Chapter 2

Perspectives in Multimedia Software Engineering

As discussed in Chapter 1, we can view multimedia software engineering in two different, yet complementary, roles: 1) to apply multimedia technology to the practice of software engineering, or 2) to apply software engineering principles to the design of multimedia systems.

Multimedia has two fundamental characteristics that can be expressed by the following formula: *Multimedia = Multiple Media + Hypermedia.*

Advantages of multiple media are: 1) full utilization of all senses (eye, ear. etc.), 2) dynamic presentations, and 3) better understanding by the user. The disadvantages include: 1) greater demands on storage, bandwidth and computing resources, 2) cognitive overload, and 3) system complexity.

Hypermedia is a style of building systems for information representation and management around a network of multimedia nodes connected together by typed links [Hala95]. The advantages of hypermedia include: 1) ease of documentation, 2) ease of conceptualization and/or visualization, and 3) dynamic expansion of information hyperspace. The disadvantages are also well recognized: 1) disorientation due to "lost in hyperspace" phenomenon, 2) cognitive overload, and 3) system complexity.

In this chapter we will give a focussed survey of current research in MSE to apply multimedia technology to the practice of software engineering, or to apply software engineering principles to the design of multimedia systems. Since multimedia is basically multimedia objects plus links, in the following survey the concept of *links and association* will come up time and again as the central theme. The survey is by no means exhaustive, but the topics presented in this survey are a fair representation of the current research issues in multimedia software engineering.

1. PROJECT MANAGEMENT USING MULTIMEDIA TOOLS

One successful application of multimedia technology to software engineering is in project management using hypermedia CASE tools [Wild98]. Since the traditional project management tools lack the ability to capture a multitude of decisions and do not provide document control, a new Decision Based Systems Development paradigm (DBSD) was developed [Wild91]. In decision based systems development, usually the following steps are to be taken:

- Identification and Articulation of the Problem
- Identify Alternative Solutions
- Choose Decision Criteria
- Justify Alternatives
- Evaluate Conditional Decisions
- Put the Decision into Context
- Build Decision View

Multimedia technology allows the decision makers to use text for problem description, graphs and diagrams r for representing problem space, and different colors and symbols for denoting status and the latest information. Moreover, hypermedia allows the linking of documents and people in a variety of ways.

In the Decision-based Hyper-multimedia CASE (DHC) tool, objects in an extended document base are linked by five types of links: 1) *Reference to problems/decisions in problem space* or SEE links, 2) *Reference to a single on-line document* or REFER on-line links, 3) *Reference to a Decision View of a document set* or VIEW links, 4) *Timed reference to a contact person* or CONTACT links, and 5) *References to off-line documents* or REFER off-line links.

Some link types may be motivated by unix/DOS system commands. For example REFER is similar to xloadimage for bit-mapped pictures in DOCS.

This DHC tool was applied to Low-Visibility Landing and Surface Operations (LVLASO) project at NASA Langley Research Center. The impact on LVLASO is that the DHC tool is applicable to early stages in systems development. Also, brainstorming was easily documented and not lost. It encourages people to be more goal-oriented so that tasks that did not clearly fit in were immediately dropped. Finally it keeps people up to date with decisions. Future additions may include a CONTACT hyperlink, group Decisions by functional areas, and better GUI. The importance of this project is that it indicates the desirability of having many different types of links for information/people association.

2. SOFTWARE DOCUMENTATION

Another successful area of application of multimedia technology is in software documentation. By exploiting the nature of hypermedia, powerful multimedia-based program documentation systems can be developed.

2.1 RST Documentation Model

Reliable Software Technologies Documentation Model uses standard C comments with the addition of design documents on the WWW. Standard C Comments are added to program lines whose purpose is not clear, to the beginning of each function or procedure to explain its purpose and the Pre and Post conditions, and to the header function to describe its purpose.

The following information is also maintained:

- CVS revision history
- Requirements document
- Relevant research
- Architecture diagram
- UML diagrams

RST documentation web page includes the following information:

- Welcome to the new program web page
- Short description of the project goals and the specific problem to solve.
- Related links
- Project design document
- Project specifications document
- Architecture document
- Research Links: Link1, Link2, Link3

