中華民國八十六年全國計算機會議

瀑布模型之多媒體簡報設計
A Water Fall Approach to Multimedia Presentation Designs

施國琛,江定榮,洪啓舜,蕭鈺恒,蘇溢芳
Timothy K. Shih, Ding-Rong Jiang, Jason C. Hung, Yu-Heng Xiao, and I-Fang Su

淡江大學資訊工程研究所
Dept. of Computer Science and Information Engineering
Tamkang University, Tamsui, Taipei, Taiwan 251,R.O.C
Email : tshih@cs.tku.edu.tw

白文章,王俊嘉
Wen C. Pai, Chun-Chia Wang

光武工專資訊管理系
Department of Information Management
Kuang Wu Institute of Technology and Commerce
Paitou, Taipei, Taiwan 112, R.O.C
Email : ccwang@mine.tku.edu.tw

摘要

在這篇論文中，我們提供了一個基於軟體工程中瀑布模型的系統，並允許多媒體簡報使用資料/控制流程圖來設計，我們也使用了修改過的具有一些新元件的派翠網路，來讓設計者基於展示物件的同步做控制，我們將描述那些新引入符記所代表的意義，我們也將會討論實作我們這套系統的演算法。

關鍵字:多媒體簡報,瀑布模式,逐步求精,資料流程圖,結構化分析,派翠網路,軟體工程,多媒體虛擬機器

Abstract

In this paper, we propose a system based on the Water Fall paradigm in Software Engineering to allow multimedia presentation designs using data flow/control flow diagrams. We also use a revised Petri net, with a number of newly introduced components, to allow the designers to control the event based synchronization of presentation objects. Semantics of the newly introduced notations are described. Algorithms for implementing our system are also discussed.

Keywords : Multimedia Presentation, Water Fall Paradigm, Stepwise Refinement, Data Flow Diagram, Structured Analysis, Petri Net, Software Engineering, Multimedia Abstract Machine*

1. Introduction

Structured analysis and design have been used in software development. This Water Fall paradigm [1] 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 comparison of Water Fall paradigm and presentation development are as the following :

Comparison of Water Fall Paradigm and Presentation Development

Water Fall Paradigm	Presentation Development Paradigm
Analysis	Presentation Requirements Analysis
Specification	Multimedia Presentation DFD/CFD
Design	Interactive Multimedia Petri Net
Implementation	Automatic Presentation Generation
Testing	Automatic Presentation Testing
Maintenance	Presentation Documentation to Ease Maintenance

This paper is organized as follows. We present the life cycle of multimedia presentations in section 2. The analysis of presentation requirements is discussed in section 2.1. Presentation specification and the proposed multimedia data flow/control flow diagram is discussed in section 2.2. An interactive Petri net program serves as the presentation design assistant tool is proposed in section 2.3. Section 2.3.1, as an extension to the interactive Petri net, proposes a new direction of multimedia presentation designs. Unlike other systems, which generates static presentations, our system allows dynamic multimedia presentations. Semantics of the newly introduced objects are also discussed. Multimedia presentation implementation, testing, and maintenance are discussed in section 2.4 and section 2.5. The implementation of our presentation system is discussed in section 3. 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 3.1. The algorithms of our Petri net machine are discussed in section 3.2. A short conclusion summarizes our contributions is also presented in

* This project is supported by the NSC grant # NSC 87-2213-E-032-019 of the Republic of China.

section 4.

2. Presentation Life Cycle

In this section, we discuss these phases of presentation life cycle in detail.

2.1 Analysis of Presentation Requirements

The analysis of presentation requirements is the most important factor achieving the success of a multimedia presentation. How to attract the audience and demonstrate the main theme of a presentation through text, sound, and graphic illustrations are the purposes of presentation analysis. The next step of presentation development is to write a specification of the presentation which is discussed in section 2.2.

