The evolving role of software engineering in the production of multimedia applications

Timothy Arndt

Department of Computer and Information Science,
Cleveland State University, Cleveland, Ohio, USA;
arndt@cis.csuohio.edu

Abstract

The traditional approach to software engineering is based on the waterfall model or some variation of this model. It is well known, however, that this approach is not particularly well suited to the production of one-of-a-kind or experimental systems due to fuzzy user requirements - users are not sure what can be accomplished. Since most multimedia applications are currently of this type, we must search for alternative approaches. One such alternative is based on rapid prototyping in which we quickly generate a working system that we use to validate user requirements. For distributed multimedia applications, we have experimented with a prototyping environment based on grammar formalisms and utilizing web technologies such as HTML/XML to generate a working prototype. Such an approach seems to be promising and we are currently pursuing this line of research. For the future, researchers working at the interface of software engineering and multimedia systems may identify high-level abstractions common to such applications which will permit the traditional approach to software engineering, backed up with appropriate CASE tools, to be used.

1. Introduction

Multimedia applications are becoming increasingly important in areas such as education (digital libraries, training, presentation, distance learning), healthcare (telemedicine, health information management, medical image systems), entertainment (video-on-demand, music databases, interactive TV), information dissemination (news-on-demand, advertising, TV broadcasting), and manufacturing (distributed manufacturing, distributed collaborative authoring)[1]. As such applications move from the realm of research prototypes to production systems, the need to apply software engineering techniques to the production of these applications becomes more pressing. In this paper, I will examine the role that traditional software engineering can play in the production of multimedia applications as well as where and how traditional approaches need to be modified for the special needs of multimedia.

2. Traditional Software Engineering

Traditional software engineering emerged around 30 years ago as software projects became increasingly ambitious leading to missed deadlines, cost overruns, and difficulties in software project management. One of the major accomplishments in the field of software engineering was the recognition of the software lifecycle. The waterfall model of software engineering views the production of software as being divided into a number of distinct phases: problem definition; requirements analysis; program specification; coding; testing; and maintenance. Each phase of the lifecycle results in the production of one or more deliverables. These deliverables give visibility to the software lifecycle and allow managers to better plan and control software projects [15].

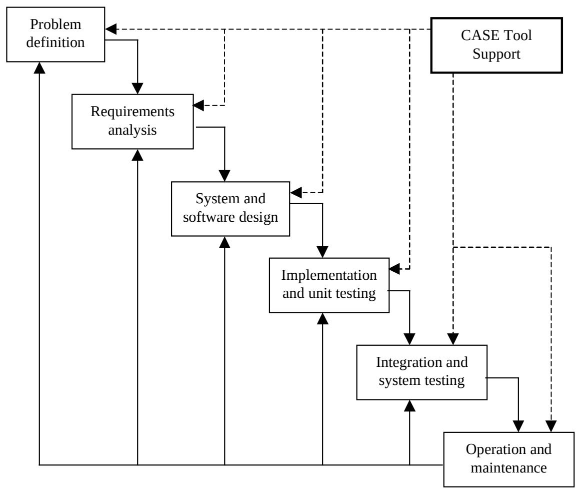

Problem definition
Requirements analysis
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance
CASE Tool Support

Figure 1 – Traditional Software Engineering

An organization may have a standard set of techniques to be applied to each one of these phases based on such concepts as structured analysis and programming, object-oriented analysis, design and programming, etc. The organization then follows a certain pre-defined software process model. For maximum productivity, the various components of a software process model should be supported by a set of Computer Aided Software Engineering (CASE) Tools. Case tools are usually based on such well-known software engineering abstractions as data flow diagrams, the entity-relationship model, and objects.