2.2 Linux HQ Kernel Documentation

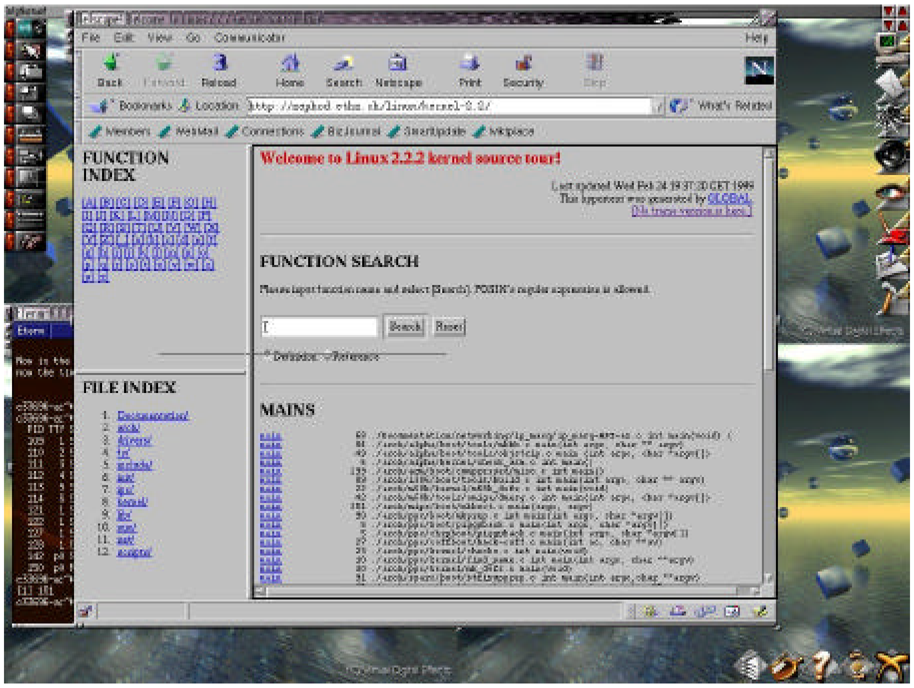
Linux HQ Kernel Documentation provides the following:

- 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 included header files. However, it still relies on the reader to understand the comments.

An example of Linux HQ Documentation of the Linux Kernel is illustrated in Figure 1. More information can be found at: http://www.linuxhq.com



FUNCTION INDEX

FILE INDEX

1. Documentation/
2. arch/
3. drivers/
4. fs/
5. include/
6. init/
7. ipc/
8. kernel/
9. lib/
10. mm/
11. net/
12. scripts/

Welcome to Linux 2.2.2 kernel source tour!

Last updated Wed Feb 24 19:37:30 CET 1999
This hypertext was generated by GLOBAL.
[No frame version is here.]

FUNCTION SEARCH

Please input function name and select [Search]. POSIX's regular expression is allowed.

Definition Reference

MAINS

```
main    69 /Documentation/networking/ip_masq/ip_masq-API-ex.c int main(void) {
main    81 /arch/alpha/boot/tools/mkbb.c main(int argc, char ** argv)
main    40 /arch/alpha/boot/tools/objstrip.c main (int argc, char *argv[])
main     8 /arch/alpha/kernel/check_asm.c int main()
main   135 /arch/arm/boot/compressed/misc.c int main()
main    83 /arch/i386/boot/tools/build.c int main(int argc, char ** argv)
main    22 /arch/m68k/kernel/m68k_defs.c int main(void)
main    42 /arch/m68k/tools/amiga/dmesg.c int main(int argc, char *argv[])
main   181 /arch/mips/boot/addinitrd.c main(argc, argv)
main    50 /arch/ppc/boot/mkprep.c int main(int argc, char *argv[])
main     9 /arch/ppc/boot/piggyback.c main(int argc, char *argv[])
main     5 /arch/ppc/chrpboot/piggyback.c main(int argc, char *argv[])
main    27 /arch/ppc/coffboot/hack-coff.c main(int ac, char **av)
main    25 /arch/ppc/kernel/checks.c int main(void)
main    20 /arch/ppc/kernel/find_name.c int main(int argc, char **argv)
main    20 /arch/ppc/kernel/mk_defs.c main(void)
main    51 /arch/sparc/boot/piggyback.c int main(int argc,char **argv)
```

Figure 1. Documetation of the Linux Kernel – An example.

2.3 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 [Chiu98] 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.

3. 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 [Hira99]. There is an ongoing paradigm shift -- from business orientation to entertainment orientation [Hira99]. 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

3.1 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 3 axes: the temporal relations, delays and exec-based behavior. The representation using OO supports complex object definition and queries.

The DAMSEL execution environment is illustrated in Figure 2. The DAMSEL implementation architecture is illustrated in Figure 3.

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.

