Int. J. Engng Ed. Vol. 12, No. 4, p. 257–263, 1996
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 1996 TEMPUS Publications.

Effective Courseware Development

JANA DOSPISIL
LIZ KENDALL
ANDREW JENNINGS
Royal Melbourne Institute of Technology, Department of Computer Systems, Melbourne 3001, *Australia*

The use of hypermedia and multimedia technology is perceived as one of the most efficient ways to enhance student access to information and improve learning interaction in the undergraduate computer science course. Such an approach will guarantee high-level quality teaching whilst permitting significant reduction in lecturer-student contact hours. In this paper we investigate how some special properties of multimedia data can be handled by conceptual modelling techniques. We propose a simplified conceptual framework for the development of interactive courseware. In our proposal, we examine firstly the static structure of the system focusing on the structure of objects and their relationship to each other while temporarily disregarding the dynamic nature of the system. To simplify the conceptual model construction we use predefined constructs and topic descriptors. In a later stage the conceptual model is used to generate the functional prototype. The dynamic behaviour of the system—flow of control, interactions, sequencing of operations on active objects—is examined in real time using the functional prototype, corrective member, and the feedback loop.

1. MOTIVATION

TECHNOLOGY-based learning is one of the 'hottest' disciplines not only in educational research but also in computer science research. Computers and electronic networks are becoming an inseparable portion of our education systems. The main component of the enabling technology has its roots in distributed computing. In higher education institutions, distributed computing was applied and gained a wide acceptance predominantly in computer-oriented disciplines and in computer science departments with well-established research, development facilities and interests. The major progress so far can be observed at MIT (Project ATHENA [1]), Brown University (Project INTERMEDIA [2]), and Carnegie Mellon University (Project Andrew [3,4]).

There is a significant difference between the users of multimedia in the entertainment industry and university students. Three aspects where the entertainment industry development differs from courseware development include:

- Market size. If a game is developed and CD-ROM produced, there is potentially a very large audience available to recover the development cost and make some profit. With courseware, even if we take into account worldwide university audiences, the total potential audience is still quite small.
- Product maintenance. In the entertainment industry, a final product is not expected to be maintained on a regular basis. It may be improved and a new version released but no ongoing maintenance is planned. Any courseware product, on the contrary, is a subject of constant evolution as a result of changes in a syllabus or the need to incorporate students' latest courseware evaluation results.
- Financial position of potential buyers. Typically, the entertainment products are developed by vendors with solid commercial development experience, priced accordingly, and sold to public in large quantities. The courseware development is financed by the educational institutions with very little experience in development of commercial projects. To appeal to students' community, the products must be inexpensive.

In summary, money is important aspect in the development of multimedia products. This factor will not change much soon. To produce multimedia-based courseware within university budget, we have to cut cost of production and introduce well-defined software engineering practices [5].

The process that plays the critical role in such a development environment is the concept acquisition and requirements specification, collectively called requirements engineering. As we will discuss later, the hypermedia design and programming development environment show some peculiarities that make them different from the commercial development processes. In this case study, we assume a typical interactive multimedia development process in a teaching environment. The objective is to simplify the methodology for designing a courseware conceptual model.

2. MULTIMEDIA KEY CHARACTERISTICS

Definitions

Multimedia is a system that combines diverse media in a coherent system. Multimedia artefacts

* Accepted July 5, 1996

are presented to the user as continuous stream of atomic elements (e.g. motion video is a sequence of frames). A key element of this activity is interactivity [6]. *Hypermedia* provides a mechanism to store and retrieve unstructured, related information in a meaningful way [7]. It is an information handling model in which separate units (objects) of information are linked in a structured network.

Time-dependent multimedia data

An important aspect of multimedia information is its *temporal nature*. It consists of sequences of atomic units that have to be manipulated with a predefined data rate. For example, motion video is a sequence of frames which are played at a data rate of 30 frames/s (NTSC format of playback). A composite media object may then involve elaborate synchronization of audio and motion video frames and sequences. The tolerance of data to time synchronization during playback varies widely. Audio and video data are tightly bound with a precision of hundreds of milliseconds. Text and associated image will tolerate seconds. To capture unambiguously the user requirements involving temporal transformations and object interdependencies of media composite objects and incorporate internal medium dependencies is a very complex task. In addition, the tools and methodologies that would support modelling of the temporal media relationship are still in the research stage [8,9] and therefore commercially unavailable.