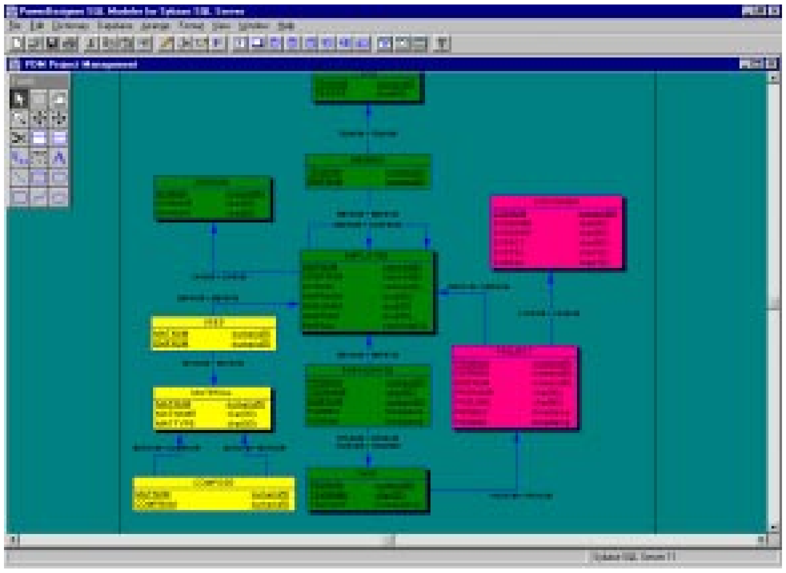


Figure 2 – A Typical CASE Tool

While the approach described above works well for many software projects (in particular for traditional EDP projects), it is well known that one of a kind or experimental systems present particular problems for such an approach. The principal problem associated with one of a kind or experimental systems is the difficulty of establishing requirements before production has commenced. Users have difficulties stating requirements for types of systems for which they are not familiar, and developers have difficulties understanding what can be accomplished given time and budget constraints. Another complicating factor may be the unavailability of CASE tools capable of supporting the project.

For critical components of software systems, an approach based on formal specification is often indicated [13]. Formal specifications allow for a greater degree of certainty concerning the behavior of the component being specified to be obtained. Since the component is represented by a mathematical formalism, properties of the component can be proved and the representation can be manipulated mathematically. Formal specifications also hold out the promise of being able to transform the specification into a working program, although this is not yet a widespread reality outside of research labs. Due to the difficulty of composing formal specifications, their use has been limited to critical components of real-time systems

3. Modifications to the Software Lifecycle for Multimedia Applications

For systems in which it is difficult for the user to precisely state a set of requirements that can be used to guide the rest of the lifecycle, an approach based on prototyping is often indicated. The prototyping approach starts with a minimal set of requirements and/or a problem definition and uses this information to quickly produce a running system which can be used by the user to more clearly define the requirements for the project. A running prototype allows the user to experiment with the system and to see which parts of the system need to be reworked. It is not necessary to have a completely functional prototype in order to elicit requirements, performance considerations may be put off until the actual system is implemented. In a typical prototyping approach, the prototype may need to be reworked several times, based on user feedback, in order to come up with a complete set of requirements.

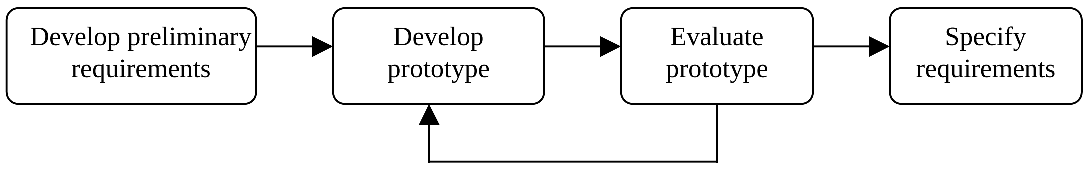


Figure 3 – Prototyping

The prototyping approach has several variations. In one variation, the prototype serves merely to elicit user requirements. This is known as throw-away prototyping. In evolutionary prototyping on the other hand, the working prototype is modified and refined to become the working system [6]. This requires a more disciplined approach to the production of the prototype since it will become a part of the production system.

Prototyping is often supported by specialized tools such as fourth generation languages (4GLs), screen generators, report generators, etc. The existence or lack thereof of such tools can be a determining factor in the success of prototyping.

Besides the difficulty of producing a set of requirements, multimedia applications have other characteristics that affect the software lifecycle. The traditional DP application is long-lived, therefore a major part of the effort associated with such applications is maintenance and upgrade. Multimedia applications, on the other hand, will probably have a much shorter lifespan due to the rapidly changing environment in which they operate. Multimedia standards, de facto and de jure, as well the networking infrastructure and operating system support for multimedia are evolving rapidly. In this situation, few multimedia applications are likely to be long-lived.

The CASE tools that are used to support the production of traditional software products are not useful for the production of multimedia applications since they are largely based on the manipulation of alphanumeric data, while continuous media are not handled. Persistent data is generally assumed to be stored in a relational database (e.g. entity-relationship models are translated to tables in a relational database) while many researchers argue that object-oriented databases are the way to go for multimedia applications [12].

The fundamental problem regarding CASE tools for multimedia application development is that the underlying abstractions (DFDs, E-R diagrams, etc.) are best suited to DP applications and are not well suited for multimedia. A number of researchers have studied abstractions which might be useful for multimedia [11]. These abstractions usually focus on synchronization among various multimedia objects [10], layout of multimedia objects [16] and user interactions [9]. None of these abstractions has been widely adopted, however.