2.2 Presentation Specification

We propose a revised DFD/CFD mechanism for producing multimedia presentation specifications. 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:

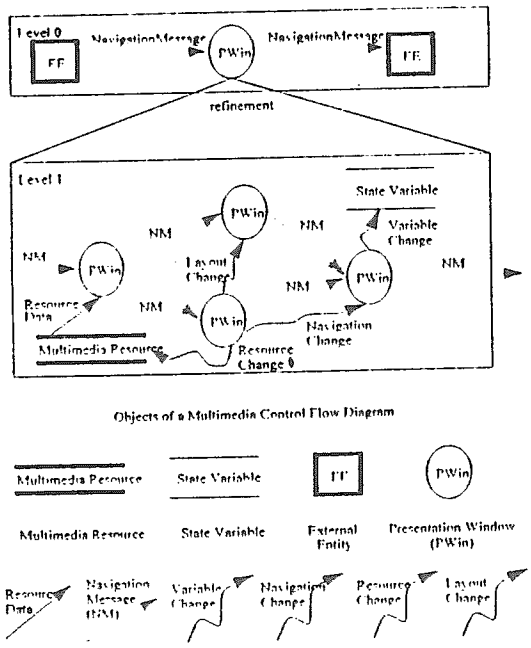

Figure 1 : Multimedia DFD/CFD

- **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 2.3.1).
- **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
  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.

Figure 1 illustrates the various types of components of a multimedia DFD/CFD.

2.3 Presentation Design

A presentation window must be refined. Our 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 [4]. The multimedia database resource browser [4] allows the user to select multimedia resources used in a presentation.

An example showing the last level refinement is shown in figure 2. The spatial organization of the presentation is specified by using the graphical user interface of our system. Our interactive multimedia Petri net is a variation of timed Petri net, with the addition of User Transitions and Sync Arc. For instance, in figure 2, 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 refinement between a presentation window and its child DFD/CFD or Petri net needs to maintain two types of balances: the navigation message/dynamic mutation balance and the multimedia resource/state variable balance. 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 ellipses.
- **User Arc** : connects from a user transition to a transition. User arcs are shown as curved arrowheaded arcs.
- **Sync Arc** : connects from a place to a transition, or from a transition to a place. Sync arcs are shown as straight arrowheaded lines.

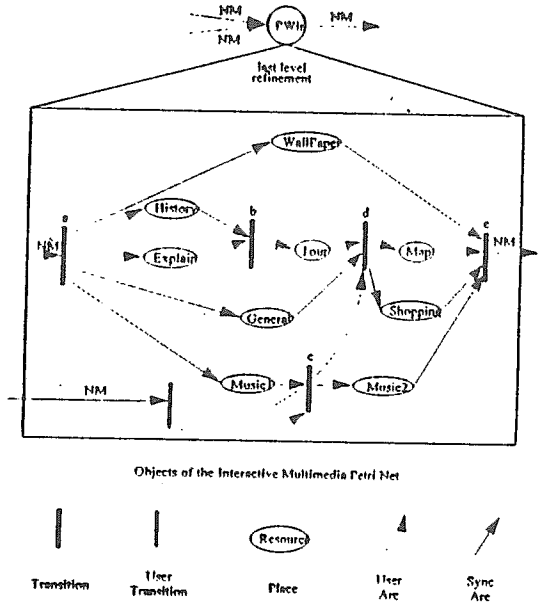

Figure 2 : An Interactive Multimedia Petri Net

2.3.1 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. 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 3 illustrates components of a mutable Petri net. On the top of the revised timed Petri net (see figure 2), we further add the following components:

- **Selection** : selects one of the outgoing navigation messages. A selection could be a push button, a menu item, or a key stroke of the presentation window. A selection holds an internal representation of which outgoing navigation messages will be sent upon the activation of the selection. A selection is represented as a circle labeled with an "S" (or a name) in the Petri net diagram.
- **Assignment** : assigns a value to a state variable. An assignment uses 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 has pairs of state variables and values. The value is checked against 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.