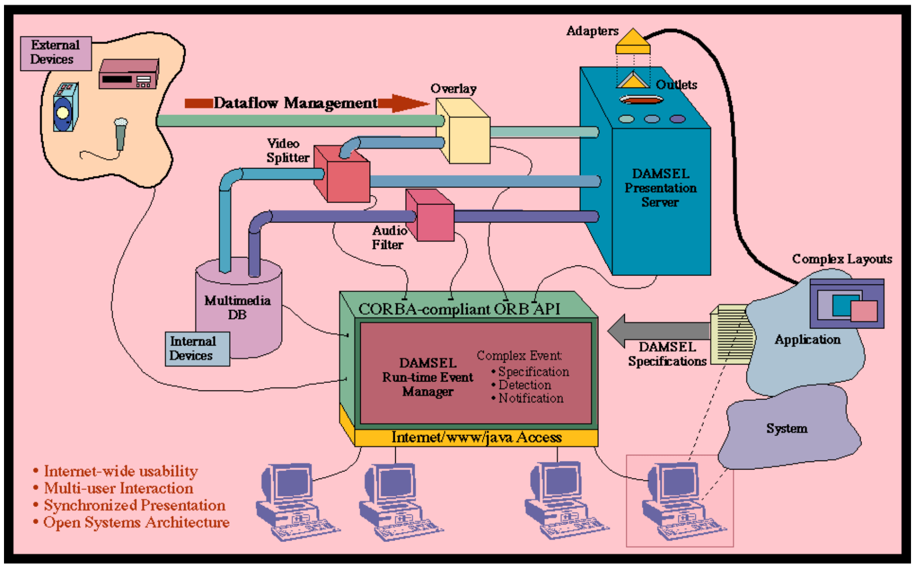


Figure 2. DAMSEL execution environment.

Data flow model assists in analyzing and modifying multimedia data. Data is modelled 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

3.2 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 predefined object composition, event dispatching and message flow.

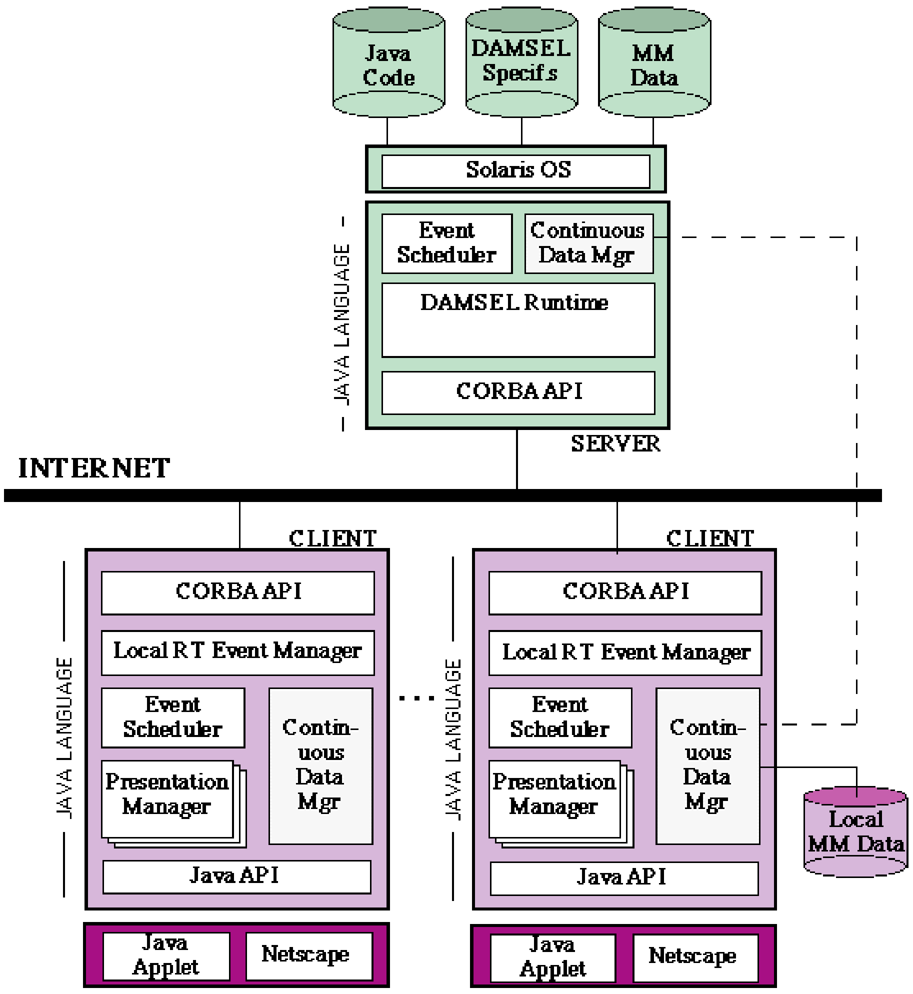


Figure 3. DAMSEL implementation architecture.

Figure 4 illustrates the MET++ architecture, which is based on OO application framework ET++. The ET++ OO class library integrates user interface building blocks, basic data structures, support for object I/O, and high-level application components. 3D graphics, audio classes, time structures are added for multimedia support. Hardware dependencies are hidden in a portability layer.