4. The MICE Approach

The Multimedia IC Developer's Environment (MICE) is a set of tools being developed at the University of Pittsburgh, Cleveland State University, and the University of Salerno to support the construction of multimedia applications. The most important aspects of MICE are the following: a powerful underlying abstraction for multimedia applications – the Teleaction Object (TAO); integrated toolkit based on this abstraction; rapid prototyping through visual developer's tools; formal specification of the media objects interactions; visual language interface to the formal specification language; browser-based prototype development using a TAO-specific extension of HTML.

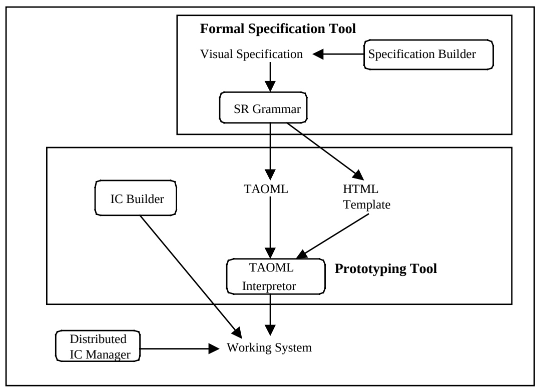
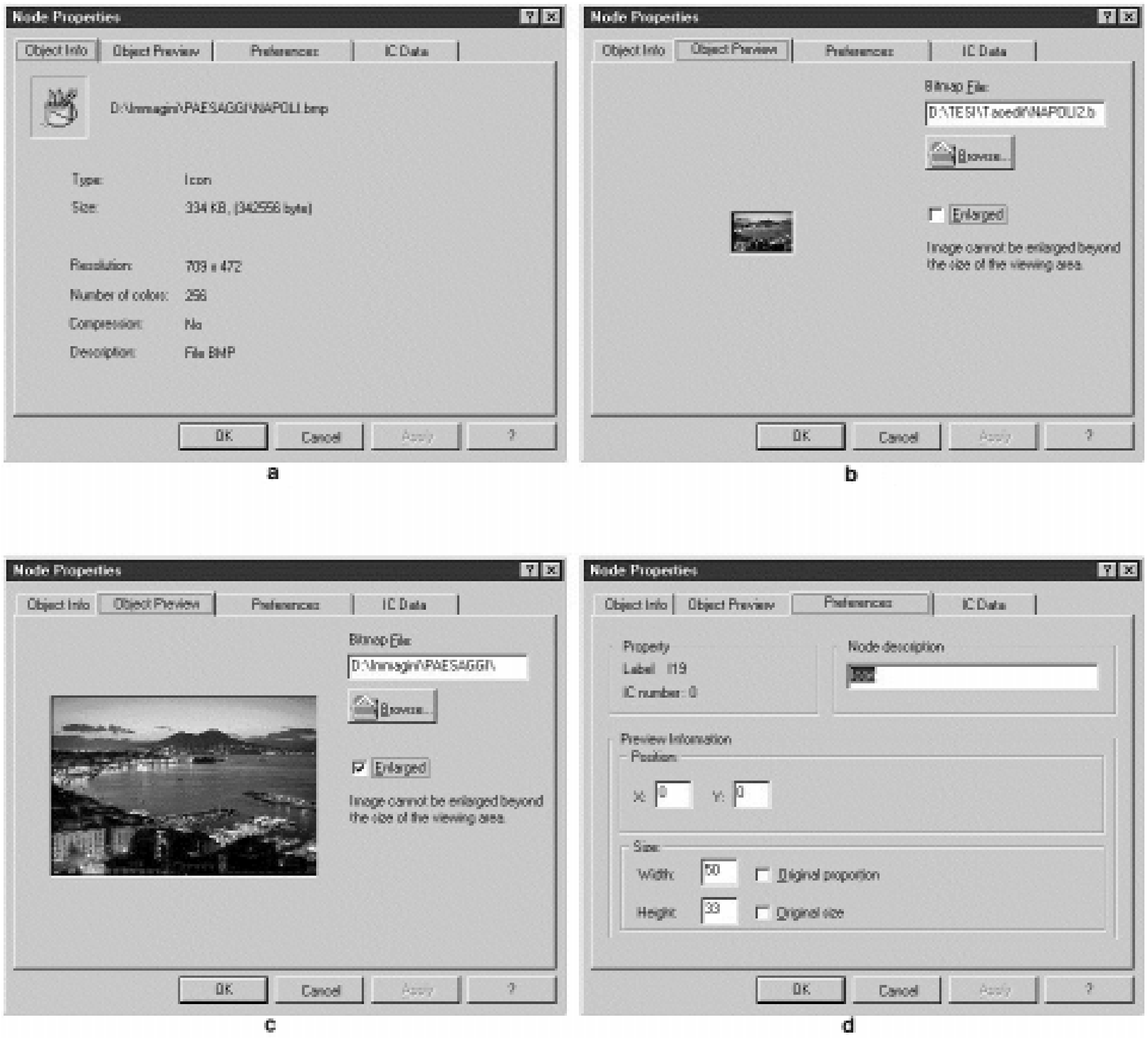