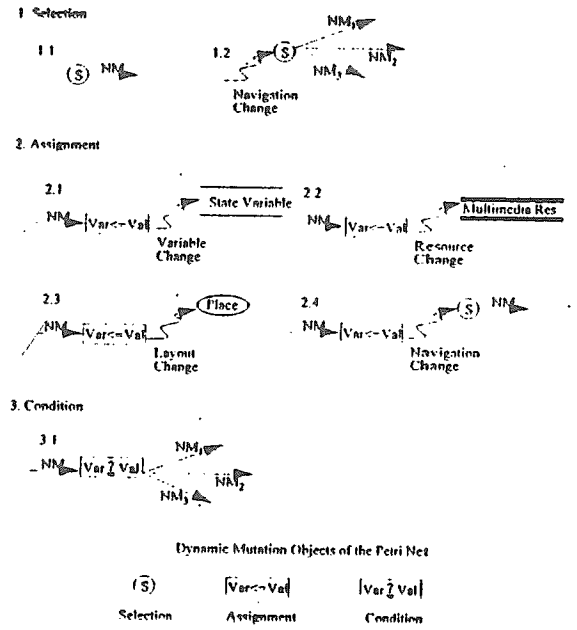

Figure 3 : Components of A Dynamic Mutable Multimedia Petri Net

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 again. All state variables are global. That is, they are accessible from all Petri nets. Therefore, information sharing is feasible. A condition decides whether to proceed with a change to the propagation of a navigation message. A condition may change the execution flow of a Petri net. In figure 3, we have introduced three new Petri net objects (i.e., selection, assignment, and condition). The semantics of these objects are defined as the following:

Selection

Item : 1.1
Input : none
Output : NM
Semantics : when the selection is activated, the NM is sent.

Item : 1.2
Input : *NavigationChange*
Output : NM_1, NM_2, ..., NM_n
Semantics : upon receiving the *NavigationChange*, the NMi to be sent by the selection is changed according to the *NavigationChange*. Note that, a *NavigationChange* does not cause a message sending. However, when the selection is activated at the next time, the navigation sequence will be different. All possible navigation sequences are re- defined via the GUI. That is, all NM_i's are drawn before the presentation proceeds. No new NM_i is created when a presentation is running.

Assignment

Item : 2.1
Input : NM
Output : *VariableChange*
Semantics : when NM is received, the *StateVariable* is set to a value specified in the NM.

Item : 2.2
Input : NM
Output : *ResourceChange*
Semantics : when NM is received, the *MultimediaResource* used is set to a resource specified in the NM.

Item : 2.3
Input : NM
Output : *LayoutChange*
Semantics : when NM is received, the layout of a resource is set to the one specified in the NM.

Item : 2.4
Input : NM
Output : *NavigationChange*
Semantics : when NM is received, the NM sent by the selection adjacent from this assignment is changed according to the one specified in the incoming NM to the assignment.

Condition

Item : 3.1
Input : NM
Output : NM_1, NM_2, ..., NM_n
Semantics : when NM is received, a NM_i, $1 \le i \le n$, is sent.

2.4 Presentation Implementation and Testing

It is possible to obtain the software metrics of a presentation. Our system is able to calculate the metrics of a presentation based on the following criteria:

- number of presentation windows
- number and size of multimedia resources
- number of navigation messages
- number of state variables
- number of dynamic mutations
- number of selections
- number of assignments
- number of conditions

Software metrics of a presentation indicates the complexity and the amount of efforts to test the presentation. Our system facilitates automatic testing by means of an tool. The tool traverses the presentation based on the following testing criteria:

- every presentation window should be tested
- every multimedia resource should be tested
- every navigation message should be tested
- every selection should be tested
- every assignment should be tested
- every condition should be tested