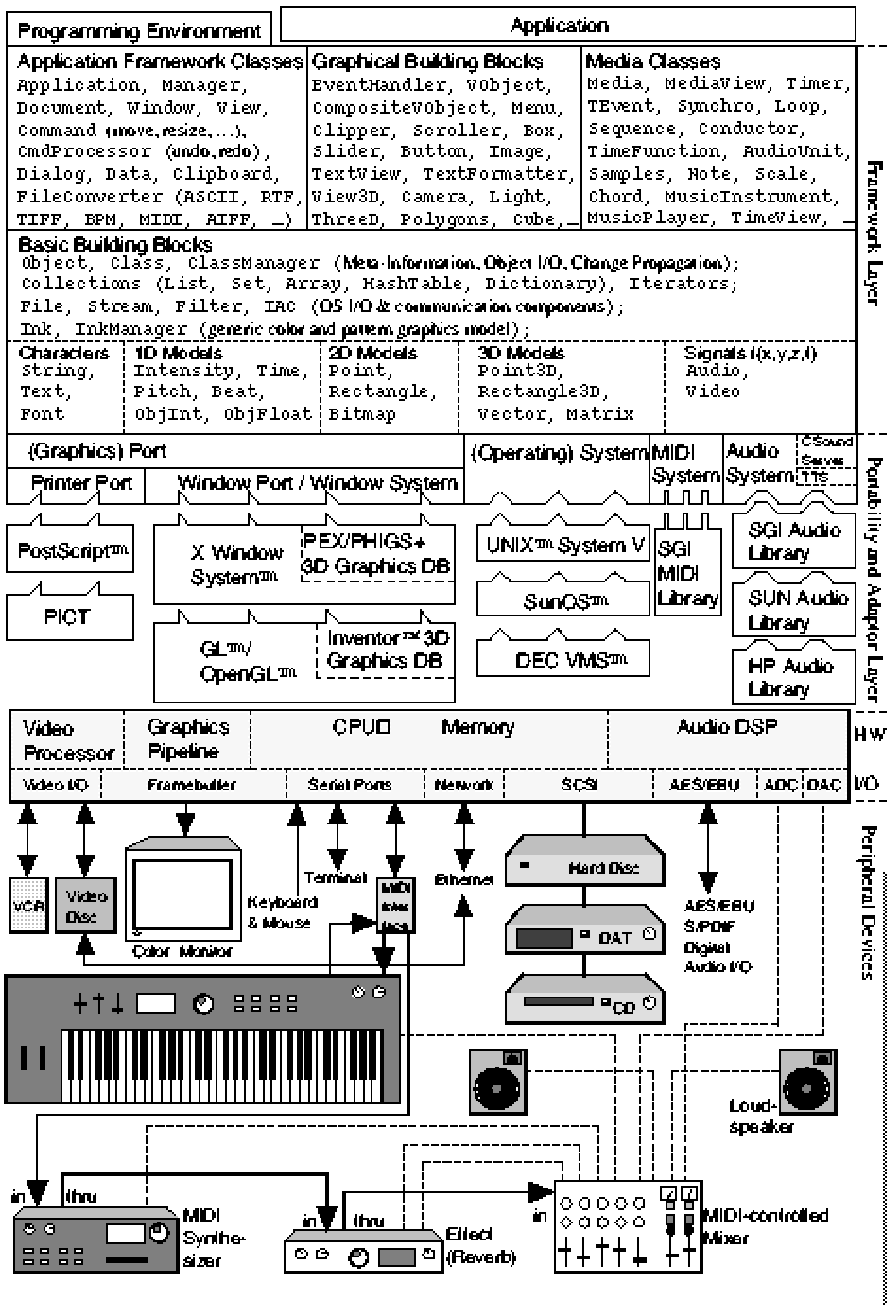

Programming Environment
Application
Application Framework Classes
Application, Manager, Document, Window, View, Command (move, resize, ...), CmdProcessor (undo, redo), Dialog, Data, Clipboard, FileConverter (ASCII, RTF, TIFF, BPM, MIDI, AIFF, ...)
Graphical Building Blocks
EventHandler, VObject, CompositeVObject, Menu, Clipper, Scroller, Box, Slider, Button, Image, TextView, TextFormatter, View3D, Camera, Light, ThreeD, Polygons, Cube, ...
Media Classes
Media, MediaView, Timer, TEvent, Synchro, Loop, Sequence, Conductor, TimeFunction, AudioUnit, Samples, Note, Scale, Chord, MusicInstrument, MusicPlayer, TimeView, ...
Basic Building Blocks
Object, Class, ClassManager (Meta-Information, Object I/O, Change Propagation);
Collections (List, Set, Array, HashTable, Dictionary), Iterators;
File, Stream, Filter, IAC (OS I/O & communication components);
Ink, InkManager (generic color and pattern graphics model);
Characters
String, Text, Font
1D Models
Intensity, Time, Pitch, Beat, ObjInt, ObjFloat
2D Models
Point, Rectangle, Bitmap
3D Models
Point3D, Rectangle3D, Vector, Matrix
Signals f(x,y,z,t)
Audio, Video
Framework Layer
(Graphics) Port
Printer Port
Window Port / Window System
(Operating) System
MIDI System
Audio System
CSound Server
TTS
PostScript™
PICT
X Window System™
PEX/PHIGS+ 3D Graphics DB
GL™/ OpenGL™
Inventor™ 3D Graphics DB
UNIX™ System V
SunOS™
DEC VMS™
SGI MIDI Library
SGI Audio Library
SUN Audio Library
HP Audio Library
Portability and Adaptor Layer
Video Processor
Graphics Pipeline
CPU
Memory
Audio DSP
HW
Video I/O
Framebuffer
Serial Ports
Network
SCSI
AES/EBU
ADC
DAC
I/O
VCR
Video Disc
Color Monitor
Keyboard & Mouse
Terminal
MIDI Interface
Ethernet
Hard Disc
DAT
CD
AES/EBU S/PDIF Digital Audio I/O
Peripheral Devices
Loud-speaker
in
thru
MIDI Synthe-sizer
in
thru
Effect (Reverb)
in
MIDI-controlled Mixer

Figure 4. MET++ System Architecture.