Composition mechanism

In multimedia applications, the *composition mechanism* and *presentation time* play crucial roles. The nature of the multimedia assets enforces a perception of a set of autonomous units of motion video, audio or graphical information. These units must be combined in an integrated system that conveys information to the user with minimum redundancy. Such an integration is typically constrained for example by the hardware platform, the properties of the authoring tools, or an access to shared resources (e.g. the number of video channels is physically limited, or Digital Video Interactive™ (DVI) supports only nine simultaneous stills). Again, these constraints are difficult to capture and model with current modelling and programming tools.

Interactive courseware

The main feature of multimedia and hypermedia systems is their interactivity, which means that the user is controlling the object behaviour [10]. Interactivity requires software control capabilities. The authoring tools currently available (e.g. AVC™ or MediaScript for OS/2) lack maturity and provide only limited interactivity features. For instance, the lowest level of interactivity is typically perceived as the usage of the '*stop*' points (sometimes termed '*one button*' interface) in which the user is required to select the next topic (branching). On the other hand, a high level of interactivity may be perceived as the capability to browse and annotate any media object and manipulate embedded objects (providing the user/author possesses a required authorization level).

This high level of interaction is so far associated with a number of technological difficulties. For example, in the OS/2-based environment, cut/paste functions applied to as portion of the digitized motion video frame require the designers to develop a special 'video editor' process. The other example, embedded video object manipulation, is typically resolved by deploying the Dynamic Data Exchange mechanism (DDE) in Windows and Presentation Manager which does not provide true embedding. If a user requires the capability of clicking on the digital video embedded object and editing, then DDE does not solve the problem and a more sophisticated approach must be considered. This is possible only with Andrew Toolkit [11].

Shared resources

Multimedia objects are typically modelled as 'active objects' [12,13]. Using active objects always involves concurrency problems [14,15]. A general criterion for the design of a concurrent mechanism is that it should support many different forms of concurrency: shared memory, multitasking, physical object distribution, distributed processing, etc. It cannot be expected that there will be a language that will be capable of supporting all the applications. It seems that a mechanism that is external to the language will have to be devised. Such a mechanism then will be able to provide a rich set of constructs and mappings from the abstract threads of control associated with active objects to physically available resources and processes [15].

Hypermedia node-link modelling

Each presentation segment in the hypermedia system could show interrelationship with the other segments in the structure. Too few links typically indicate that the structure is inappropriate; too many links may be distracting to the learning [16]. For groups of different learners with different levels of experience with computers, this requirement will remain unclear and will be difficult to specify regardless of the specification notation. Furthermore, efficient link management is still in the research stage [11,16].

3. CURRENT APPROACHES TO MODELLING MULTIMEDIA TIME-DEPENDENT OBJECTS

As mentioned above, continuous media such as digital video, digital audio, or animation introduce a few different aspects into data modelling technologies that are foreign to conventional data models. The aspects that differentiate multimedia streams from the conventional data include storage and retrieval of large sequences of binary data, synchronization of the external and internal media data streams, and resource sharing with long processing times. The main research focus is on two processing fields: functionality of the database engine for multimedia, and presentation logic in the presence of real-time constraints (-dynamic modelling). In this section, we attempt to provide a brief overview of the current approaches to continuous media modelling, in particular synchronization of time-dependent multimedia.

The object-oriented approach taken in [12,17] is based on active objects (ActiveObject) which produce/consume multimedia values. Multimedia are viewed as a collection of ports. A port is determined by its data type (CDAudio, NTSC, etc.) and usage type (input or output). Multimedia objects are divided into three categories: source, sink, and filter. The multimedia object which cannot be decomposed forms a multimedia primitive. Composite objects are then a collection of spatially ordered multimedia primitives. The ActiveObject provides basic control activities for a multimedia object. The multimedia object, besides methods and parameters inherited from the class ActiveObject, introduces special constructs that support Timed Streams. (A Timed Stream is a finite sequence of tuples of the form <element, start time, duration> [17].)