The testing includes all Petri nets, which may accept navigation messages sent to transitions or user transitions. Synchronization of multimedia resources used by places are tested as well. Moreover, selections, assignments, and conditions are tested. Our testing tool traverses a graph consists of nodes (i.e., selections, assignments, conditions, and starting transitions/user transitions) and links (i.e., navigation messages) based on a modified breadth first search (BFS) algorithm. Another feature of our system is an evaluation tool of the presentation design status. It is very likely that a user design an incomplete or inconsistent presentation. The following is a list of potential mistakes that a user may create in a presentation:

- non-used state variables
- non-used multimedia resources
- non-used assignments
- non-used conditions
- incomplete navigation messages
- incomplete selections
- incomplete assignments
- incomplete conditions
- incomplete presentation windows (i.e., if any component of the window is incomplete)
- inconsistent presentation windows (i.e., unbalanced links)
- undocumented presentation windows

Non-used items, such as state variables not used in any condition, multimedia resources not linked to places, or assignments and conditions not triggered by navigation messages, are marked by the tool. Incomplete items, with one of their properties missing, are also marked. A presentation window is inconsistent if the window and its refinement have different links (i.e., the unbalanced link problem). An inconsistent or undocumented presentation window is also marked. The user is able to check the list of problematic items before releasing the presentation.

2.5 Presentation Maintenance

Presentation maintenance is an important issue. A presentation specification in our system contains a number of presentation windows. Each presentation window has a documentation box. It is important to document the presentation script in the box for further references. Moreover, how to keep track of different versions of a presentation window is important. In a multimedia presentation database we developed [3,4], presentation windows as well as presentation resources are reusable objects. Each of these objects has several versions. A presentation is a composed object from these reusable objects. The database system serves as an underlying supporting tool for several multimedia presentation systems that we have developed [3,5],

including the one we propose in this paper.

3. Implementation of the Presentation System

We use Microsoft Visual C++(VC) and Visual Basic (VB) to implement our system. The graphical user interface is implemented in VB and the presentation engine is implemented in VC with the supports of Media Control Interface (MCI) functions of Windows 95. In this section, we discuss our system from the overview perspective -- a multimedia abstract machine (MAM). We also present algorithms of the multimedia Petri net.

3.1 A Multimedia Abstract Machine

The main theme of our system is a timed Petri Net based multimedia abstract machine (MAM). MAM is a software architecture and system for multimedia documentation demonstrations and presentations. We have the MAM instruction set defined [2]. 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.

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.

3.2 Petri Net Algorithms

To run a presentation, its schedule must be decided first. Considering the Petri net with places and transitions shown in figure 2, we want to construct a transition graph from the Petri net. In the transition graph, the weighted edges represent places with durations, and nodes represent transitions. From this transition graph, the schedule of presentation can be computed. Figure 4 shows such a graph corresponding to the Petri net shown in figure 2, starting from transition "a". Note that, in constructing the transition graph, a user arc has a weight equals to zero since a user transition causes the immediate firing of the corresponding linked transitions.

From the transition graph, the algorithm initials the node pointed by the message (i.e., node "a") to zero. And, the nodes of the graph are sorted according to their topological order. Finally, the node time of each node is set to the maximum estimated time of all incoming edges of the node. An estimated time is computed from the node time of the source node and the duration of a multimedia resource. Algorithm **Compute_Node_Time** summarizes the above computation.

```
Algorithm: Compute_Node_Time
  Construct a Transition_Graph reachable from one of the
  starting Navigation_Messages;
  Initialize the node pointed by the Navigation_Message
  to Node_Time = 0;
  Sort the Transition_Graph to a Transition_List by
  Topological sort;
  For each node N in the Transition_List, except the first,
    do
    For each incoming edge, compute the Estimated_Time
                    = Node_Time + Edge_Duration;
    Set Node_Time of node N to the maximum of all
    Estimated_Time of incoming edges;
```

From the transition shown in figure 4, the algorithm computes a transition list with node time of each transition:

Transition_List:

```
Node_Time(a) = 0
Node_Time(b) = max{0+20, 0+15} = 20
Node_Time(d) = max{20+20, 0+40) = 40
Node_Time(c) = max{0+30} = 30
Node_Time(e) = max{0+70, 40+15, 40+20, 30+35) = 70
```