Time synchronization is an important part of a multimedia presentation. In MET++, this is specified in a hierarchical composition. As illustrated by Figure 5, implicit information is made visually explicit due to the hierarchy.

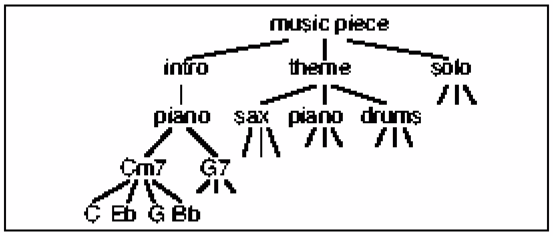


Figure 5. Temporal composition.

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++

3.3 MME

The MME Multimedia Extension is a project by Computer Graphics Center (Fraunhofer), Germany. The software package features a class hierarchy and development tools for composing multimedia applications. The goals of MME are:

- OO modelling of various media
- encapsulation of distributed media access and control
- modelling 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-uig/software/MME

3.4 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/

4. SPECIFICATION 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 [Tsai99].

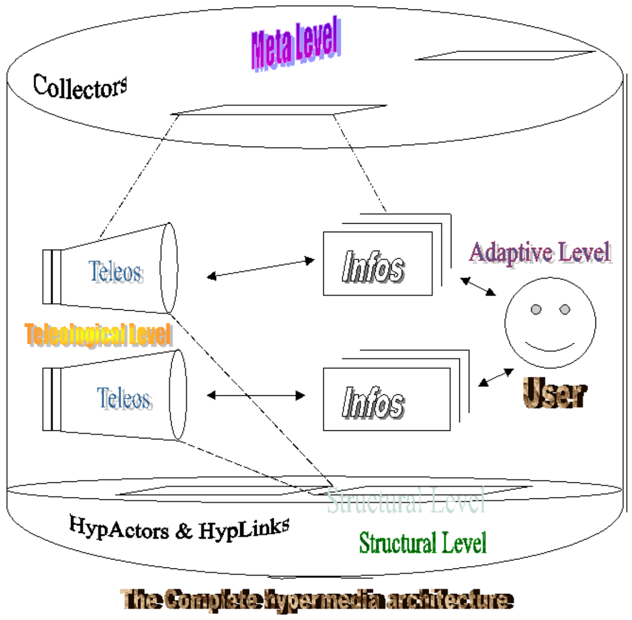


Figure 6. Hypermedia information system based upon the actor model.

Dattolo [Datto97] applies the actor model for modelling software as collections of distributed, cooperative entities, as illustrated in Figure 6. 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 applied in the development of a hypermedia system named DiBlue, which is a distributed version of Blue, a traditional OPLA hypermedia programming environment. It supports an object-oriented logic programming system in OPLA, a hybrid language originated from the marriage between Prolog and CLOS.

5. MODEL- AND PATTERN-BASED DESIGN APPROACHES

5.1 Model-based Approach to Hypermedia Design

In the model-based approach to hypermedia design, the key concept is to provide a comprehensive model for software specification and design. For instance, the Relationship Management Methodology (RMM) comprises 1) Entity-Relationship design, 2) Application diagram design, 3) M-slice (aggregate) design, 4) Navigational design, 5) User interface design, 6) Protocol conversion design, 7) Run-time behavior, and 8) Construction and testing [Isak96].

