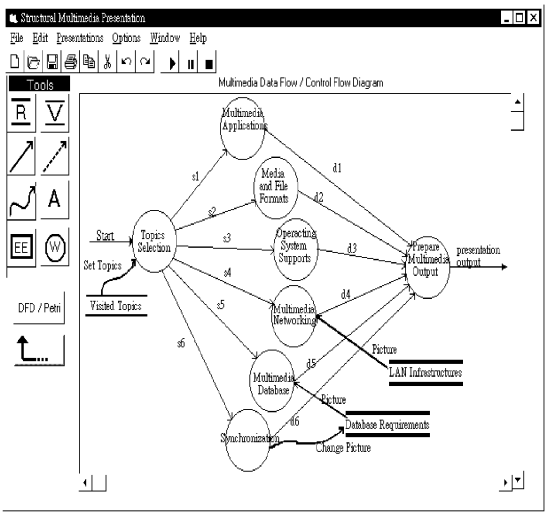

Figure 6: A Structured Multimedia Presentation Design Tool

6 The Multimedia Presentation Design Tool

In this section, we present the graphical user interface of our proposed multimedia presentation system. Figure 6 shows the main window, which has a tool box containing some ICONs. The ICONs in the left column allows the designer to create: Multimedia Resource, Resource Data link, Dynamic Mutation, and External Entity respectively. The ICONs in the right column are to create: State Variable, Navigation Message, text label, and Presentation Window. We also have two push buttons below the tool box allowing the user to choose a DFD/CFD or Petri net diagram, and to navigate to an upper level of the diagram. A presentation designer has to use a drag and drop mechanism, similar to that of a standard windowing system, to create, insert, or delete diagram components. Clicking on a Presentation Window circle brings up the next level refinement of that window. Clicking on other objects results in other design windows. We have text and selection windows allowing the editing of individual type of diagram components. These windows contain text boxes and push buttons which allow the user to enter specific information of each component.

In figure 7, the tool box on the left of the main window is changed for Petri net diagram designs. ICONs in the left column are to create: User Transition, Sync Arc, Transition, Selection, and Condition. ICONs in the right columns are for: Place, User arc, text label, and Assignment. The mechanism to create the diagram is similar to those in figure 6. We also have a navigation message window. One or more messages with optional parameters are entered by the user.

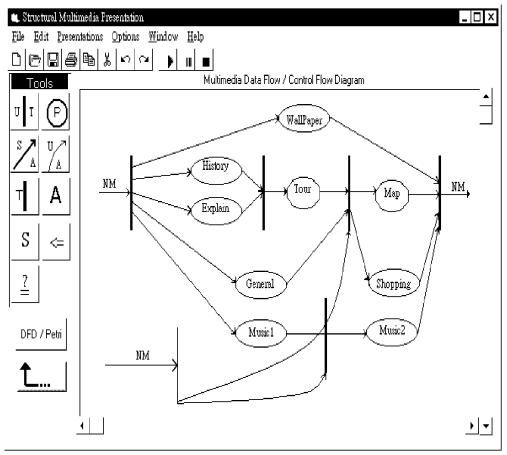

Figure 7: A Multimedia Petri Net Tool

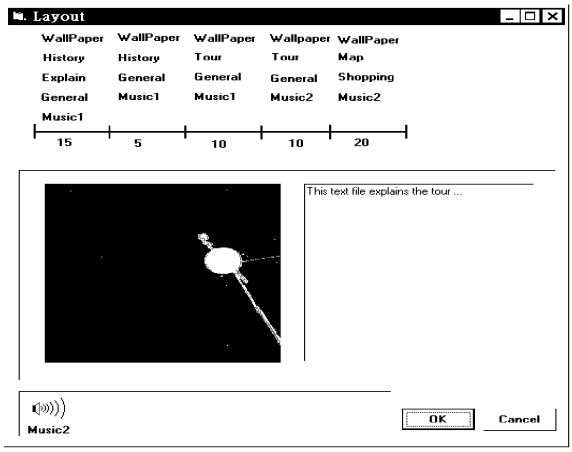

Figure 8: The Layout Design Window

The layout design window shown in figure 8 is a little complicate. The ruler below a number of place names indicate that the presentation is divided into five states. In a multimedia presentation, the start or the end point of a multimedia resource makes a state change. The change of state usually changes the presentation layout. While in a state, the layout is fixed. Clicking on a transition or user transition brings up the presentation layout starting from the transition. For example, clicking on the first transition in figure 7 results in the layout design window in figure 8. Presentation states are deduced from the propagation starting from a transition. Each presentation state is associated with a number of places (above the ruler). Clicking on a section of the ruler (representing the state) allows the designer to alter the location or size of objects of the state. However, Sound and MIDI objects does not have layout. They are displayed as ICONs below the layout area. Each state in the ruler is given a number indicate the number of executing cycles of the state.