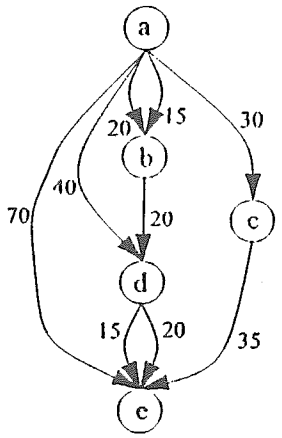

Figure 4 : A Transition Graph Reachable from a Navigation Message

The node time in the transition list above indicates the clock cycle which a number of resources are to be demonstrated. The set of resources to be played with respect to a transition is collected from the outgoing sync arcs of the transition. For instance, the following sync arc sets are computed for the five transitions:

Sync_Arc_Sets:

```
Sync_Arc_Set(a) = { WallPaper, History, Explain,
                    General, Music1 }
Sync_Arc_Set(b) = { Tour }
Sync_Arc_Set(c) = { Music2 }
Sync_Arc_Set(d) = { Map, Shopping }
Sync_Arc_Set(e) = { }
```

With the node time and sync arc sets, we know when

and what resources to play, as shown in the presentation schedule chart in figure 5.

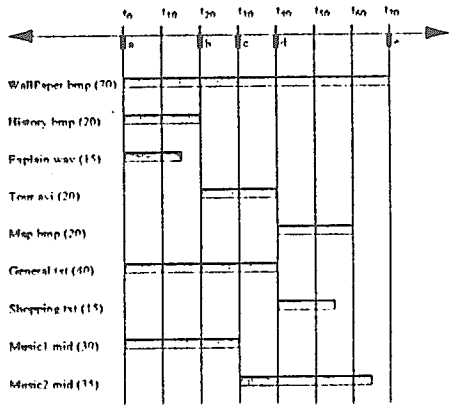

Figure 5 : A Presentation Schedule

The presentation engine is a concurrent program, which is simulated by the multithread/multiprocess facility of Windows 95. The first process plays resources in the sorted transition list. The second process takes cares of interrupt events, such as the receiving of another navigation message, as shown in algorithm **Run_Petri_Net**.

```
Algorithm: Run_Petri_Net
For each node N in the Transition_List do
 Collect Places pointed by Sync_Arcs from node N in
 Sync_Set(N);
 Sort the Transition_List according to Node_Time in a
 non-decreasing order;
CoBegin
  Process1:
    For each node N in the Transition_List do
      sequentially Play resources in Sync_Set(N) at
      Node_Time(N) concurrently;
  Process2:
    LoopBegin
      If Process1 ends, then Process2 ends;
If there is a Navigation_Message of a
  User_Transition
then Exits with the next starting
      Navigation_Message;
If there is a Selection, Assignment, or Condition
then Perform the action accordingly;
    LoopEnd
Coend
```

Finally, algorithm **Run_Presentation** uses the above algorithms and serves as a presentation engine.

```
Algorithm: Run_Presentation
LoopBegin
 If there is a starting Navigation_Message then
    Compute_Node_Time;
    Run_Petri_Net;
 else
   End Presentation
LoopEnd
```

4. Conclusions

The preliminary experiences of using the system show that, the proposed multimedia DFD/CFD is easy to learn since data flow diagrams have been used by many engineers and managers for decades. The concept is adapted easily. Comments from many undergraduate students taking a multimedia course show that it is quite easy to use our system. Contributions in this paper are: firstly, we applied the Water Fall paradigm to presentation development and revised data flow/control flow diagrams for the use of multimedia presentations. Next, we proposed a multimedia Petri net for dynamic multimedia presentations. Finally, a system was developed on MS Windows 95/NT 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] R. S. Pressman. ``Software Engineering: A Practitioner's Approach". McGraw-Hill,1992.

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

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

[4] 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.

[5] 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.