Figure 4 – MICE Application Development

Teleaction Objects complex multimedia objects with an attached knowledge structure [3,4]. The TAO may consist of a combination of multimedia objects (audio, video, image, etc.) combined using spatial, temporal and logical operators. The physical part of the TAO forms the user's interface with the TAO. The aspect which makes TAOs particularly powerful is the attached knowledge structure which is structured as an active index [5]. The active index allows the TAO to react to external events and to adapt to a changing environment. Several applications have been developed using TAOs.

The MICE approach then is based on CASE tools supporting the TAO paradigm. The IC builder allows the user to graphically construct the knowledge structure of a TAO while the TAOML builder is a graphical tool which allows the user to construct the physical part of a TAO. The TAOML builder is based on an SR grammar which provides a formal specification of the physical aspects of the TAO. The result of parsing the SR grammar is the production of the physical part of the TAO. In order to allow for rapid prototyping of distributed multimedia applications, the TAO is implemented using TAOML, an extension of HTML. The TAOML Builder interprets the visual specification and produces TAOML as output. This output in turn is given as input to the TAOML interpreter which produces standard HTML corresponding to the TAO specified by the user.



D:\Immagini\PAESAGGI\NAPOLI.bmp

Type:	Icon
Size:	334 KB, (342556 byte)
Resolution:	709 x 472
Number of colors:	256
Compression:	No
Description:	File BMP

Bitmap File: D:\TESI\Taoedit\NAPOLI2.b

☐ Enlarged

Image cannot be enlarged beyond the size of the viewing area.

Bitmap File: D:\Immagini\PAESAGGI\

☑ Enlarged

Image cannot be enlarged beyond the size of the viewing area.

Property
Label: I19
IC number: 0

Node description

Preview Information
Position
X: 0 Y: 0

Size
Width: 50 ☐ Original proportion
Height: 33 ☐ Original size

Figure 5 a-d - Tabbed Dialog Showing Node Properties

The TAO is the abstraction that a designer of a multimedia application needs in order to express his design of the system in terms which are closer to his way of thinking about the application than the actual implementation of the application. In order to make the TAO abstraction still more useful to the user, a visual representation of the TAO is used by the TAOML builder (see figures 5-7). This visual representation is the crucial point in the success of commercial CASE tools, and we hope that it will make the MICE environment useful for multimedia application designers. Another visual representation is given for the index cells which represent the knowledge of the system in the IC Builder tool.

