

3.2. *Linux HQ kernel documentation*

Linux HQ Kernel Documentation provides the following:

- Hypertext transformation of the code.
- Links to function definitions.
- Function search engine.

Similar to CVS file hierarchy with links to associated files, Linux documentation maintains alphabetic listing of file names. It supports a function search engine. Function calls are linked to function definition, and there are links to include header files. However, it still relies on the reader to understand the comments. More information can be found at: <http://www.linuxhq.com>.

3.3. *Variorum*

The American Heritage Dictionary (1998) defines “variorum” as follows: “contains notes or comments by many scholars or critics”. The creators of Variorum define it as a “multimedia tool that aids in the documentation of programs. . . . The integration of WWW capabilities is a key aspect of variorum’s usefulness”.

Variorum [10] allows programmers to record the process of “walking through” codes using multimedia technology. Variorum supports hypertext transformation of code and the addition of programmer/author walkthroughs as voice annotations. Variorum modifies the source code to include annotation links. However, its effectiveness depends critically on individual authors’ annotation style. The amount of voice storage can also become excessive. However, in the future audio/visual software documentation systems may overcome the deficiencies of today’s Variorum.

4. Design of Multimedia Applications Using Object-Oriented Tools

From the above survey, it can be concluded that multimedia is useful in software documentation, but whole-hearted incorporation of multimedia in software engineering has not yet happened [16]. There is an ongoing paradigm shift — from business orientation to entertainment orientation [16]. New software process models and paradigms, in particular the object-oriented approach, are needed in multimedia systems design. This section surveys several object-oriented approaches in multimedia systems design.

In what follows, we will discuss:

- DAMSEL — Dynamic Multimedia Specification Language.
- MET++ — Multimedia Application Framework.
- MME — Object-Oriented Multimedia Toolkit.
- PREMO — Presentation Environment for Multimedia Standard.

4.1. DAMSEL

DAMSEL is developed at the University of Minnesota. It includes the design and implementation of advanced multimedia constructs such as object-oriented extensible and temporal data model. It supports an execution environment based upon JAVA/CORBA. The temporal model describes along three axes: the temporal relations, delays and exec-based behavior. The representation using OO supports complex object definition and queries.

The temporal model of DAMSEL is very flexible. It supports user, system, application-generated events and therefore enables very interactive dynamic application creation. In the execution architecture, specifications are written to define the environment behavior and run time execution is determined by various events. DAMSEL specifications can be embedded with C++ or JAVA.

Data flow model assists in analyzing and modifying multimedia data. Data is modeled as stream flowing from a media source to a sink. It allows the insertion of additional stream objects as the data flows. It also enables modification and analysis of data. The presentation model provides higher level abstraction for dealing with and controlling presentations. Data flow is connected to a presentation server that interacts with the application.

The heart of DAMSEL is a distributed client-server run-time event manager, which is a multi-user, interactive, internet-wide execution environment implemented using CORBA and JAVA.

More information about DAMSEL can be found at its web site: <http://www-users.cs.umn.edu/~pazandak/damsel.html>

4.2. MET++

MET++ is an object-oriented application framework developed at the University of Zurich. It supports the development of multimedia applications using reusable objects for 2D, 3D graphics, audio, video, etc. The framework consists of a set of interconnected objects that provide the basic functionality of a working application, which can be easily specialized into individual application. Through subclassing and inheritance, it supports the reuse of code as a class library and the reuse of design structures. Similar dependencies between object are pre-implemented through pre-defined object composition, event dispatching and message flow.

Time synchronization [1, 19] is an important part of a multimedia presentation. In MET++, this is specified in a hierarchical composition. As illustrated by Fig. 1, implicit information is made visually explicit due to the hierarchy.

Example applications done using MET++ and more information about MET++ can be found at its web site: <http://www.ifi.unizh.ch/groups/mml/projects/met++>

4.3. MME

The MME Multimedia Extension is a project by Computer Graphics Center (Fraunhofer), Germany. The software package features a class hierarchy and

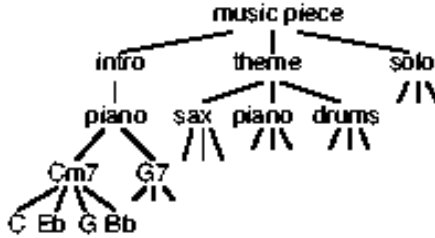


Fig. 1. Temporal composition.

development tools for composing multimedia applications. The goals of MME are:

- OO modeling of various media.
- Encapsulation of distributed media access and control.
- Modeling of time as a precondition to define arbitrary temporal relations.
- Relations between media objects designed at a higher abstraction level and its realization at run time.

MME objects have the following characteristics:

- Application (AO), Multimedia (MO).
- Media objects handle the transfer of media data from a set of ports (source) to another.
- Set of ports (sinks).
- Ports (devices like VCR, windows, files or sockets).
- Complex media objects handle the definition and maintenance of relations (temporal and spatial) objects.

MME is realized by executing:

- Instantiation of media objects out of predefined multimedia classes.
- Instantiation of complex media objects to define spatial/temporal relations.
- Definition of new media classes as subclasses of predefined objects and classes.

User interaction such as starting, stopping and cueing are supported in real time (interactive multimedia). MME is implemented in C++ on top of UNIX, Xwindows system with Xvideo extension, with about 4500 lines of code. The OO benefits include reuse, encapsulation, and less time to develop (about one man-year). More information about MME can be found at its web site: <http://zgdv.igd.fhg.de/www/zgdv-uir/software/MME>

4.4. *PREMO*

PREMO (Presentation Environment for Multimedia Objects) is a new standard under development by ISO. The goals of PREMO are:

- To provide a general framework and a reference model for the creation and programming of distributed multimedia applications.
- To allow existing media devices to be interfaced to an application.
- OO programming infrastructure to support the development.
- To recognize the evolution of multimedia system technologies for research tools to mature.
- To certify products that meets QoS and fundamental requirements.

PREMO is being developed at the CWI-Computer Center, Netherlands, and more information about PREMO can be found at its web site: <http://dbs.cwi.nl/cwwwi/owa/>

5. Specification and Design of Multimedia Software Systems

Specification and design of multimedia applications pose new challenge to authoring systems due to temporal and spatial relations. Common design of hierarchical composition of objects needs to be found, thus leading to object-oriented tools. For the specification of multimedia software systems, a new paradigm is espoused: software engineers will do evolutionary design of complex systems through: (1) architecture specification, (2) design rationale capture, (3) architecture V&V, and (4) architecture transformation, using an object-oriented architecture description language [24]. Another recent approach is to extend UML, the Universal Modeling Language, for the modeling of multimedia applications [22]. In what follows, we survey two more formal approaches in specification and design.

5.1. An actor based approach to multimedia software specification

Dattolo [11] applies the actor model for modeling software as collections of distributed, cooperative entities, as illustrated in Fig. 2. It is felt that the classical notion of object is too vague to support large-scale concurrency. On the other hand, actors combine object-oriented and functional programming in order to make the management of concurrency easier for the user. An actor reacts to the external environment by executing its procedure skills (scripts). An example for TeleoActor class definition based upon the ESAL (Extended Simple Actor Language) is as follows:

```
(Def TeleoActor
  {Actor}
  (stor info
  hypServices image
  inSuggestion brSuggestion cnSuggestion)
  [(apply-filter), (visualize), (tree-brws), (grph-brws),]).
```

The Dexter model for hypermedia is used, which is essentially a two-layer model — runtime layer and storage layer — for hypermedia. The architecture was