The methods of multimedia objects can be of the following three types: temporal transformations, composition, and synchronization. Temporal transformations contain a group of methods for conversion between the two coordinate systems. Temporal composition focuses on the temporal relationships between two or more multimedia primitives. Temporal relationship determines the synchronization and temporal sequencing of components. Configurational relationship indicates the connections between the input and output ports of components.

The derivation concept in data modelling [17] refers to a technique of computing the value of a specific attribute from the values currently available in the other attributes or items. The derivation D of a media object o_1 from a set of media objects O is mapping of the form $D(O,P_D) \rightarrow o_1$, where P_D is the set of parameters specific to D. o_1 is called the derived object [17]. Derivations can be applied to the content, timing, or media type. The main motivation for this concept is to achieve the capability of representing objects whose parameters are calculated in real-time.

The problem of preserving the timing relationship between media and the process of synchronization was addressed by Little in his proposed conceptual model for a multimedia object [8]. He proposes an optimistic interval-based process approach. The binary and n-ary temporal relations can be precisely derived from start time, duration, and delay from the beginning of the first interval. The model enables establishing the exact playback times for each media component that are necessary for real-time scheduling of retrieval. The database temporal model is then based on two types of nodes. The first type—terminal node—uses attributes such as media type and pointer to the storage to identify the physical aspects of the media artefact. The other type of nodes—non-terminal nodes—hold temporal information (duration, forward delay), node type and pointers to children. The model can be applied to a relational data structure and thus the relational model can be used to maintain temporal relations among multimedia data.

The Object Model for multimedia programming presented in [13] deals with the problem of synchronization of active multimedia objects. In this model, all objects are active objects with their own thread of control. The active objects exchange synchronous messages. The internal control over the message acceptance is managed by a caller using message sampling or message queuing techniques. Synchronization of different media is considered with the relationship of reference points in each media (reference points may be video frame, audio sequences, etc.). The synchronization among media objects is modelled using the event synchronization mechanism for concurrent processes. In addition, the concept of delegation [15] replaces the inheritance and class mechanism. Objects here are viewed as prototypes that delegate their behaviour to related objects thus eliminating the need for classes.

4. THE PROPOSED FRAMEWORK

The current approaches to multimedia conceptual modelling have one common denominator: they are somewhat restrictive in their capability to capture the system as a whole. They produce a well-defined mechanism for programming constructs that will capture time dependencies among multiple multimedia objects, but they lack a means for conceptual modelling of time dependencies as well as other relationships. In particular, the fault-tolerant behaviour of the system and management of shared resources are addressed insufficiently. We believe that it is necessary to create a higher-level abstraction which would promote a simplified modelling approach capable of handling of a methodology paradigm shift from the classical approaches of either object-oriented or information engineering to a creative type of design that is capable of seamless incorporation and management of original artwork within a software engineering environment.

We define a *hypothetical hypermedia system*. The system is composed of a number of aggregate topics that form composite objects (termed also pages)—see Fig. 1. The elementary objects within a page are connected by the static links. Each page may contain different media elementary objects

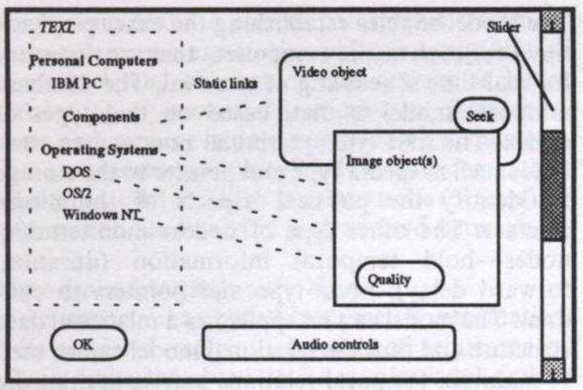

Fig. 1. Aggregate topic composition (page).

and several user interaction controls (buttons, sliders, etc.). Each elementary object *may be* associated with its own thread of control.

Each aggregate topic represents a 'template' (conceptual class definition) for a group of elementary objects with the same properties and strong logical relations. Aggregate topics then form a conceptual domain (also termed information segments). Each aggregate topic *must be* associated with its own thread of control. Links within the conceptual domain can be of both types—static or dynamic (Fig. 2).