In figure 9, we show a multimedia resource browser. The browser is to help the designer to select resources needed in a presentation [8].

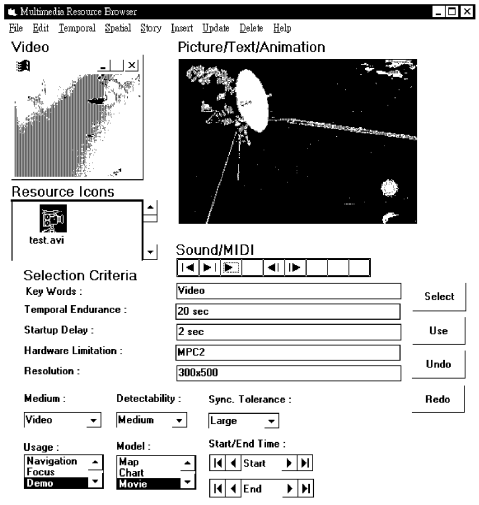

Figure 9: The Multimedia Resource Browser

7 Conclusions

We used Visual Basic and Visual C++ as the implementation languages of our system. The preliminary experiences show that, the proposed multimedia DFD/CFD is easy to learn since data flow diagrams have been used by many engineers and managers for two decades. The concept is easily adapted. Even the implementation is not perfect, we have some contributions in this paper. Firstly, we revised structured analysis/design methodology and data flow/control flow diagrams for the use of multimedia presentation designs. Next, we proposed

a multimedia Petri net for dynamic multimedia presentations. Finally, a prototype system was developed on MS Windows 95/3.1 to justify our approach. With this system, we hope to bring the spirit of structured analysis/design methodology to the design of multimedia presentations. The system can be used for general-purposed presentations as well as education software, demonstrations, and others.

References

[1] H. Khalfallah and A. Karmouch. "An architecture and a data model for integrated multimedia documents and presentational applications". *Multimedia Systems, Springer-Verlag*, 3:238–250, 1995.

[2] T. Kirsta and W. Hubner. "An Open Hypermedia System for Multimedia Applications". In *Eurographic Seminars, Tutorials and Perspectives in Computer Graphics, L. Kjelldahl(ED.) Multimedia Systems,Interaction and Applications Chapter 17*, chapter 17. 1991.

[3] A. Lundeberg, T. Yamamoto, and T. Usuki. "SAL, A Hypermedia Prototype System". In *Eurographic Seminars, Tutorials and Perspectives in Computer Graphics, L. Kjelldahl(ED.) Multimedia Systems,Interaction and Applications Chapter 10*, chapter 10. 1991.

[4] R. S. Pressman. *"Software Engineering: A Practitioner's Approach"*. McGraw Hill, 1992.

[5] G. A. Schloss and M. J. Wynblatt. "Presentation Layer Primitives for the Layered Multimedia Data Model". In *Proceedings of the IEEE 1995 International Conference on Multimedia Computing and Systems, May 15-18, Washington DC*, pages 231–238, 1995.

[6] T. K. Shih. "Multimedia Abstract Machine". In *Proceedings of the Third Joint Conference on Information Science*, 1997.

[7] T. K. Shih and R. E. Davis. "IMMPS: A Multimedia Presentation Design System". *IEEE Multimedia*, Summer, 1997.

[8] T. K. Shih, C.-H. Kuo, and K.-S. An. "An Object-Oriented Database for Intelligent Multimedia Presentations". In *Proceedings of the IEEE International Conference on System, Man, and Cybernetics Information, Intelligence and Systems Conference*, 1996.

[9] T. K. Shih, C.-H. Kuo, and H.-J. Lin. "Visual Multimedia Presentation Designs". In *Proceedings of the 1996 Pacific Workshop on Distributed Multimedia Systems*, pages 306 – 315, 1996.

[10] R. H. Thomas and e. a. Harry C. Forsdick. "Diamond : A Multimedia Message System Built on a Distributed Architecture". *IEEE Computer*, December:65–78, 1985.

[11] M. Vazirgiannis and C. Mourlas. "An Object-Oriented Model for Interactive Multimedia Presentations". *The Computer Journal*, 36(1), 1993.