The Relationship Management Data Model (RMDM) is the cornerstone of the RMM methodology. It includes elements for representing information domain concepts such as entities and relationships, along with navigation elements such as links. As illustrated by Figure 7, an application design is described via an RMDM diagram.

During the E-R Design a study of the relevant entities and relationships of the application domain is conducted. These elements form the basis of the hypermedia application and show up as nodes or links. During the navigational design, relevant relationships are identified and made available for navigation. Since information units can have a very large number of attributes, the RMM groups the attributes into slices called M-Slices.

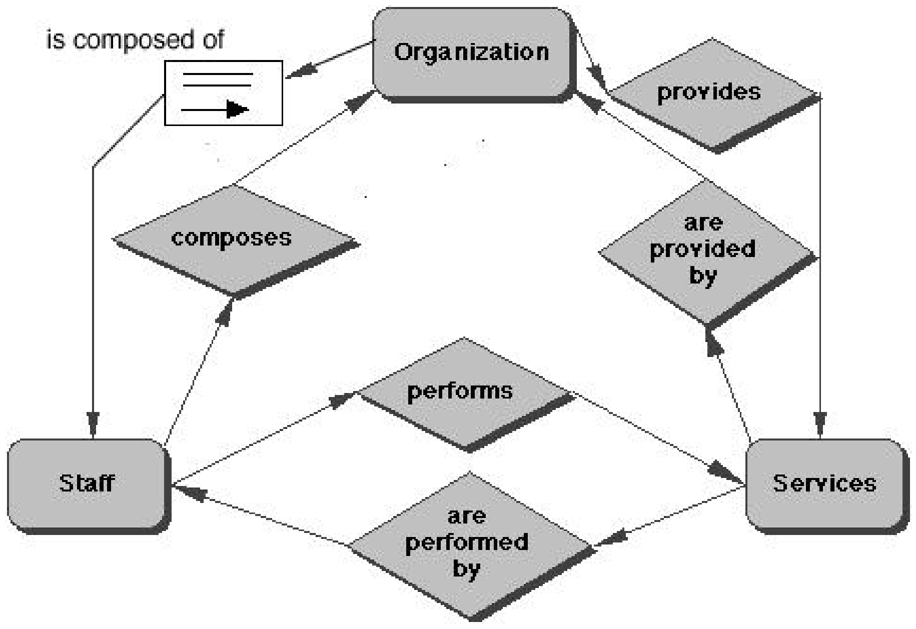


Figure 7. An RMDM diagram.

The way navigation is applied in the application by RMM is through the Access Primitives. The RMM Access Primitives include: 1) Unidirectional link, 2) Bi-directional link, 3) Grouping, 4) Conditional index, 5) Conditional guided tour, and 6) Conditional indexed guided tour.

Regarding links, navigation establishes an association between an original data component and destination data components. The associative link, or simply link, is what effectively allows navigation through data components, and represents the main characteristic of hypertext and hypermedia. The consideration of link types by definition establishes semantic information that allows hypermedia systems to efficiently manage data or data modelling, preventing this task to be the sole responsibility of designers or users. Link types can be used through almost every step of model-based approaches to hypermedia design.

What presently are found in most of the hypermedia systems are static, persistent and explicit links, defined over content. In RMM, there are links that can initiate a process such as email, a video exhibition, an audio exhibition, or a file download. Finally, *destination influence* establishes what of the operational services is going to be supported by the link. Links types are further broken down into two categories:

- Historical category (related to performed navigation)
- Speculative category (related to possible navigation)

A category that supports the retrieval service allows the presentation of the meaning associated to the description when it exists. *Destination influence* can provoke the definition of new links that were not initially related to infological modelling.

In conclusion, a framework is presented to categorize link types, which allows the design and construction of richer hypermedia applications. Link design, which is the kernel of model-based approaches to hypermedia design, is one of the areas that can benefit from the use of link types. Both information and operational elements influence link creation, so these issues should be considered during link design efforts. Information influence is related to conceptual relationships from the domain of interest, while operational influence is related to data manipulation services. It is also proposed that service and destination influences should be considered during link design. Links types and categories can also be identified and considered both with model based approaches and with authoring tools to hypermedia design.

5.2 Pattern-based Approach to Design

In the pattern-based approach to design the idea is to identify important and relevant design patterns, so that complex design can be constructed from these design patterns [Lyar97]. This approach can be applied to hypermedia applications [Garr97]. The patterns for hypermedia applications mainly deal with navigational structure and interface organization. The navigational contexts may include: class-derived context, link-derived context, composite-derived context, and arbitrary context. Two important navigational structures are *News* for new information and *Landmarks* for subsystems. In what follows, one such pattern will be discussed.