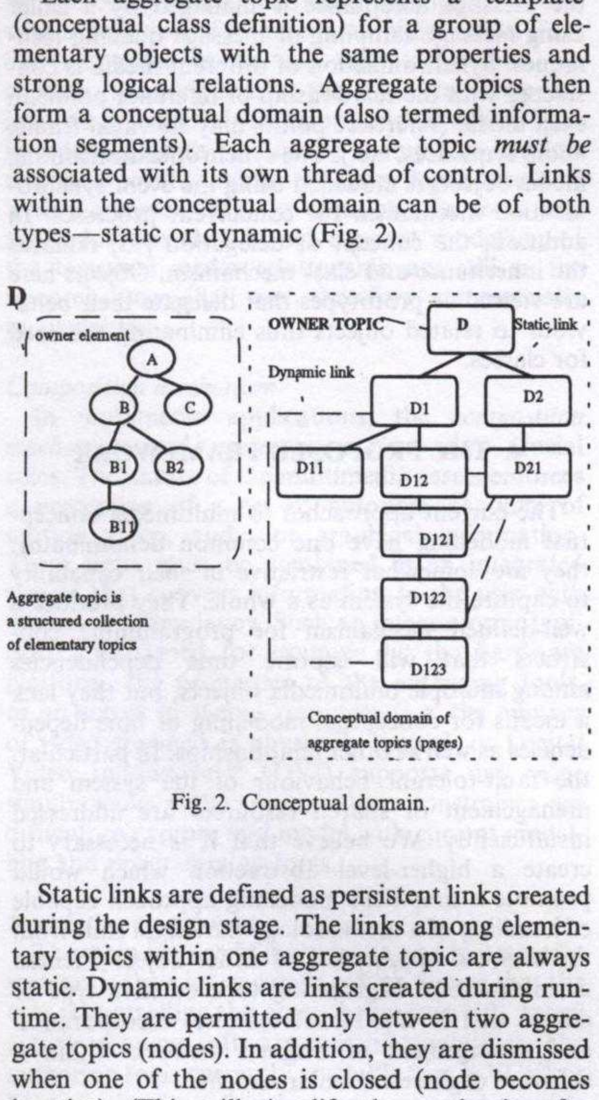
D
owner element
A
B
C
B1
B2
B11
Aggregate topic is a structured collection of elementary topics
OWNER TOPIC
D
Static link
Dynamic link
D1
D2
D11
D12
D21
D121
D122
D123
Conceptual domain of aggregate topics (pages)

Fig. 2. Conceptual domain.

Static links are defined as persistent links created during the design stage. The links among elementary topics within one aggregate topic are always static. Dynamic links are links created during runtime. They are permitted only between two aggregate topics (nodes). In addition, they are dismissed when one of the nodes is closed (node becomes inactive). (This will simplify the mechanism for freeing some shared resources.)

4.1 *Modified development lifecycle*

We believe that in the hypermedia development project the prototyping model must offer a consistent method to:

- gain experience with a particular class of application;
- eliminate the negative impact of a particular mental model people tend to use when approaching a problem; and
- solicit and clarify user requirements.

The majority of people who are developing multimedia software are not computer professionals and have no great desire to become computer professionals but want to use multimedia in their applications. In addition, as our evaluations of a number of authoring tools have indicated, they tend to a more radical approach, in particular to the recursive/parallel lifecycle [18], in the development process of the hypermedia projects as dictated by the restricted power of the authoring software packages and by the technology itself. This enforces the need to solicit an approval to a large number of the user requirements by showing him/her partial results.

The principle of our modified development lifecycle (see Fig. 3) is the feedback loop and the corrective member that combines the input from the user with the feedback on system dynamic behaviour and produces a new version of the system requirements specification (SRS). The role of the corrective member is to evaluate the behaviour feedback and the user input, and to distribute the request for change to the appropriate development class (navigation patterns, assets creation, presentation logic). In each of these classes the formal specification of the requirements is produced and later used to generate the functional prototype for testing. The system *derived model* is produced from the functional prototype and used as the basis for implementation.