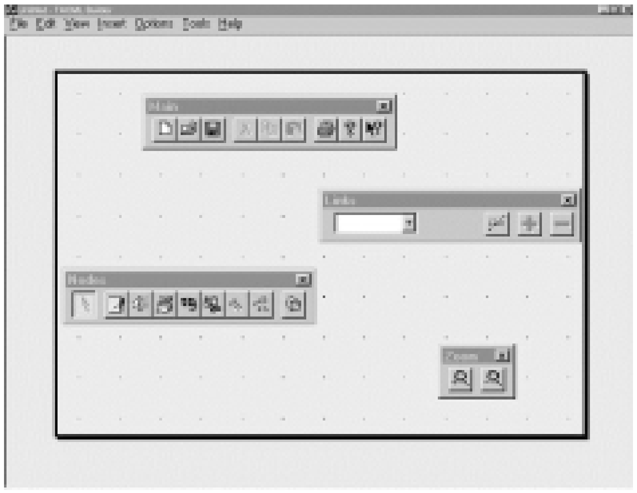


Figure 6 - TAOML Builder Toolbars

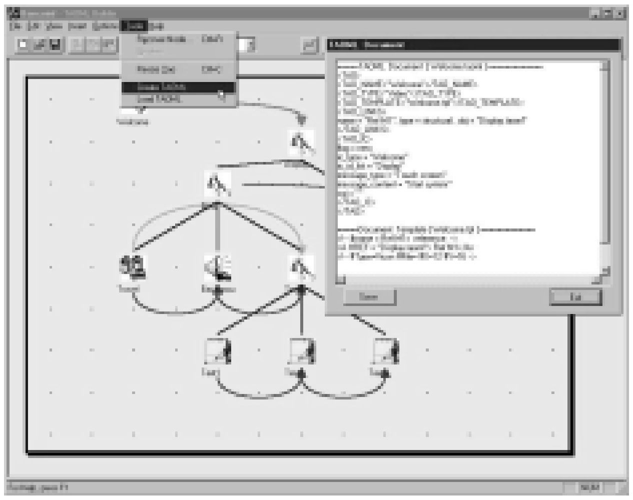


Figure 7 - Hypergraph and Matching TAOML

5. Future Directions

The output of the TAOML builder is currently TAOML, an extended form of HTML which cannot be understood by standard browsers. For this reason, the TAOML must first be translated by the TAOML interpretor to produce standard HTML. We are currently investigating the implementation of TAOML as an XML application. XML is a standard currently being formulated by the W3C committee which is used to describe structured documents. XML allows the contents of documents to be structured in such a way that they can be parsed by XML parsers to ensure correctness. XML will be implemented in the next generation of web browsers, thus making TAOML an XML application will ensure that applications prototyped with MICE can be run in standard browsers.

In section 4, I concentrated on showing how the MICE environment implements CASE tool-like functionality by supporting the TAO paradigm for multimedia application development. I believe that the widespread adoption of TAO or some other abstraction for multimedia is a necessary prerequisite to the development of commercial CASE tools which should lead to much greater efficiency in multimedia application development. This adoption, along with the adoption of multimedia standards (for operating systems and networks) would remove multimedia from the category of one-of-a-kind systems. At this point, prototyping would no longer be a necessity since requirements would be much easier to concretize. This process of standards adoption is the reason that I believe that the role of software engineering in multimedia is still evolving.

Another important aspect of the MICE environment which was not heavily emphasized in this paper is that the TAOML Builder is based on a formal method – the use of a Symbol Relation Grammar to represent TAOs [2,7]. In the future, we will be exploring the use of this grammar to prove properties of the design such as the fulfillment of certain QOS measures. This seems to be a worthwhile avenue to explore due to the time-dependent nature of multimedia [14].

Bibliography

[1] Adjeroh DA, Nwosu KC (1997) Multimedia database management – requirements and issues. IEEE MultiMedia 4:24-33

[2] Arndt T, Cafiero A, Guercio A (1997) Symbol Relation Grammars for Teleaction Objects. Technical Report, Dipartimento di Informatica ed Applicazioni, University of Salerno

[3] Chang HJ, Hou TY, Hsu A, Chang SK (1995) The management and application of tele-action objects. ACM Multimedia Systems J. 3: 204-216

[4] Chang SK (1996) Extending visual languages for multimedia. IEEE MultiMedia 3:18-26

[5] Chang SK (1995) Towards a theory of active index. Journal of Visual Languages and Computing 6:101-118

[6] Connell JL, Shafer LB (1989) Structured Rapid Prototyping. Yourdon Press.

[7] Ferrucci F, Pacini G, Satta G, Sessa MI, Tortora G, Tucci M, Vitiello G (1996) Symbol relation grammars: a formalism for graphical languages. Information and Computation 131:1-46

[8] Hirakawa M Call for papers first international workshop on multimedia software engineering. URL: `http://www.huis.hiroshima-u.ac.jp/~hirakawa/MSE98/mse98.html`

[9] Liao W, Li VOK (1998) Synchronization of distributed multimedia systems with user interactions. ACM Multimedia Systems J. 6:196-205

[10] Little TDC (1993) Interval-based conceptual models for time-dependent multimedia. IEEE Transactions on Knowledge and Data Engineering 5:246-260

[11] Muelhaeuser M (1996) Services, frameworks, and paradigms for distributed multimedia applications. IEEE MultiMedia 3:48-61

[12] Pazandak P, Srivastava J (1997) Evaluating object DBMSs for multimedia. IEEE MultiMedia 4:34-49

[13] Saiedian H (1996) An invitation to formal methods. Computer 29:16-17

[14] Shih TK, Davis RE (1997) IMMPS: a multimedia presentation design system. IEEE MultiMedia 4:67-78

[15] Somerville I (1992) Software Engineering. Fourth Edition, Addison-Wesley.

[16] Weitzman L, Wittenburg K (1996) Grammar-based articulation for multimedia document design. ACM Multimedia Systems J 4:99-111