News is motivated by the following problem: “Given a large and dynamic web site, how do you tell users that there is new information or updates somewhere inside the site?” In huge web sites, it is not a good solution to compromise structure in order to make updates easy available. The solution is to structure the home page in such a way that a space is devoted to the newest additions:

- Presenting descriptive “headlines” regarding them
- Using those headlines as anchors to link them with their related pages

In summary a software engineering strategy for developing hypermedia applications is proposed, emphasizing the use of an object-oriented design method, the use of design patterns, and the partitioning of the development life cycle in a set of activities addressing different design concerns. The design patterns may help to capture and reuse experience and to provide more comfortable and understandable navigation space.

5.3 Virtual Multimedia Object

Links can be explicit or virtual. For example, queries can be considered as virtual links. Multimedia objects can also be real or virtual. An example of virtual multimedia objects is illustrated in Figure 8.

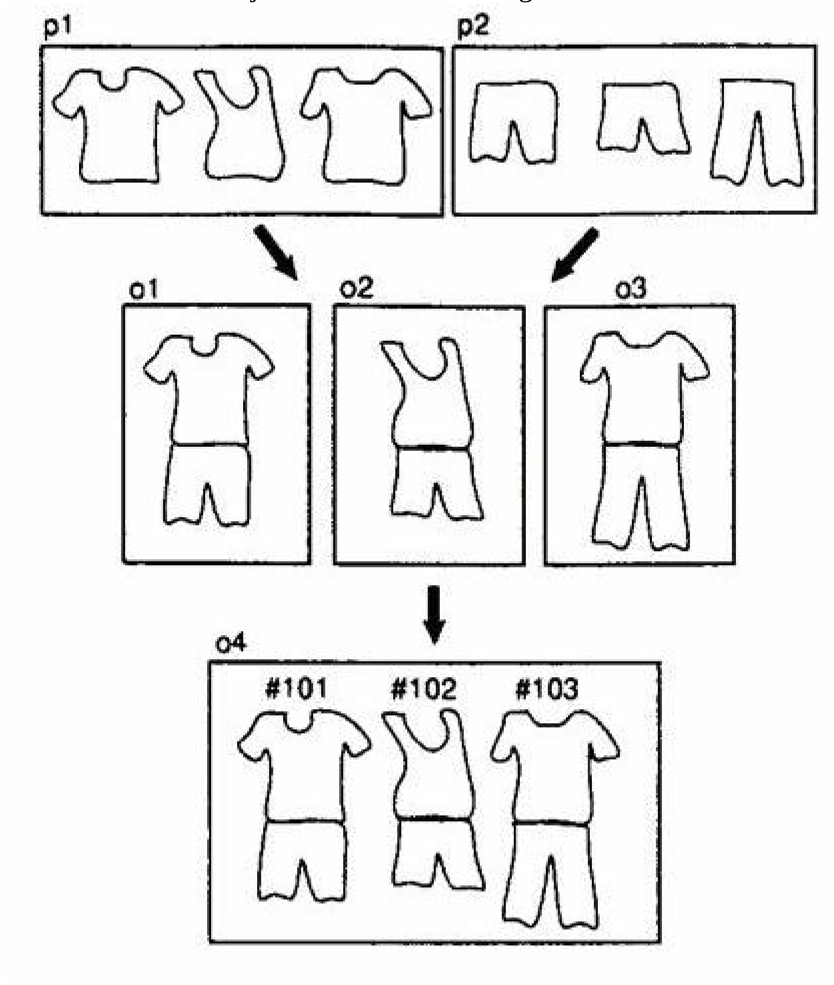


Figure 8. Virtual multimedia objects.

The Virtual Multimedia Object (VMO) [Hoch98] is virtually created based upon other multimedia objects. A pointing entry (a kind of virtual link) represents the mapping between a VMO and its original data. It can be implemented as user-defined deriving procedures.

A prototype system has been constructed under Solaris 2.5.1 in C++. The system has two layers: Virtual Object layer and Storage layer. Virtual Object Layer manages VMO and their types. Storage layer has the responsibility for storing objects. The technique can be applied to any media type.

6. MULTIMEDIA SOFTWARE PROJECT EFFORT MEASUREMENT

In Section 1 of this chapter we discussed how multimedia technology is utilized in software engineering project management. In this section we discuss how to apply software engineering technology in multimedia project management, and address the issue of multimedia software project effort measurement.

Why multimedia software project effort measurement is an issue? Can't multimedia projects be treated in the same way as other types of software projects? To answer this question, we note that there have been substantial efforts in measuring traditional transaction processing & process control systems, but relatively little effort in determining and evaluating multimedia effort measurements. In fact multimedia software is unique in its emphasis on contents and the need for storyboarding early in the development cycle. Because of that, we not only need tools and techniques specific to multimedia, but also should emphasize different aspects in software project management. For one, managers from diverse background are needed for such projects, to form cross-disciplinary teams. For another, due to the inherently different nature of multimedia systems, the direct application of existing models and measures may not meet multimedia project management needs.