4.2 *Properties of the conceptual model*

Our approach is based on defining the structural aspects and semantic relationships of the domain concepts that are now transformed into classes. We have observed that the semantic relationships typically follow two structural patterns: topics

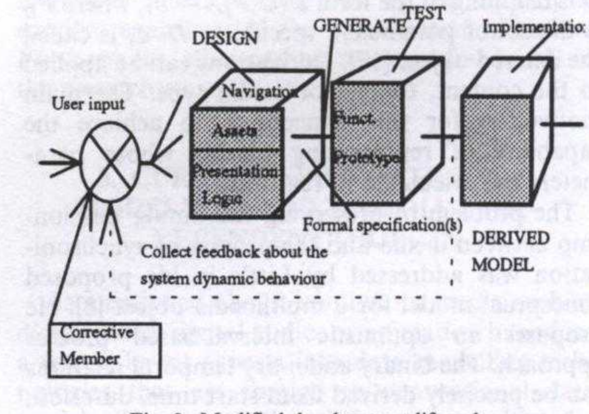

Fig. 3. Modified development lifecycle.

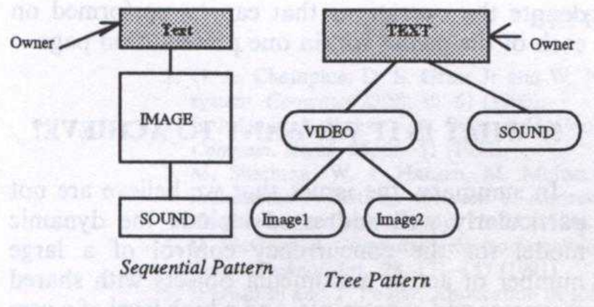

Fig. 4. Sequential and tree structures.

are linked in a sequential, or hierarchical manner (see Fig. 4). The two main components of the conceptual class are the class content and the class structural pattern. Similarly, the two main components of a conceptual domain are a collection of aggregate topics and a domain structural pattern. This approach allows for the explicit definition of two categories of superclasses: sequential pattern based classes and aggregate classes with hierarchically linked subclasses.

The framework for specifying inter-topic dependencies and navigation patterns is based on two aspects: a formal model of the inter-topic dependency specification (Topic Descriptor, TD) and requirements specification language(s) for providing a description of each component.

The TD is a construct similar to a class descriptor [14]: $TD = [O, C, P, R, B]$ where:

- O (*owner*) is the pointer to the owner element in an aggregate topic. The owner element also acts as a reference topic (origin) for establishing static links among the aggregate topics.
- C (*child*) is a dependent topic(s) element within an aggregate topic.
- P denotes the structural pattern within the aggregate class (static links pattern).
- R is a Boolean-valued predicate which specifies the relationship between subclass and origin. It evaluates to TRUE if all the relationship conditions are satisfied. A FALSE result indicates exceptions in navigation, presentation patterns, resource dependency constraints, or the presence of temporal dependencies.
- B is a collection of presentation patterns.

P, R, and B are three main components of a Topic Descriptor (see Fig. 5). The Topic Descriptors form an Intertopic Schema (IS). In other words, all IS is a collection of all Topic Descriptors to be enforced in the hypermedia system for managing navigation and topic presentation within a page. This approach enables the design of the aggregate classes (which in turn depict the aggregate topics).

Aggregate topics (conceptual classes)

Conceptual classes enable the designer to create complex concept domains through an assembly of different structural patterns. These concept domains can be either homogeneous or heterogeneous. The homogeneous concept domains represent a group of topics that are logically related. The inclusion of the heterogeneous concept domains enables the designer to 'walk across' to the logically distant topics. The structural pattern that connects conceptual domains is typically a tree. Each aggregate class *must* have an owner element (page root) and each concept domain *must* have an owner aggregate topic.

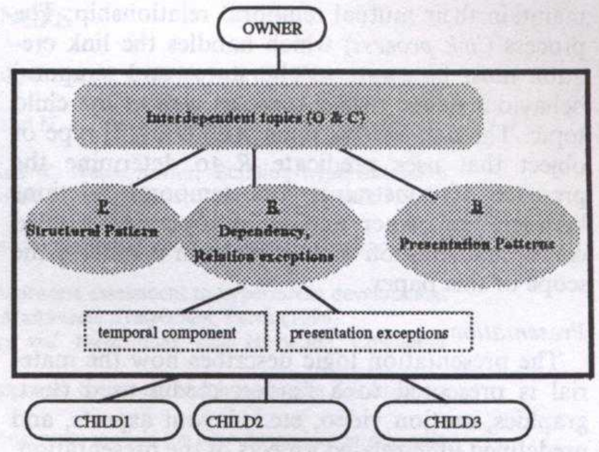

Fig. 5. Major components of a Topic Descriptor.