A preliminary empirical study to develop an algorithmic mapping to 'effort' using product characteristics appropriate to multimedia software systems other than LOC or external files is reported in [Fletc98]. The development effort is regarded as a function of 1) building the system content - each media form might have a different impact on development effort - including file name, media type, creation effort, digitizing effort, editing effort, and 2) Authoring the system - a screen that incorporates a greater number of objects and events would take proportionally greater effort to develop. The empirical study on projects developed by 4th year students and delivered in two stages: prototype and final delivery. The students are information science students focussed on programming and design students concentrated on content and interface development. Forty five observations were recorded in media component development, but no correlation between development effort and either of the component variables - media type and media status. Since data sets were small, generalizations are not possible from this study.

There is a lack of industry driven determination of important attributes. Moreover, data collection needs to be exploratory and student projects are not representative of industry developments. Thus there is a need to develop an industry-based metrics framework in order to determine system and component characteristics considered influential in multimedia software system development efforts. The Goal/Question/Metric (GQM) model shown in Figure 9 is one such framework.

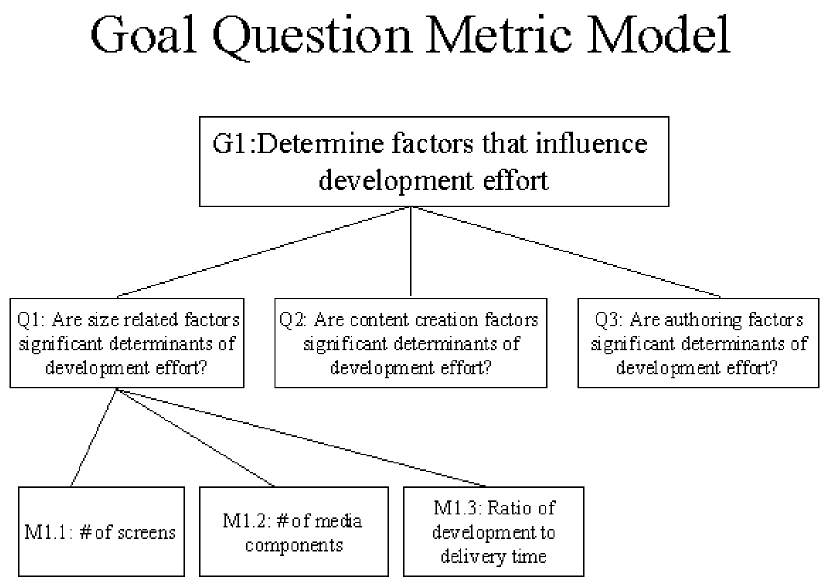

Goal Question Metric Model
G1:Determine factors that influence development effort
Q1: Are size related factors significant determinants of development effort?
Q2: Are content creation factors significant determinants of development effort?
Q3: Are authoring factors significant determinants of development effort?
M1.1: # of screens
M1.2: # of media components
M1.3: Ratio of development to delivery time

Figure 9. The Goal/Question/Metric Model (a small portion is shown for illustrative purpose).

The key issues considered to reflect industry perspective include:

- Development Tools: Authoring tools and other more complex tools offering visually based high productivity development languages or high level scripting languages.
- Delivery Platform: Platform dictates media format and optimization considerations, for example, CD-I must conform to PAL or NTSC.

- Content Development: Media content development is labor intensive, but difficult to quantify. It depends on media mix, the number of components, and complexity of each component. Artistic considerations also play a role in satisfactory completion of project.
- Organizational Capability: Indirect effects such as size, personnel mix, etc.
- Personnel: Wide variety of skills is required in successful multimedia software development. The core members are a producer, programmer(s), and graphic artist(s).

To perform preliminary verification of the above framework, a pilot study was conducted [MacDo98]. Structured interviews were conducted with three multimedia development organizations. Results suggest the type of project almost entirely determines development environment and workload. Results of pilot study were combined with components of GQM framework to develop a postal survey. The focus is to determine factors that influence development efforts. It was found that fifty per cent of respondents use no formal methodology. None of the traditional software metrics is used. Rather, experience from previous projects is used. Project tracking is performed at very high level, indicating immature project management. Staff experience and project size are obviously important factors.

To summarize, 1) traditional software metrics such as COCOMO and FPA are usually not utilized in multimedia software development; 2) pilot study and questionnaire reinforce the non-formal approach to multimedia software development; 3) although there are attempts to formalize the approach to software development, there is uncertainty on how to achieve this. In other words, multimedia software engineering as a scientific discipline is still evolving, making it an exciting research area to explore.