Static structural primitives (structural superclass)

Segments of information within a concept domain are connected via connection primitives (links), which are defined by structural primitives. Since we are dealing with two predictable structural patterns (tree and sequence of elements), we can define a set of connection primitives within the class 'connections' and use multiple inheritance principles to incorporate the structural patterns into a class. This approach of using regular connection structures will reduce the designer's effort needed to specify static links. At the conceptual level, the designer has to identify the owner–child topic relationship and use the predefined link pattern to connect the owner topic to the child topic(s). The expression may be as follows:

Link ⟨*link_name*⟩ : = = ⟨*owner*⟩ ⟨*direction*⟩ ⟨*child*⟩ | ⟨*owner*⟩ *CURRENT* ⟨*point-within-the-aggregate-class*⟩

The *direction* in the first part of the expression denotes the movement type within the aggregate class using static links. The second part of the expression represents a dynamically created link within the conceptual domain (an interleaf jump dynamically created between two aggregate classes). At this point, dynamic links to elements outside the conceptual domain are not permissible.

The issues associated with this concept are more complex. Dynamic link creation or static link invocation is associated with concurrency and synchronization among elementary or aggregate object instances. For example, video and audio can be played simultaneously and they have to maintain their mutual temporal relationship. The process (*link process*) which handles the link creation must be aware of the status and temporal behaviour of the owner topic as well as the child topic. The *link process* is an *ACTOR* [14] type of object that uses predicate R to determine the presence of constraints and temporal relations between the owner and child topic. A detailed conceptual solution to this problem is outside the scope of this paper.

Presentation

The presentation logic describes how the material is presented to a reader: media used (text, graphics, motion video, etc.), layout aspects, and predefined time-related aspects of the presentation. The presentation logic primitives form the presentation superclass. An atomic unit of presented information is a page. The designer can combine a number of presentation media into one presentation page. For example, a presentation page can contain text, still images and graphics (see Fig. 1). An instance of the aggregate topic (conceptual class) can use a number of presentation pages to display information but it is always associated with one default presentation page (composition template). A simple page description is defined below:

Page⟨page_name⟩ := = ⟨topic identifier⟩⟨composition_template⟩ | ⟨topic_identifier⟩{⟨composition_descriptor⟩}$_n$

The composition descriptor is a composite structure that uniquely identifies the attributes of the media, its coordinates on the screen, predicted temporal properties (motion video playback rate) and behaviour.

Information segment manipulation methods

The information segment manipulation methods are formed by a group of the manipulation primitives. The manipulation primitives are derived for each elementary object or media type: text can be copied, cut, or pasted; motion video can be started, stopped, resumed, or a frame can be turned into a still image. In addition, some of the tree manipulation methods are also included among manipulation primitives. The multiple inheritance principle is used to associate the segment manipulation methods with the conceptual class.

To summarize, the conceptual model is a collection of conceptual constructs (classes), link patterns, presentation logic, and segment manipulation methods. A concept class is denoted by the concept (aggregate topic), and by the structural pattern as an ordered group of connection primitives. The concept presentation logic determines the ways the information is presented to the learner. Segment manipulation methods denote the operations that can be performed on each of the media within one presentation page.

5. WHAT IS IT WE WANT TO ACHIEVE?

In summary, the issues that we believe are not particularly well addressed include the dynamic model for the concurrency control of a large number of active multimedia objects with shared resources and constraints, and a high level of a user interactivity. We believe that behaviour of an ideal hypermedia system should comply with the following requirements:

1. All elementary objects within a page can be presented to the user without delay or conflict in resource requests.
2. Any presented page of the system is always complete and all objects within the page are synchronized, properly spatially placed, and scaled.
3. Any presented aggregate topic (page) can be dynamically connected to any other aggregate topic which will be subsequently presented.
4. Any link within a domain can be either static (permanent, defined in the design stage) or dynamic (created at run-time).

In real life situations, these requirements will be difficult to meet. Presentation logic (page composition and media synchronization) may be affected by system delays (e.g. artefact retrieval delays, network delays), availability of a given operating system resources, and other physical limitations. As per definition, multimedia objects are presented to the user as continuous stream of atomic elements (e.g. motion video is a sequence of frames) but they may be stored and retrieved in chunks. The system must guarantee uninterrupted retrieval and presentation of all elements of the multimedia composite object and its full synchronization with the other object(s).

Our approach is to simplify the conceptual model by using predefined constructs (structural patterns, presentation patterns and a set of manipulation methods). In the first stage, we built a static model that enables capturing of the content and structures of its objects. The conceptual model is used for generation of a functional prototype. The functional prototype is in turn used to test dynamic behaviour. In this stage, we examine systematically all aspects of flow of control (events, user interaction, sequencing of operations on active objects). The feedback loop and the corrective member (see Fig. 3) collect and classify events and interactions. These data are then used as input to the conceptual model and subsequent generation of the new version of courseware.

REFERENCES

1. G. A. Champine, D. E. Greer Jr and W. N. Ruh, Project ATHENA as a distributed computer system, *Computer* **23**(9), 40–51 (1990).
2. B. J. Haan, P. Kahn, V. A. Riley, J. H. Coombs and N. K. Meyrowitz, IRIS hypermedia services, *Commun. ACM*, **35**, 36–51 (1992).
3. M. Sherman, W. J. Hansen, M. McInerny, and T. Neuendorffer, Building hypertext on a multimedia toolkit: an overview of Andrew Toolkit hypermedia facilities, *Proc. First European Conference on Hypertext*, pp. 13–24 (1990).
4. D. McConnell, Computers, electronic networking and education: some American experiences, *Ed. Train. Technol. Int.*, **28**, 171–187 (1991).
5. J. Dospisil and T. Polgar, Application of software process assessment to hypermedia development environment, *Proc. 2nd International Interactive Multimedia Symposium*, Perth (1994).
6. J. A. Waterworth (ed.), *Multimedia: Technology and Applications*, Ellis Horwood, Chichester (1991).
7. R. K. Mahapatra and J. Courtney, Research issues in hypertext and hypermedia for business applications, *Database*, Fall, 12–19 (1992).
8. T. D. C. Little and A. Ghafoor, Interval-based conceptual models for time-dependent multimedia data, *IEEE Trans. Knowledge Data Engng*, **5**, 551–563 (1993).
9. R. G. Herrtwich, Time capsules: an abstraction for access to continuous-media data, *Proc. 11th Real-Time Systems Symposium*, Lake Buena Vista, FL, pp. 11–20 (1990).
10. R. C. Schank, Learning via multimedia computers, *Commun. ACM*, **36**, 54–56 (1993).
11. A. J. Palay, Towards an 'operating system' for user interface components, in M. M. Blattner and R. Dannenberg (eds), *Multimedia Interface Design*, ACM Press (1992).
12. S. Gibbs, L. Dami and D. Tsichritzis, An object oriented framework for multimedia composition and synchronisation, *MULTIMEDIA Systems, Interaction and Applications* (ed. L. Kejelldah), 1st Eurographics Workshop, Stockholm, pp. 97–112 (1991).
13. F. Arbab, I. Herman and G. J. Reynolds, An object oriented model for multimedia programming, *Proc. of Eurographics '93*, Vol. 12, No. 3, pp. C101–C113 (1993).
14. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorensen, *Object-Oriented Modelling and Design*, Prentice Hall, Englewood Cliffs, NJ (1991).
15. G. Booch, *Object Oriented Design with Applications*, Benjamin/Cummings, Menlo Park, CA (1991).
15. B. Meyer, Systematic concurrent object-oriented programming, *Commun. ACM*, **36**, 56–80 (1993).
16. B. Schneiderman, *Designing the User Interface: Strategies for Effective Human–Computer Interaction*, 2nd Edition, Addison-Wesley, Reading, MA (1992).
17. S. Gibbs, Ch. Breiteneden and D. Tsichritzis, Data modelling of time-based media, *Visual Objects* (ed. D. Tsichritzis), University of Geneva, pp. 1–21 (1993).
18. E. V. Berard, *Essays on Object-Oriented Software Engineering*, Vol. 1, Prentice-Hall, Englewood Cliffs, NJ (1993).